



**INTER-FACULTY MASTER PROGRAM
on COMPLEX SYSTEMS and NETWORKS**
SCHOOL of MATHEMATICS
SCHOOL of BIOLOGY
SCHOOL of GEOLOGY
SCHOOL of ECONOMICS
ARISTOTLE UNIVERSITY of THESSALONIKI



Master Thesis

Quantum Statistics and Data Analysis

Anestis Kosmidis

SUPERVISOR: Ioannis Antoniou, Professor, AUTH

Thessaloniki , November 2019





ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
στα ΠΟΛΥΠΛΟΚΑ ΣΥΣΤΗΜΑΤΑ και ΔΙΚΤΥΑ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ
ΤΜΗΜΑ ΒΙΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΓΕΩΛΟΓΙΑΣ
ΤΜΗΜΑ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ



ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κβαντική Στατιστική και Ανάλυση Δεδομένων

Ανέστης Κοσμίδης

ΕΠΙΒΛΕΠΩΝ: Ιωάννης Αντωνίου, Καθηγητής Α.Π.Θ.

Εγκρίθηκε από την Τριμελή Εξεταστική Επιτροπή την 28η Σεπτεμβρίου 2019.

.....
Ι. Αντωνίου
Καθηγητής Α.Π.Θ.

.....
Κ. Χατζησάββας
Διδάκτορας Α.Π.Θ.

.....
Χ. Μπράτσας
Διδάκτορας Α.Π.Θ.

Θεσσαλονίκη , Νοέμβριος 2019



.....
Ανέστης Κ. Κοσμίδης

Πτυχιούχος Μαθηματικός Α.Π.Θ.

Copyright © Ανέστης Κ. Κοσμίδης, 2019

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευτεί ότι εκφράζουν τις επίσημες θέσεις του Α.Π.Θ.



ABSTRACT

Criteria for selecting quantum probability models versus Kolmogorov probability models are usually expressed in terms of inequalities formulated from the original Bell inequality. If the inequality is violated, Kolmogorov probability should be replaced by quantum probability. We discuss these criteria and the maximal violations and we illustrate the applicability with data sets. We explore the possibility to apply quantum probability models and related statistical algorithms in three selected applications, namely: 1) violation of Wigner-d’Espagnat inequality by a simple data set, 2) k-Means clustering versus quantum clustering, 3) Multiple linear regression versus quantum regression.

KEY WORDS

Quantum Probability, Bell inequalities, Quantum statistics, Quantum Learning, Data Analysis

ΣΥΝΟΨΗ

Τα κριτήρια για την επιλογή μοντέλων κβαντικής πιθανότητας έναντι μοντέλων πιθανότητας Kolmogorov εκφράζονται συνήθως με βάση τις ανισότητες που διαμορφώνονται από την αρχική ανισότητα Bell. Αν η ανισότητα παραβιάζεται, τότε η πιθανότητα Kolmogorov πρέπει να αντικατασταθεί από την κβαντική πιθανότητα. Συζητούνται αυτά τα κριτήρια και οι μέγιστες παραβιάσεις τους και παρουσιάζεται η εφαρμογή τους στα σύνολα δεδομένων. Ερευνάται η δυνατότητα εφαρμογής μοντέλων κβαντικής πιθανότητας και συναφών στατιστικών αλγορίθμων σε τρεις επιλεγμένες εφαρμογές, δηλαδή: 1) παραβίαση της ανισότητας των Wigner-d'Espagnat από ένα απλό σύνολο δεδομένων, 2) k-Means μέθοδος clustering εναντίον κβαντικής μεθόδου clustering, 3) Πολλαπλή γραμμική παλινδρόμηση εναντίον κβαντικής παλινδρόμησης.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Κβαντική Πιθανότητα, Ανισότητες Bell, Κβαντική Στατιστική, Κβαντική Μάθηση, Ανάλυση Δεδομένων



CONTENTS

ΠΕΡΙΛΗΨΗ	10
ACKNOWLEDGEMENTS	16
PROLOGUE	17
CHAPTER 1: QUANTUM PROBABILITY AND BELL'S INEQUALITY	19
1.1: BRIEF PRESENTATION OF BELL'S INEQUALITY:.....	19
Local correlations:.....	19
Quantum correlations:.....	19
Proof of Bell's theorem:.....	20
Bell's thought experiment:.....	25
1.2: CRITERIA INDICATING THE PRESENCE OF QUANTUM PROBABILITY	30
Criterion: Bell's theorem for locality (1964)	30
Criterion: Bell-Kochen-Specker inequality (1967)	38
Criterion: Bipartite Bell inequalities – Clauser-Horne-Shimony-Holt inequality (1969) ...	41
Criterion: Kochen-Specker non-contextuality (1969)	42
Criterion: Roy-Singh local-realist inequalities (1979)	44
Criterion: Mermin inequality (3-qubits) (1981)	45
Criterion: Leggett-Garg macroscopic realism (1985)	46
Criterion: Svetlichny inequality (1987).....	48
Criterion: Mermin polynomials and Mermin inequalities (1990)	49
Criterion: Ardehali inequality (1992).....	52
Criterion: Bell inequality for two qutrits (2002)	52
Criterion: Bell inequality for three qutrits (2004)	53
Criterion: Bell inequality for n qutrits (2004)	55
Criterion: Bell inequality for qutrits by Acin (2004)	56
Criterion: Multi-setting tight Bell Inequality for 2 qubits from Collins, Gisin (2004)	57
Criterion: Multi-setting tight Bell Inequality for 2 qutrits (2004).....	58
Criterion: CGLMP inequality for qudits (2004).....	59
Criterion: Bell function – Buhrman and Massar inequality – Bell operator (2005).....	63
Criterion: Tight Bell Inequalities for many qubits (2006).....	66
Criterion: Experimentally testable state-independent quantum contextuality (2008)	70
Criterion: Zohren and Gill inequality for 2x2xd Bell scenario (2008).....	74
Criterion: Bell ratio – Bell operator (2016).....	75
Criterion: CGLMP inequality – Bell operator C_{nsd} (2016).....	77

Criterion: 3-qubits set Bell inequalities (2017)	80
Criterion: multi-qubits set Bell inequalities (2017).....	82
Criterion: Coefficient matrix for Bell inequalities (2018).....	83
Comments on the criteria:	85
CHAPTER 2: SELECTED APPLICATIONS OF QUANTUM STATISTICS FOR THE ANALYSIS OF DATA SETS.....	87
2.1 Example: 3-Qubits Wigner-d’Espagnat inequality	87
Construction of a counterexample:.....	92
2.2 Example: Quantum probability in macroworld. Vessels of water (Aerts, Aerts, Broekaert, Gabora, 2000).....	95
2.3 Example: Quantum probability in cognition. Cats (Aerts, Aerts, Broekaert, Gabora, 2000)	101
CHAPTER 3: QUANTUM MACHINE LEARNING VARIABLES AND CAVEATS.....	107
How to put classical data in superposition:	115
Problems:.....	116
Quantum Assisted Machine Learning:	116
Challenges in QAML:	119
CHAPTER 4: QUANTUM CLUSTERING	123
Quantum Computing Shor’s Algorithm and Hierarchical Clustering Technique:	123
Dynamic Quantum Clustering:.....	129
The Galaxies example:	131
Quantum Meila-Shi Clustering Algorithm vs k-Means Clustering Algorithm:	134
Artificially-created data set of random points data set:.....	135
Rock Crabs example:	136
Quantum Clustering Algorithms – Results visually:.....	137
QUANTUM CLUSTERING: NOVEL APPLICATION.....	149
Comments on the Results.....	162
CHAPTER 5: QUANTUM REGRESSION	163
Quantum Least Squares Regression:.....	170
Quantum Linear Regression Algorithm from classical data set:.....	172
Quantum Circuit Learning:	178
Regression Example in Python:	182
Benefits and Limitations:	187
QUANTUM REGRESSION : NOVEL APPLICATION	189
Multiple Linear Regression:	191
Logistic Regression:	195



Quantum Regression:	200
Comments on the Results.....	203
EPILOGUE	205
APPENDICES.....	207
Appendix A: Dynamic Quantum Clustering Algorithm	207
Appendix B: Detailed description of Dynamic Quantum Clustering method by Horn, Weinstein and Marvin	208
Appendix C: Quantum Clustering Algorithm by Horn and Gottlieb	210
Appendix D: Quantum Clustering first visualization (Matlab).....	212
Appendix E: Quantum Clustering second visualization (Matlab).....	216
Appendix F: Multiple Linear Regression (Python)	226
Appendix G: Logistic Regression (Python)	228
Appendix H: Quantum Regression (Python).....	232
Appendix I: Classical methods of Clustering Analysis (k-Means, Hierarchical and Model- based method)	236
Appendix J: Quantum Clustering Statistics	244
BIBLIOGRAPHY:	257

Η κβαντομηχανική προβλέπει ότι μπορεί να υπάρχουν "μη τοπικές" σχέσεις μεταξύ σωματιδίων. Το 1969 όμως ο Ιρλανδός φυσικός John Bell απέδειξε ότι οι θεωρίες κρυφών μεταβλητών που διατηρούν τις παραδοχές της τοπικότητας και του ντετερμινισμού δεν μπορούν να πετύχουν τις προβλέψεις της κβαντικής φυσικής. Ο Bell χρησιμοποίησε μια ανισότητα που αν παραβιαζόταν, τότε δεν μπορεί να ισχύει καμιά θεωρία κρυφών μεταβλητών που διατηρούν την τοπικότητα. Το θεώρημα του Bell, που διατυπώθηκε το 1964, θεωρείται μία από τις πιο θεμελιώδεις επιστημονικές ανακαλύψεις του 20^{ου} αιώνα, διότι κατέδειξε νέους πόρους στο μαθηματικό πλαίσιο της κβαντικής πιθανότητας με απρόσμενες δυνατότητες εφαρμογών, οι οποίες υλοποιούνται σήμερα. Βασισμένο στο νοητικό πείραμα των Einstein, Podolsky και Rosen (EPR), μετατόπισε τα επιχειρήματα σχετικά με τη φυσική πραγματικότητα των κβαντικών συστημάτων από το χώρο της φιλοσοφίας σε εκείνο της πειραματικής φυσικής. Ο Bell και άλλοι έδειξαν ότι είναι δυνατόν να διακρίνουμε μεταξύ κβαντομηχανικής και αυτών των θεωριών με τις κρυμμένες μεταβλητές χρησιμοποιώντας ένα συγκεκριμένο τύπο πειράματος, το οποίο μετράει μια παράμετρο γνωστή ως S παράμετρο. Οι τοπικές θεωρίες προβλέπουν ότι η S θα έχει πάντα τιμή μικρότερη του 2, ενώ η κβαντική πρόβλεψη δίνει $S = 2\sqrt{2}$. Όταν η S είναι μεγαλύτερη από 2, λέμε ότι παραβιάζεται η ανισότητα του Bell. Κρυφές μεταβλητές είναι οι προκαθορισμένες ιδιότητες των πραγμάτων, άγνωστες και στην προκειμένη περίπτωση, τοπικές διότι αλληλεπιδρούν με το φράγμα της ταχύτητας του φωτός (Massen, 2019).

Η ανισότητα του Bell είναι η εξής:

$$1 + C(b,c) \geq |C(a,b) - C(a,c)|$$

Η ανισότητα Bell σημαίνει ότι σε μια στατιστική συλλογή, αν μια ομάδα έχει την ιδιότητα A και δεν έχει την ιδιότητα B, μία άλλη ομάδα έχει την ιδιότητα B και όχι την ιδιότητα C, τότε το πλήθος των δύο ομάδων θα είναι μεγαλύτερο ή ίσο από το πλήθος μίας τρίτης ομάδας που έχει την ιδιότητα A και όχι την C (Massen, 2019). Με λίγα λόγια συμπεραίνουμε ότι αν υπάρχει κβαντική συσχέτιση (κβαντική διεμπλοκή), τότε η παραπάνω ανισότητα δεν ικανοποιείται.



Με αφετηρία την ανισότητα του Bell, συνοψίζουμε τα κριτήρια που διακρίνουν την κβαντική πιθανότητα από την πιθανότητα Kolmogorov. Στο πρώτο κεφάλαιο παρουσιάζουμε τα κριτήρια αυτά με βάση τη χρονολογική σειρά εμφάνισής τους. Παρουσιάζουμε επίσης τις γενικεύσεις τους ως προς το πλήθος των qubits και το πλήθος των διαστάσεών τους, καθώς και τις μέγιστες δυνατές παραβιάσεις τους. Στο δεύτερο κεφάλαιο, αναδιατυπώνουμε τα κριτήρια αυτά, ώστε να χρησιμοποιηθούν στη στατιστική ανάλυση συνόλων δεδομένων. Στόχος μας είναι να παρουσιάσουμε κριτήρια, των οποίων η παραβίαση συνεπάγεται την ύπαρξη κβαντικής συσχέτισης (διεμπλοκής) μεταξύ των μεταβλητών, άρα της κβαντικής πιθανότητας. Επιπλέον, παρουσιάζουμε καινοτόμες και πρωτότυπες εφαρμογές των κριτηρίων αυτών πάνω σε πραγματικά σύνολα δεδομένων. Παρουσιάζουμε το κριτήριο Wigner-d'Espagnat, το οποίο αναφέρει ότι αν θεωρήσουμε 3 γεγονότα A, B, Γ του δειγματοχώρου Ω , τότε ισχύει η ανισότητα:

$$P(A \cap B) + P(B^c \cap \Gamma) \geq P(A \cap \Gamma).$$

Με βασικές γνώσεις της Θεωρίας Πιθανοτήτων παρουσιάζουμε την απόδειξη της παραπάνω ανισότητας. Η σημασία της ανισότητας των Wigner-d'Espagnat έγκειται στο γεγονός ότι η παραβίασή της από ένα παρατηρούμενο φαινόμενο, αποτελεί ένδειξη ότι το παρατηρούμενο φαινόμενο δεν μπορεί να μοντελοποιηθεί μέσω της πιθανότητας Kolmogorov και μοντελοποιείται μέσω κβαντικής πιθανότητας.

Τα κβαντικά συστήματα μπορεί να παρουσιάζουν συσχετίσεις που δεν έχουν ανάλογο στις κλασσικές θεωρίες. Παρουσιάζουμε ένα απλό και σαφές παράδειγμα εφαρμογής του κριτηρίου Wigner-d'Espagnat σε ένα σύνολο δεδομένων. Ελέγχουμε την ανισότητα των Wigner-d'Espagnat από δεδομένα 5226 παρατηρήσεων 3 δυαδικών μεταβλητών, R, S και E που αντιστοιχούν σε επιβεβαίωση χρήσης ουσίας (R), ύπαρξη φαινοτυπικού χαρακτηριστικού (S) και εργασιακή σχέση (E). Ελέγχουμε τυχόν συσχετίσεις μεταξύ των τριών μεταβλητών ανά δύο. Παρουσιάζουμε λοιπόν τους πίνακες συνάφειας και κοινής εμπειρικής πιθανότητας. Στη συνέχεια εφαρμόζουμε το νόμο της ολικής πιθανότητας με την εξής αντιστοιχία γεγονότων:

$$\{R = 0\} = A, \{R = 1\} = A^c, \{S = 0\} = B, \{S = 1\} = B^c, \{E = 0\} = \Gamma, \{E = 1\} = \Gamma^c.$$

Αποδεικνύουμε ότι ο νόμος της ολικής πιθανότητας ισχύει σε κάθε περίπτωση.

Ελέγχουμε αν ισχύει η ανισότητα των Wigner-d'Espagnat και προκύπτει ότι όντως ισχύει. Αφού η ανισότητα των Wigner-d'Espagnat δεν παραβιάζεται, δεν υπάρχει ένδειξη κβαντικής συσχέτισης μεταξύ των μεταβλητών R , S και E .

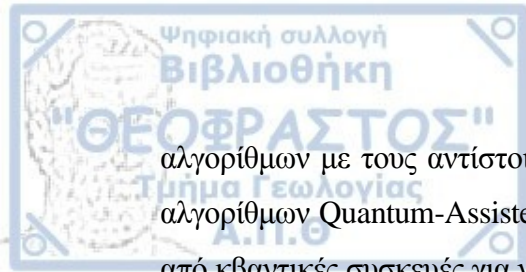
Έπειτα κατασκευάζουμε ένα νέο σύνολο δεδομένων με αυτές τις τρεις μεταβλητές με νέες συχνότητες. Ελέγχουμε πάλι τυχόν συσχετίσεις μεταξύ των τριών μεταβλητών ανά δύο. Παρουσιάζουμε τους νέους πίνακες συνάφειας και κοινής εμπειρικής πιθανότητας. Εφαρμόζουμε το νόμο της ολικής πιθανότητας με την ίδια αντιστοιχία γεγονότων όπως παραπάνω.

Αποδεικνύουμε ότι ο νόμος της ολικής πιθανότητας ισχύει σε κάθε περίπτωση.

Ελέγχουμε αν ισχύει η ανισότητα των Wigner-d'Espagnat. Ωστόσο, αυτή τη φορά προκύπτει ότι δεν ισχύει η ανισότητα των Wigner-d'Espagnat. Αφού η ανισότητα των Wigner-d'Espagnat παραβιάζεται για το παράδειγμα δεδομένων που κατασκευάσαμε, το σύστημα των τριών δυαδικών μεταβλητών R , S , E θα πρέπει να μοντελοποιηθεί μέσω της κβαντικής πιθανότητας.

Ο σκοπός του δευτέρου κεφαλαίου αυτής της εργασίας είναι να προσφέρει ένα απλό παράδειγμα βάσης δεδομένων, η οποία επειδή παραβιάζει την ανισότητα των Wigner-d'Espagnat, απαιτεί κβαντική μοντελοποίηση. Δεν βρήκαμε στη βιβλιογραφία παρόμοιο απλό παράδειγμα. Η πρακτική αξία αυτού του παραδείγματος είναι ότι όταν έχουμε μία βάση δεδομένων για την ανάλυση τριών δυαδικών μεταβλητών, πριν προχωρήσουμε σε στατιστική ανάλυση, πρέπει να εξετάσουμε αν ισχύει η ανισότητα των Wigner-d'Espagnat. Εάν ικανοποιείται η ανισότητα των Wigner-d'Espagnat (πρώτη περίπτωση), τότε προχωρούμε στη μοντελοποίηση μέσω της θεωρίας πιθανοτήτων κατά Kolmogorov, ενώ αν παραβιάζεται (δεύτερη περίπτωση), τότε είμαστε υποχρεωμένοι να κατασκευάσουμε μοντέλο κβαντικής πιθανότητας. Όταν κάνουμε στατιστική ανάλυση, συλλέγουμε δεδομένα χωρίς να γνωρίζουμε την πραγματική προέλευσή τους ούτε τον τύπο μοντελοποίησης που πρέπει να ακολουθήσουμε. Γνωρίζοντας όμως το Θεώρημα των Wigner-d'Espagnat, έχουμε ένα κριτήριο επιλογής κλασσικής ή κβαντικής μοντελοποίησης.

Στο τρίτο κεφάλαιο της έρευνάς μας εξετάζουμε τα πλεονεκτήματα και τα μειονεκτήματα της κβαντικής μηχανικής μάθησης. Παρουσιάζουμε μεθόδους κωδικοποίησης των κλασσικών δεδομένων σε κβαντικές καταστάσεις. Τίθεται το πρόβλημα τοποθέτησης των κλασσικών δεδομένων σε υπέρθεση και πώς μπορεί να αντιμετωπιστεί. Συγκρίνουμε την υπολογιστική πολυπλοκότητα των κβαντικών



αλγορίθμων με τους αντίστοιχους κλασσικούς αλγορίθμους. Επισημαίνουμε την κλάση αλγορίθμων Quantum-Assisted Machine Learning (QAML), οι οποίοι χρησιμοποιούνται από κβαντικές συσκευές για να αντιμετωπιστούν δεδομένα πολλών διαστάσεων συνεχών μεταβλητών.

Συζητούμε για την κβαντική μνήμη τυχαίας προσπέλασης (Quantum Random Access Memory), η οποία είναι μια κβαντική συσκευή που μπορεί να κωδικοποιεί σε υπέρθεση N κλασσικά διανύσματα d -διαστάσεων σε $\log(N d)$ qubits με υπολογιστική πολυπλοκότητα $O(\log(N d))$. Επιπλέον, παρουσιάζουμε τις προκλήσεις που αντιμετωπίζει η κβαντική μνήμη τυχαίας προσπέλασης (QRAM).

Στο τέταρτο κεφάλαιο, παρουσιάζουμε από τη βιβλιογραφία ένα παράδειγμα που συγκρίνει τους αλγορίθμους Quantum Computing Shor's Algorithm και την ιεραρχική μέθοδο clustering για ένα σύνολο δεδομένων ασθενών με καρκίνο. Παρουσιάζουμε επίσης τον αλγόριθμο Dynamic Quantum Clustering και ένα παράδειγμα εφαρμογής του σε γαλαξίες. Στη συνέχεια συγκρίνουμε τους αλγορίθμους Quantum Meila-Shi Clustering Algorithm και k-Means Clustering Algorithm σε ένα παράδειγμα συνόλου δεδομένων από είδη καβουριών. Εξηγούμε τους λόγους, για τους οποίους τα αποτελέσματα κβαντικής μεθόδου ταξινόμησης (quantum clustering) είναι καλύτερα σε ορισμένες περιπτώσεις σε σχέση με τα αντίστοιχα αποτελέσματα των κλασσικών μεθόδων ταξινόμησης (k-Means clustering).

Σε επιλεγμένη εφαρμογή εφαρμόζουμε πρώτα κλασσικούς αλγορίθμους για την ταξινόμηση δεδομένων, δηλαδή τη μέθοδο k-Means, την ιεραρχική μέθοδο, καθώς και τη μέθοδο Model-based που συνδυάζει κριτήρια Bayes και εκτίμηση μέγιστης πιθανοφάνειας σε ένα πραγματικό σύνολο δεδομένων, όπως η απουσία των ατόμων από τη δουλειά τους. Στη συνέχεια στο ίδιο σύνολο δεδομένων εφαρμόζουμε τον πρωτότυπο αλγόριθμο κβαντικής ταξινόμησης (quantum clustering) και παρουσιάζουμε στατιστικά αποτελέσματα και οπτικοποίηση. Ο αλγόριθμος αυτός μπορεί να εφαρμοστεί σε κάθε σύνολο δεδομένων. Επιχειρούμε να συγκρίνουμε τα στατιστικά αποτελέσματα των κλασσικών και κβαντικών μεθόδων ταξινόμησης, ώστε να βρούμε την προτιμότερη μέθοδο για το συγκεκριμένο σύνολο δεδομένων.

Στο πέμπτο κεφάλαιο, παρουσιάζουμε από τη βιβλιογραφία την κλάση αλγορίθμων Quantum-inspired Machine Learning για παλινδρόμηση. Αυτοί οι αλγόριθμοι μηχανικής μάθησης βασίζονται σε κάποια κβαντικά θεωρητικά στοιχεία, αλλά δεν

απαιτούν μια κβαντική συσκευή για την εφαρμογή των αλγορίθμων. Παρουσιάζουμε ένα νέο αλγόριθμο παλινδρόμησης βασισμένο στη κβαντική μηχανική και στη θεωρητική σύνδεση μεταξύ κβαντικών ερμηνειών και αλγορίθμων μηχανικής μάθησης. Συγκρίνουμε τους αλγορίθμους κβαντικής παλινδρόμησης (Quantum-Inspired Ensemble Linear Regressors) και γραμμικής παλινδρόμησης (Random Ensemble Linear Regressors) με βάση το μέσο τετραγωνικό σφάλμα των παρατηρήσεων και τις τυπικές αποκλίσεις του και παρατηρούμε ότι σε ορισμένες περιπτώσεις είναι προτιμότερος ο κβαντικός αλγόριθμος σε σχέση με τον αντίστοιχο κλασσικό. Στη βιβλιογραφία έχει επίσης αναπτυχθεί αλγόριθμος κβαντικής παλινδρόμησης ελαχίστων τετραγώνων (Quantum Least Squares Regression algorithm). Αναλύουμε το ποσοστό σφάλματος (error rate) και την υπολογιστική πολυπλοκότητα του αλγορίθμου αυτού. Παρουσιάζουμε ακόμη έναν κβαντικό αλγόριθμο γραμμικής παλινδρόμησης (Quantum Linear Regression Algorithm) βασισμένο σε κλασσικά σύνολα δεδομένων.

Επιπρόσθετα, παρουσιάζουμε την κβαντική μάθηση κυκλωμάτων (Quantum Circuit Learning), όπου περιγράφουμε τη θεωρία και τις λεπτομέρειες του αλγορίθμου κβαντικής παλινδρόμησης που χρησιμοποιούμε παρακάτω για την πρωτότυπη εφαρμογή στην Python. Ο αλγόριθμος αυτός ενώνει δύο τομείς, Quantum Computing και Μηχανική Μάθηση. Παρουσιάζουμε ένα απλό παράδειγμα κβαντικής παλινδρόμησης στην Python από τη βιβλιογραφία. Επισημαίνουμε ακόμη τα πλεονεκτήματα και τους περιορισμούς των αλγορίθμων που ανήκουν στην κβαντική μάθηση κυκλωμάτων.

Σε επιλεγμένη εφαρμογή εφαρμόζουμε πρώτα κλασσικούς αλγορίθμους για την παλινδρόμηση δεδομένων, δηλαδή τη μέθοδο πολλαπλής γραμμικής παλινδρόμησης και τη λογιστική παλινδρόμηση σε ένα πραγματικό σύνολο δεδομένων, όπως το σύστημα βαθμολογιών σε ένα σχολείο. Στη συνέχεια στο ίδιο σύνολο δεδομένων εφαρμόζουμε τον πρωτότυπο αλγόριθμο κβαντικής παλινδρόμησης (quantum regression) και παρουσιάζουμε στατιστικά αποτελέσματα και οπτικοποίηση. Ο αλγόριθμος αυτός μπορεί να εφαρμοστεί σε κάθε σύνολο δεδομένων. Επιχειρούμε να συγκρίνουμε τα στατιστικά αποτελέσματα των κλασσικών και κβαντικών μεθόδων παλινδρόμησης, ώστε να βρούμε την προτιμότερη μέθοδο για το συγκεκριμένο σύνολο δεδομένων. Σημειώνουμε ότι αυτός ο αλγόριθμος κβαντικής παλινδρόμησης στην Python αναπτύχθηκε από το αμερικανικό πανεπιστήμιο MIT, ωστόσο όταν τον εφαρμόσαμε στο σύνολο δεδομένων μας, δεν λειτούργησε. Μετά από συγκεκριμένες τροποποιήσεις, ο αλγόριθμος έτρεξε και απέδωσε τα στατιστικά αποτελέσματα της παλινδρόμησης. Για την οπτικοποίηση των αποτελεσμάτων βασίζομαστε στον αλγόριθμο του David Horn που αναπτύχθηκε στο



πρόγραμμα Matlab. Σημειώνουμε ότι και ο συγκεκριμένος αλγόριθμος παρουσίασε πολλά προβλήματα στην εμφάνιση των αποτελεσμάτων, ωστόσο μετά από συγκεκριμένες τροποποιήσεις, ο αλγόριθμος έτρεξε και απέδωσε τις ζητούμενες γραφικές παραστάσεις.

Για τις εφαρμογές της έρευνάς μας χρησιμοποιούνται οι γλώσσες προγραμματισμού R και Python, καθώς και το στατιστικό πρόγραμμα Matlab.

Στόχος μας είναι να συγκρίνουμε, αν είναι δυνατόν, τα αποτελέσματα κβαντικής μηχανικής μάθησης με τα αντίστοιχα της κλασσικής μηχανικής μάθησης. Ωστόσο αυτό δεν είναι πάντα εύκολο. Αλλά είναι ένα πρώτο βήμα στην κβαντική μηχανική μάθηση και στη σύνδεσή της με τις παραβιάσεις των ανισοτήτων του Bell. Είμαστε αισιόδοξοι ότι μπορούμε να βρούμε πραγματικά σύνολα δεδομένων, τα οποία παρουσιάζουν κβαντικές συσχετίσεις (διεμπλοκή) και για τα οποία μπορεί να αποδειχθεί στο εγγύς μέλλον ότι η εφαρμογή μεθόδων κβαντικής μηχανικής μάθησης είναι προτιμότερη από την εφαρμογή μεθόδων κλασσικής μηχανικής μάθησης.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Ioannis Antoniou of the Department of Statistics and Operational Research in School of Mathematics at Aristotle University of Thessaloniki. The door to Professor Antoniou office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. Thanks are also due to Dr. K. Chatzisavvas, who taught us the implementation of quantum processing, to Mr P. Tzounakis for his remarks and to Dr. Ch. Bratsas for his insightful remarks on machine learning.

Finally, my very profound gratitude goes to my parents, Kostas and Despina, my brother, Kyriakos, and to Georgia, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This Thesis would be a much more difficult task without their support. Thank you!



The goal of this work is to examine if there are criteria which we could implement in order to select and apply quantum statistical analysis instead of classical statistical analysis on a given data set. Data sets become larger and more complex. The analysis of Big Data has been recently appreciated as the fourth scientific paradigm (Hey, Tansley, Tolle, 2009). We consider quantum statistical analysis as a way to deal with Big Data. We shall investigate whether employing quantum computers we could handle better the information, which comes from a data set. With Quantum Machine Learning we could get results for a large data set by using new methods and graphical representations in the fields of Medicine, Physics, Chemistry and Cosmology. Our analysis indicates that the quantum method is more useful for certain data sets from the perspective of computational complexity, huge amount of data, data visualization, correlations which could not come from the classical methods such as regression and clustering.

Bell's inequalities are the first criterion to check if the classical representation is violated and so we have to use the quantum statistics. Criteria based on Bell's inequalities are presented in Chapter 1. The fundamental Bell's inequality (CHSH) is referred to 2 qubits and if it is violated, then there is quantum entanglement.

The knowledge so far is not satisfactory, because Bell's inequalities are referred to 2 qubits. However, a large data set contains many variables, so the need arises to search and investigate new criteria. Classical models may not fit well to data, so we search for new quantum models. For example, we could implement linear or logistic regression for a data set. However, it may exist a corresponding quantum regression method under certain circumstances, which could be better than the other methods for the specific data set. New quantum methods of information processing include Quantum Machine Learning.

In Chapter 2 we present selected applications on Bell inequality criteria for real data sets. Starting from the Bell's inequality for 2 qubits (CHSH), we find out new inequalities – criteria and generalizations with respect to the number of qubits and the dimensions of qudits. We find out the bounds of maximal violation between classical – quantum statistics for the above criteria – inequalities. We present calculation examples of the above criteria and their purpose of existence. Moreover, we simulate our own

example for a data set for different number of qubits and different dimensions of qudits. We use Python for the quantum regression implementation. We use R for the quantum clustering implementation. We use Matlab for the visualization of the quantum clustering results. Finally, we make comparisons between classical and quantum statistical methods in terms of the results and conclusions. The methodology is one from the first times it is applied and especially my own implementations, because we try to figure out if there are criteria which motivate us to select and apply quantum statistics instead of classical statistics in a data set. For the implementation of the above methodology we need a large data set with many variables and many observations for the cases of 2-qubits, 3-qubits, 2-qudits and qudits.

In Chapter 3, we present the benefits and caveats of Quantum Machine Learning.

In Chapter 4 and 5, we present novel applications on Quantum Clustering and Quantum Regression. Data has to be suitable for the classical algorithms, such as multiple linear regression and clustering, so that the corresponding quantum methods of regression and clustering could be also implemented later to compare the results.

There are available data sets suitable for my implementations. The data sets are downloaded from the UCI library (https://archive.ics.uci.edu/ml/data_sets.php, 21-03-2019). We have permission to process them, according to the directions of the above library. The data sets seem to be sufficient for the above methodology. The data is expected to be as large as the number of dimensions of the data sets we use in our implementations. However, not all the data sets are in workable form, since some variables contain data types as characters and for that reason I have to transform them into arithmetic characters to process them. We note the code of quantum regression in Python firstly introduced by MIT university and the code in Matlab by David Horn. However, when we implemented them in our data sets, these codes were not working. Therefore, we processed and developed these codes and finally we get our results.

CHAPTER 1: QUANTUM PROBABILITY AND BELL'S INEQUALITY

1.1: BRIEF PRESENTATION OF BELL'S INEQUALITY:

A “local” theory is defined as a one where the outcomes of an experiment on a system are independent of the actions performed on a different system which has no causal connection with the first. For example, the temperature of this room is independent on whether we choose to wear purple socks today (Bell, 1987).

A “realistic” theory is defined as one whose experiments reveal pre-existing properties. In other words, in a realistic theory the position, momentum and spin of an electron exist and we simply measure them. (Wheeler, Zurek, 1983).

Local correlations:

We can now form an idea of what locality means. A hidden variables theory usually assumes that there exist some other variables, λ , on which the outcomes a and b depend. This hidden factors can account for the correlations between Alice's and Bob's experiments by having a joint causal influence on the two. The probability to obtain values a, b given the measurement contexts M_A, M_B for Alice and Bob correspondingly and the hidden variables λ , is $p(ab|M_A M_B, \lambda)$ (1.1) (Brunner, Cavalcanti, Pironio, Scarani, Wehner, 2014).

Locality means simply that the local measurements are independent:

$$p(ab|M_A M_B, \lambda) = p(a|M_A, \lambda) p(b|M_B, \lambda).$$

Quantum correlations:

To define quantum behaviors, we need to define a state ρ_{AB} shared by the two parties, and measurement operators, M_A and M_B , acting on the Hilbert spaces where Alice's and Bob's part of the shared state belongs H_A and H_B , respectively. The conditional probability (1.1) is:

$p(ab|M_A M_B) = \text{tr}(\rho_{AB} M_A \otimes M_B)$, where M_A and M_B are the projection operators on the subspaces corresponding to the values a, b of the Hilbert spaces H_A, H_B of the systems A, B (Mackey 2004, Nielsen & Chuang, 2010, Diósi, 2011, Wilde, 2013).

This is a rephrasing of Born's rule. If the density operator ρ_{AB} is the pure state ψ on the tensor product space H_A, H_B , the conditional probability is simplified as follows:

$$p(ab|M_A M_B) = \langle \psi | M_A \otimes M_B | \psi \rangle.$$

Proof of Bell's theorem:

We will use the Bell inequality. Suppose we have two identical objects, namely they have the same properties. Suppose also that these properties are predetermined (counterfactual definiteness) and not generated by their measurement, and that the determination of the properties of one object will not influence any property of the other object (locality) (Mermin 1981, Preskill, 2018).

We will only need three properties A, B and C that can each take two values: "0" and "1". For example, if the objects are coins, then $A = 0$ might mean that the coin is gold and $A = 1$ that the coin is copper (property A , material), $B = 0$ means the coin is shiny and $B = 1$ it is dull (property B , texture), and $C = 0$ means the coin is large and $C = 1$ it is small (property C , size).

Suppose we do not know the properties because the two coins are a gift in two wrapped boxes. We only know the gift is two identical coins, but we do not know whether they are two gold, shiny, small coins ($A = 0, B = 0, C = 1$) or two copper, shiny, large coins ($1, 0, 0$) or two gold, dull, large coins ($1, 1, 0$), etc. We do know that the properties "exist" (namely, they are counterfactual and predetermined even if we cannot see them directly) and they are local (namely, acting on one box will not change any property of the coin in the other box: the properties refer separately to each coin). These are quite reasonable assumptions for two coins! Our ignorance of the properties is expressed through probabilities that represent either our expectation of finding a property (Bayesian view), or the result of performing many repeated experiments with boxes and coins and averaging over some possibly hidden variable, typically indicated with the letter λ , that determines the property (frequentist view).

For example, we might say the gift bearer will give me two gold coins with a 20% probability (Mermin et al., 1981).

Bell's inequality refers to the correlation among measurement outcomes of the properties. We call $P_{same}(A, B)$ the probability that the properties A of the first object and B of the second are the same: A and B are both 0 (the first coin is gold and the second is shiny) or they are both 1 (the first is copper and the second is dull). For example, $P_{same}(A, B) = \frac{1}{2}$ tells us that with 50% chance $A = B$ (namely they are both 0 or both 1). Since the two coins have equal counterfactual properties, this also implies that with 50% chance we get two gold shiny coins or two copper dull coins. Note that the fact that the two coins have the same properties means that $P_{same}(A, A) = P_{same}(B, B) = P_{same}(C, C) = 1$: if one is made of gold, also the other one will be, or if one is made of copper, also the other one will be, etc.

Under the conditions that three arbitrary two-valued properties A, B, C satisfy counterfactual definiteness and locality, and that $P_{same}(X, X) = 1$, for $X = A, B, C$ (i.e. the two objects have same properties), the following inequality among correlations holds:

$$P_{same}(A, B) + P_{same}(A, C) + P_{same}(B, C) \geq 1 \quad (1.2)$$

namely, a Bell inequality. The proof of such inequality is given graphically in Figure 1 below (Mermin et al., 1981).

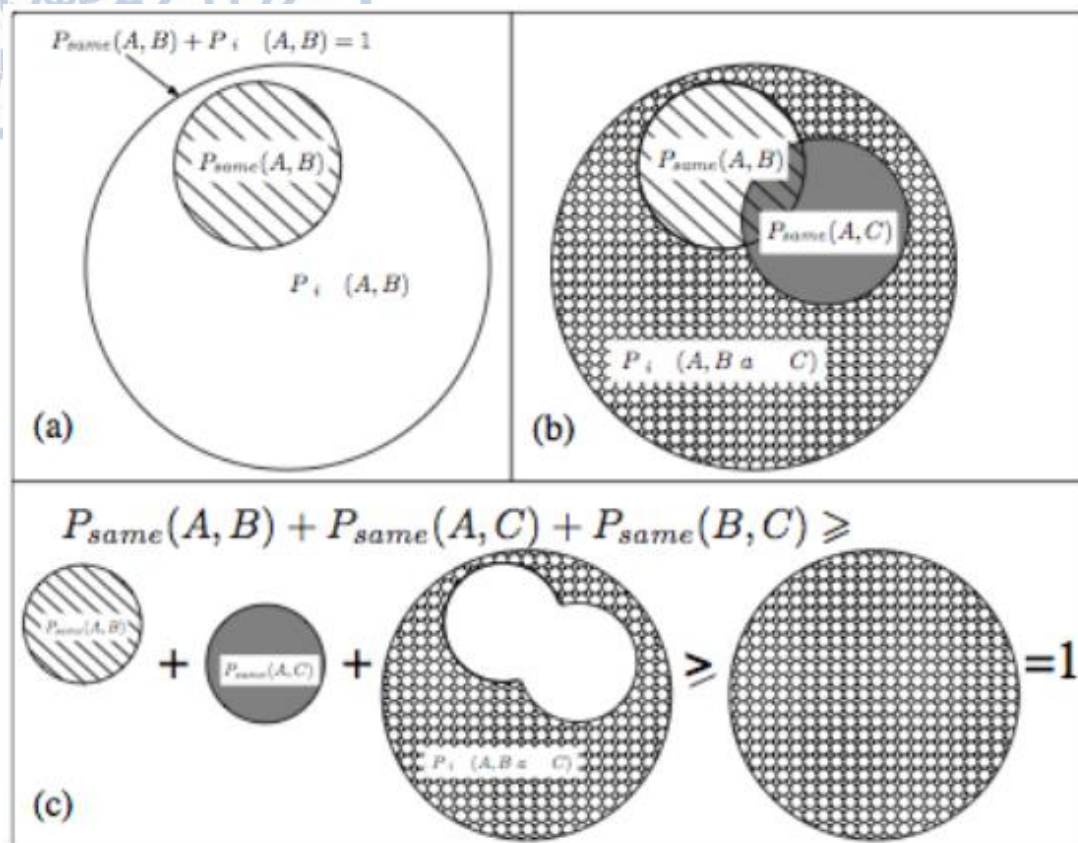


Figure 1: Proof of Bell inequality (1.2) using areas to represent probabilities. (a) The dashed area represents the probability that property A of the first object and B of the second are equal (both 1 or both 0): $P_{\text{same}}(A, B)$. The white area represents the probability that they are different: $P_{\text{diff}}(A, B)$. The whole circle has area $1 = P_{\text{same}}(A, B) + P_{\text{diff}}(A, B)$. (b) The gray area represents the probability that A and C are equal, and the non-gray area represents the probability that A and C are different. If A of the first object is different from both B and C of the second (dotted area), then B and C of the second object must be the same. Hence, the probability that B and C are the same must be larger than (or equal to) the dotted area: since B is the same for the two objects, $P_{\text{same}}(B, C)$ must be larger than (or equal to) the dotted area. (c) The quantity $P_{\text{same}}(A, B) + P_{\text{same}}(A, C) + P_{\text{same}}(B, C)$ is hence larger than (or equal to) the sum of the dashed + gray + dotted areas, which is in turn larger than (or equal to) the full circle of area 1: this proves the Bell inequality (1.2). The reasoning fails if we do not employ counterfactual properties, for example if complementarity prevents us from assigning values to both properties B and C of the second object. It also fails if we employ non-local properties, for example if a measurement of B on an object to find its value changes the value of A of the other object (Mermin et al., 1981).

The inequality basically says that the sum of the probabilities that the two properties are the same if we consider respectively A and B, A and C, B and C must be larger than one. This is intuitively clear: since the two coins have the same properties, the sum of the probabilities that the coins are gold and shiny, copper and dull, gold and large, copper and small, shiny and small, dull and large is greater than one: all the combinations have been counted, possibly more than once (Mermin et al., 1981).

In Figure 2 the events to which the probabilities represented by the Venn diagrams of Figure 1 refer are made explicit. This is true, of course, only if the two objects have same counterfactual properties and the measurement of one does not affect the outcome of the other. If we lack counterfactual properties, we cannot infer that the first coin is shiny only because we measured the second to be shiny, even if we know that the two coins have the same properties: without counterfactual definiteness, we cannot even speak of the first coin's texture unless we measure it. Moreover, if a measurement of the second coin's texture can change the one of the first coin (non-locality) again we cannot infer the first coin's texture from a measurement of the second: even if we know that the initial texture of the coins was the same, the measurement on the second may change such property of the first (Mermin et al., 1981).

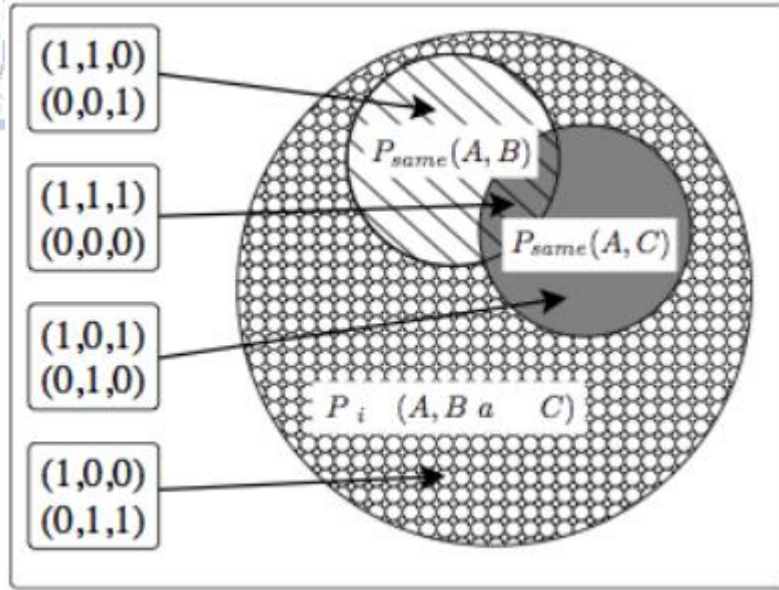


Figure 2: Explicit depiction of the properties whose probabilities are represented by the areas of the Venn diagrams in **Figure 1**. The properties are represented by a triplet of numbers (A, B, C) that indicate the (counterfactual, local) values of the properties A , B , and C for both objects. Note that in the dotted area A must be different from both B and C , so that B and C must be equal there (B and C are equal also in the intersection between the two smaller sets, but that is irrelevant to the proof) (Mermin et al., 1981).

To prove Bell's theorem, we now provide as a counter example a quantum system that violates the above inequality. Consider two two-level systems (qubits) in the joint entangled state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, and consider the 2 valued properties A , B , and C obtained by projecting the qubit on the states

$$A: \begin{cases} |a_0\rangle \equiv |0\rangle \\ |a_1\rangle \equiv |1\rangle \end{cases} \quad B: \begin{cases} |b_0\rangle \equiv \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \\ |b_1\rangle \equiv \frac{\sqrt{3}}{2}|0\rangle - \frac{1}{2}|1\rangle \end{cases} \quad C: \begin{cases} |c_0\rangle \equiv \frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle \\ |c_1\rangle \equiv \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \end{cases}$$

where it is easy to check that $|b_1\rangle$ is orthogonal to $|b_0\rangle$ and $|c_1\rangle$ is orthogonal to $|c_0\rangle$. It is also easy to check that

$$|\Phi^+\rangle = \frac{|a_0 a_0\rangle + |a_1 a_1\rangle}{\sqrt{2}} = \frac{|b_0 b_0\rangle + |b_1 b_1\rangle}{\sqrt{2}} = \frac{|c_0 c_0\rangle + |c_1 c_1\rangle}{\sqrt{2}}$$

so that the two qubits have the same properties, namely $P_{\text{same}}(A, A) = P_{\text{same}}(B, B) = P_{\text{same}}(C, C) = 1$: the measurement of the same property on both qubits always yields the same outcome, both 0 or both 1.

We are now ready to calculate the quantity on the left of Bell's inequality (1.2). We just write the state $|\Phi^+\rangle$ in terms of the eigenstates of the properties A, B, and C. e.g., it is easy to find the value of $P_{same}(A, B)$ if we write:

$$|\Phi^+\rangle = \frac{|a_0\rangle(|b_0\rangle + \sqrt{3}|b_1\rangle) + |a_1\rangle(\sqrt{3}|b_0\rangle - |b_1\rangle)}{2\sqrt{2}}$$

In fact, the probability of obtaining 0 for both properties is the square modulus of the coefficient of $|a_0\rangle|b_0\rangle$, namely, $|1/2\sqrt{2}|^2 = 1/8$, while the probability of obtaining 1 for both is the square modulus of the coefficient of $|a_1\rangle|b_1\rangle$, again 1/8.

Hence, $P_{same}(A, B) = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$. Analogously, we find that $P_{same}(A, C) = \frac{1}{4}$ and that $P_{same}(B, C) = \frac{1}{4}$ by expressing the state respectively as:

$$|\Phi^+\rangle = \frac{|a_0\rangle(|c_0\rangle + \sqrt{3}|c_1\rangle) - |a_1\rangle(\sqrt{3}|c_0\rangle - |c_1\rangle)}{2\sqrt{2}}$$

$$|\Phi^+\rangle = \frac{(|b_0\rangle + \sqrt{3}|b_1\rangle)(|c_0\rangle + \sqrt{3}|c_1\rangle) - (\sqrt{3}|b_0\rangle - |b_1\rangle)(\sqrt{3}|c_0\rangle - |c_1\rangle)}{4\sqrt{2}}$$

Summarizing, we have found

$$P_{same}(A, B) + P_{same}(A, C) + P_{same}(B, C) = \frac{3}{4} < 1$$

which violates Bell's inequality (1.2).

This proves Bell's theorem: all local counterfactual theories must satisfy inequality (1.2) which is violated by quantum mechanics. Then, quantum mechanics cannot be a local counterfactual theory: it must either be non-counterfactual (as in the Copenhagen interpretation) or non-local (as in the de Broglie-Bohm interpretation) (Mermin et al., 1981). ■

Bell's thought experiment:

Bell considered a setup in which two observers, Alice and Bob, perform independent measurements on a system S prepared in some fixed state. Each observer

has a detector with which to make measurements. On each trial, Alice and Bob can independently choose between various detector settings. Alice can choose a detector setting a to obtain a measurement M_A and Bob can choose a detector setting b to measure M_B . After repeated trials Alice and Bob collect statistics on their measurements and correlate the results (Bell, 1987).

There are two key assumptions in Bell's analysis: (1) each measurement reveals an objective physical property of the system and (2) a measurement taken by one observer has no effect on the measurement taken by the other.

In the language of probability theory, repeated measurements of system properties can be regarded as repeated sampling of random variables. One might expect that measurements by Alice and Bob to be somehow correlated with each other: the random variables are assumed to not be independent, but linked in some way. Nonetheless, there is a limit to the amount of correlation one might expect to see. The Bell inequality expresses that maximum amount of correlation one can expect (Bell, 1987).

A version of the Bell inequality appropriate for this example is given by Clauser, Horne, Shimony and Holt, and is called the CHSH form:

$$C(M_A, M_B) + C(M_A, M'_B) + C(M'_A, M_B) - C(M'_A, M'_B) \leq 2 \quad (1.3) ,$$

where C denotes correlation and M_A, M'_A refer to measurement settings for Alice and M_B, M'_B refer to measurement settings for Bob (Bell, 1987).

The CHSH inequality involves two settings for Alice and two settings for Bob. Now we denote by A_i the measurement settings for Alice, B_j the measurement settings for Bob, a the outcome of the measurement setting for Alice and b the outcome of the measurement setting for Bob. Let us take the eigenvalues of both A_i and B_j to be ± 1 , and let E_{ij} denote the expectation value for measurement settings i and j respectively:

$$E_{ij} = \langle A_i B_j \rangle = \sum_{a,b} ab \cdot p(ab|A_i B_j) \quad (1.4) \text{ (Clauser, Horne, Shimony, Holt, 1969)}$$

The inequality (1.3) then reads:

$$S = E_{00} + E_{01} + E_{10} - E_{11} \leq 2 \quad (1.5)$$

with 2 being the maximum value of S allowed by local realist theories.

The maximum quantum value is

$$S = 2\sqrt{2} > 2 \quad (1.6) \text{ (Clauser, Horne, Shimony, Holt, 1969)}$$

Equation (1.6) illustrates the content of Bell's theorem, establishing the nonlocal character of quantum theory. All bipartite Bell inequalities that involve two dichotomic measurements on both parties are equivalent (up to permutations of inputs and outputs) to the CHSH (Clauser, Horne, Shimony, Holt, 1969).

We will now prove the bounds of CHSH. To prove the local bound we assign values to the expectation values of the operators, maximizing S . We keep in mind that, for local behaviors, it holds that $\langle A_i B_j \rangle = \langle A_i \rangle \langle B_j \rangle$.

There are 4^2 possible assignments, and to find the maximum value one needs simply to go over them (see Table 1). But it is easy to see that the value $S = 2$ cannot be exceeded. We maximize the terms that come into S with a plus sign by assigning the value +1 to each A_i and B_j , thus maximizing each term. Since the last term, $A_1 B_1$ which comes into S with a minus sign is also 1, the total value of S is 2 in this scenario. If we, on the contrary, minimize the negative term, by assigning opposite sign values to A_1 and B_1 , the positive term is also minimized, and the total value of S is again 2 (Clauser, Horne, Shimony, Holt, 1969).

$\langle A_0 \rangle$	$\langle A_1 \rangle$	$\langle B_0 \rangle$	$\langle B_1 \rangle$	E_{00}	E_{01}	E_{10}	E_{11}	S
1	1	1	1	1	1	1	1	2
1	1	1	-1	1	-1	1	-1	2
1	1	-1	1	-1	1	-1	1	-2
1	1	-1	-1	-1	-1	1	1	-2
1	-1	1	1	1	1	-1	-1	2
1	-1	1	-1	1	-1	-1	1	-2
1	-1	-1	1	-1	1	1	-1	2
1	-1	-1	-1	-1	-1	1	1	-2
-1	1	1	1	-1	-1	1	1	-2
-1	1	1	-1	-1	1	1	-1	2
-1	1	-1	1	1	-1	-1	1	-2
-1	1	-1	-1	-1	-1	1	1	-2
-1	-1	1	1	-1	-1	-1	-1	2
-1	-1	1	-1	-1	1	-1	1	-2
-1	-1	-1	1	1	-1	1	-1	2
-1	-1	-1	-1	1	1	1	1	2

Table 1: the local values of S

In quantum mechanics, we can choose a state and some operators such that, when plugging them in equation (1.4), we obtain a behavior that violates the CHSH inequality. We give an example of such a choice here (Clauser, Horne, Shimony, Holt, 1969).

Let us take the state to be the singlet state of two qubits, $|\psi\rangle = (|01\rangle + |10\rangle) / \sqrt{2}$ and Alice's operators to be $A_0 = Z_A$ and $A_1 = X_A$, where Z_A and X_A are the Pauli operators acting on Alice's Hilbert space, in the z and x directions, respectively. We choose Bob's operators to be:

$$B_0 = \frac{-Z_B - X_B}{\sqrt{2}} \quad B_1 = \frac{-Z_B + X_B}{\sqrt{2}}$$

where Z_B and X_B are the corresponding Pauli operators on Bob's Hilbert space. We then have $\langle A_0 B_0 \rangle = \langle A_0 B_1 \rangle = \langle A_1 B_0 \rangle = 1/\sqrt{2}$ and $\langle A_1 B_1 \rangle = -1/\sqrt{2}$.

Putting these values together in S , we get $S = 2\sqrt{2} > 2$, at odds with (1.5). We have shown that quantum mechanics allows for the value $2\sqrt{2}$, thus proving Bell's theorem (Clauser, Horne, Shimony, Holt, 1969).

In order to prove that $2\sqrt{2}$ is indeed the maximum value allowed by quantum mechanics, we start by defining the operator:

$$F = A_0 B_0 + A_0 B_1 + A_1 B_0 - A_1 B_1$$

Since the eigenvalues of A_i (and B_j) are ± 1 , it follows the operators are all involutions, i.e. they all square to the identity: $A_i^2 = I_A$ and $B_j^2 = I_B$. Using this, we have:

$$F^2 = 4I_{AB} - [A_0, A_1][B_0, B_1] \quad (1.7)$$

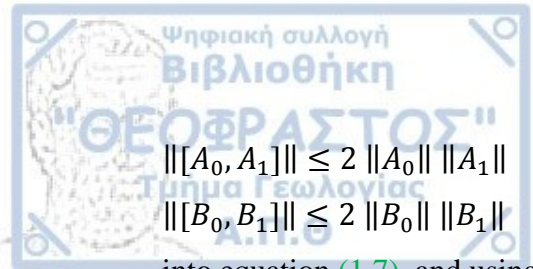
We also need to define the norm of an operator O , as following:

$$\|O\| = \sqrt{\langle O^\dagger O \rangle}$$

or simply

$$\|O\| = \sqrt{\langle O^2 \rangle}$$

since we are only concerned with Hermitian operators. Plugging the following norm inequalities:


$$\| [A_0, A_1] \| \leq 2 \| A_0 \| \| A_1 \|$$

$$\| [B_0, B_1] \| \leq 2 \| B_0 \| \| B_1 \|$$

into equation (1.7), and using the fact that $\langle A_i \rangle \leq 1$ and $\langle B_i \rangle \leq 1$, the quantum limit follows (Tsirelson, 1993).

1.2: CRITERIA INDICATING THE PRESENCE OF QUANTUM PROBABILITY

Quantum systems may exhibit correlations that go that have no analogue classical theories. We shall present criteria for selecting quantum probability instead of classical probability. The order of criteria will be presented following their historical appearance, starting from Bell's theorem for locality.

Quantum theory and classical probability are often seen as two very different theories...

Major differences

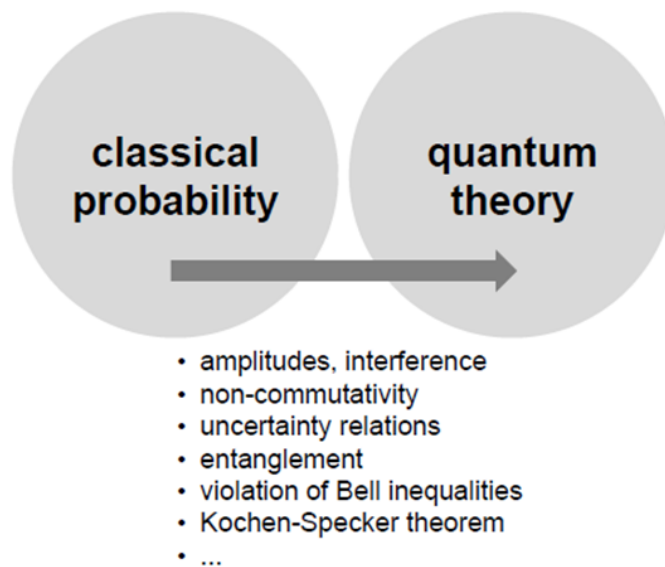


Figure 3: Major Differences between classical probability and quantum theory (Rau, 2009).

Criterion: Bell's theorem for locality (1964)

In order to ensure non-disturbance, the most stringent physical requirement is to carry out measurements on systems that cannot possibly influence each other. According to the special theory of relativity, information (or information-bearing physical carriers) propagate with a speed bounded by that of light in vacuum, c . Hence, performing measurements on two systems separated by a sufficient distance

such that no signal could reach each from the other during the performance of the experiment seems to forestall any possibility of influence between the experiments (Bell, 1964).

This is, in fact, the assumption of locality made by Bell (1964). In our setup, this corresponds to assuming a joint system, described by a density operator ρ_{AB} in the joint Hilbert space $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$, on which local measurements of the form $A_i \otimes \mathbb{1}$ and $\mathbb{1} \otimes B_j$ are performed. Thus, the CHSH-expression (John Clauser, Michael Horne, Abner Shimony, and Richard Holt) becomes:

$$\langle C_{CHSH} \rangle = \langle A_1 \otimes B_1 \rangle + \langle A_1 \otimes B_2 \rangle + \langle A_2 \otimes B_1 \rangle - \langle A_2 \otimes B_2 \rangle \quad (2.1)$$

If the locality-assumption now suffices to certify non-disturbance, and if we are furthermore justified in assigning definite values to these quantum mechanical observables, then the above expression should be bounded by 2:

$$\langle C_{CHSH} \rangle \leq 2 \quad (2.2)$$

However, if we take the state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \quad (2.3)$$

together with the observables:

$$\begin{aligned} A_1 &= \sigma_x, \quad B_1 = \frac{1}{\sqrt{2}} (\sigma_x + \sigma_z) \\ A_2 &= \sigma_z, \quad B_2 = \frac{1}{\sqrt{2}} (\sigma_x - \sigma_z) \end{aligned} \quad (2.4)$$

where σ_x and σ_z are the Pauli operators acting on Alice's Hilbert space, in the x and z directions, respectively and B_1, B_2 are the chosen Bob's operators where σ_x and σ_z are the corresponding Pauli operators on Bob's Hilbert space, then a straightforward calculation of the expectation values:

$$\langle A_i B_j \rangle = \text{tr}(A_i \otimes B_j |\Phi^+\rangle \langle \Phi^+|) \quad (2.5)$$

shows that:

$$\langle C_{CHSH} \rangle = \langle A_1 \otimes B_1 \rangle + \langle A_1 \otimes B_2 \rangle + \langle A_2 \otimes B_1 \rangle - \langle A_2 \otimes B_2 \rangle =$$

$$= \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} - \left(-\frac{1}{\sqrt{2}}\right) = 2\sqrt{2} > 2 \quad (2.6)$$

which, as can be shown, is in fact the maximum value (Tsirelson, 1993). Thus, despite the locality requirement, there are quantum mechanical measurements that do not possess a joint probability distribution.

The impossibility to reconcile a local realistic picture with the predictions of quantum mechanics is the essence of Bell's theorem. The reason for this irreconcilability does indeed lie with the failure of co-measurability of the observables: neither A_1 and A_2 , nor B_1 and B_2 are jointly measurable, since both $[A_1, A_2]$ and $[B_1, B_2]$ are nonzero. The necessity of this requirement can be seen easily by taking the square of the CHSH-operator:

$$C_{CHSH}^2 = 4 \cdot \mathbb{1} + (A_1 A_2 - A_2 A_1) \otimes (B_2 B_1 - B_1 B_2) = 4 \cdot \mathbb{1} - [A_1, A_2] \otimes [B_1, B_2] \quad (2.7)$$

where we have used that dichotomic observables square to the identity.

Hence, a violation of the CHSH-inequality is only possible if both commutators are non-vanishing (Bell, 1964).

Note, however, that while this is a necessary condition, it is not alone sufficient. For a state of the form $\rho_{prod} = |\psi_1\rangle\langle\psi_1| \otimes |\psi_2\rangle\langle\psi_2|$, since the observables A_i act nontrivially only on $|\psi_1\rangle$, while the observables B_j act only on $|\psi_2\rangle$, the correlators factorize, yielding for the expectation value of the CHSH-operator:

$$\begin{aligned} \langle C_{CHSH} \rangle &= \langle A_1 \rangle \langle B_1 \rangle + \langle A_1 \rangle \langle B_2 \rangle + \langle A_2 \rangle \langle B_1 \rangle - \langle A_2 \rangle \langle B_2 \rangle = \\ &= \langle A_1 \rangle (\langle B_1 \rangle + \langle B_2 \rangle) + \langle A_2 \rangle (\langle B_1 \rangle - \langle B_2 \rangle) \leq 2 \end{aligned} \quad (2.8)$$

since $\langle A_i \rangle, \langle B_j \rangle \leq 1$ (Bell, 1964).

This extends to convex combinations $\rho_{sep} = \sum_i p_i \rho_{prod}^i$ since each of the terms in the combination is bounded by 2. Thus, separable states, i.e. states that can be written as a convex combination of product states, cannot violate the bound

$| \langle C_{CHSH} \rangle | \leq 2$. It follows that, besides non-jointly measurable observables, entanglement is a critical resource for Bell inequality violation (Bell, 1964).

Representation of Bell's inequality with Venn's diagrams

In his original paper, Bell showed that under conditions of independence classical random variables A, B, C will satisfy:

$$|P_{\text{same}}(A, B) - P_{\text{same}}(A, C)| < P_{\text{same}}(B, C) + 1 \quad (2.9)$$

where $P_{\text{same}}(A, B)$ is the probability that the pair of random variables A, B have some identical property. Bell showed that similar measurement of entangled quantum variables can lead to a violation of the inequality (2.9) and, therefore, such violation can serve as a divide between classical and quantum variables (Bell, 1964).

However, Bell inequality is also violated in many classical situations where long-range correlations persist (Bell, 1964).

It is quite clear that a constraint on n variables will be projected as several constraints on subsets of these variables. Therefore, if we are only given the constraints on the subsets of the variables, it cannot be said if they were derived from the same function on all the variables. Consider (with Boole) three events A, B, and C. Let $P(AB) = r$, $P(BC) = s$ and $P(AC) = t$. If a Venn diagram is drawn and we write:

$$P(ABC) = \lambda, \quad P(AB\bar{C}) = \mu, \quad P(A\bar{B}C) = \nu, \quad P(\bar{A}BC) = \eta$$

$$\text{Then } \lambda + \mu = r, \quad \lambda + \eta = s, \quad \lambda + \nu = t$$

and:

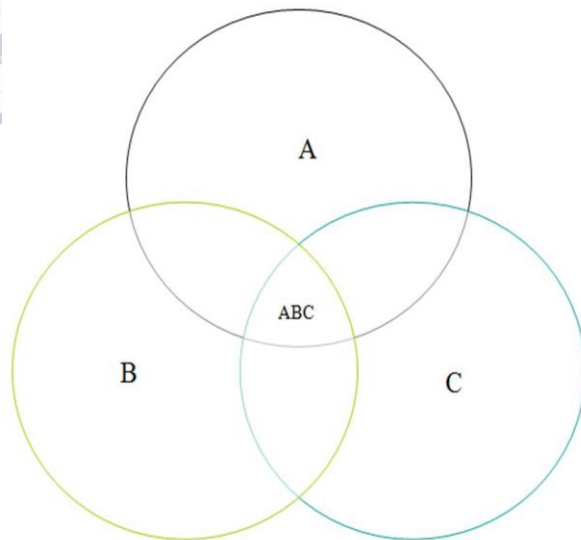


Figure 4: Venn diagram for three events A, B, C (Kak, 2013).

A straightforward computation shows that the following constraints need to be satisfied for the data to be consistent:

$$r > s + t - 1 \quad (2.10a)$$

$$s > t + r - 1 \quad (2.10b)$$

$$t > r + s - 1 \quad (2.10c)$$

These are of the form:

$$P(AB) - P(AC) > P(BC) - 1$$

which may be written as:

$$|P(AB) - P(AC)| < 1 - P(BC) \quad (2.10d).$$

This is a form similar to that of the original Bell inequality (2.9).



For the past 60 years, the best guide to that boundary has been a theorem called Bell's Inequality, but now a new paper shows that Bell's Inequality is not the guidepost it was believed to be, which means that as the world of quantum computing brings quantum strangeness closer to our daily lives, we understand the frontiers of that world less well than scientists have thought.

In this new paper, published in the July 20 edition of *Optica*, University of Rochester researchers show that a classical beam of light that would be expected to obey Bell's Inequality can fail this test in the lab, if the beam is properly prepared to have a particular feature: entanglement.

Not only does Bell's test not serve to define the boundary, the new findings don't push the boundary deeper into the quantum realm but do just the opposite. They show that some features of the real world must share a key ingredient of the quantum domain. This key ingredient is called entanglement, exactly the feature of quantum physics that Einstein labeled as spooky (Qian, Little, Howell, Eberly, 2015).

According to Joseph Eberly, professor of physics and one of the paper's authors, it now appears that Bell's test only distinguishes those systems that are entangled from those that are not. It does not distinguish whether they are "classical" or quantum.

In the forthcoming paper the Rochester researchers explain how entanglement can be found in something as ordinary as a beam of light.

Eberly explained that "it takes two to tangle." For example, think about two hands clapping regularly. What you can be sure of is that when the right hand is moving to the right, the left hand is moving to the left, and vice versa. But if you were asked to guess without listening or looking whether at some moment the right hand was moving to the right, or maybe to the left, you wouldn't know. But you would still know that whatever the right hand was doing at that time, the left hand would be doing the opposite. The ability to know for sure about a common property without knowing anything for sure about an individual property is the essence of perfect entanglement (Qian, Little, Howell, Eberly, 2015).

Eberly added that many think of entanglement as a quantum feature because "Schrodinger coined the term 'entanglement' to refer to his famous cat scenario." But their experiment shows that some features of the "real" world must share a key ingredient of Schrodinger's Cat domain: entanglement.

With this result, Eberly and his colleagues have shown experimentally "that the border is not where it's usually thought to be, and moreover that Bell's Inequalities should no longer be used to define the boundary".

The growing recognition that entanglement is not exclusively a quantum property, and does not even originate with Schrodinger's famous remark about it, prompts the examination of its role in marking the quantum-classical boundary. We have done this by subjecting correlations of classical optical fields to new Bell-analysis experiments and report here values of the Bell parameter greater than $B = 2.54$. (Qian, Little, Howell, Eberly, 2015)

This is many standard deviations outside the limit $B = 2$ established by the Clauser–Horne–Shimony–Holt Bell inequality, agreement with our theoretical classical prediction, and not far from the Tsirelson limit $B = 2.828\dots$. These results cast a new light on the standard quantum-classical boundary description, and suggest a reinterpretation of it (Qian, Little, Howell, Eberly, 2015).

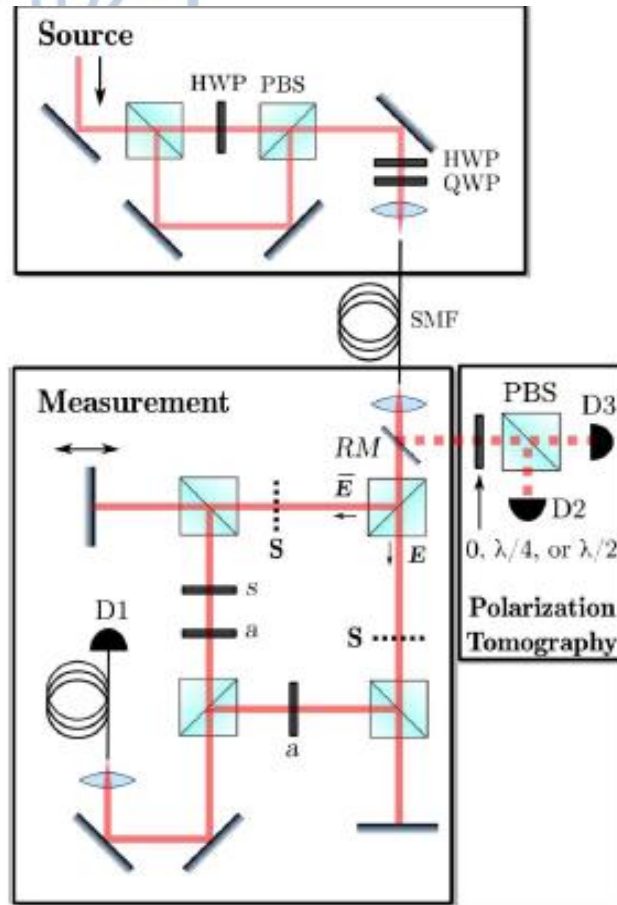


Figure 5: Experimental setup consists of a source of unpolarized light and a measurement using a modified MZ interferometer. HWP and a QWP control the polarization of the source. All beam splitters are 50:50 unless marked as a PBS. Intensities needed for obtaining the required joint projections are measured as detector D1. Shutters S independently block the arms of the interferometer in order to measure light through the arms separately. A removable mirror (RM) directs the light to a polarization tomography setup where the orthogonal components of the polarization in the basis determined by the wave plate are measured at detectors D2 and D3 (Qian, Little, Howell, Eberly, 2015).

Our theoretical sketch for the simplest case, unpolarized light, indicated that such fields or states are predicted to possess a range of correlation strengths equal to that of two-party quantum systems, that is, outside the bound $B \leq 2$ of the CHSH Bell inequality and potentially as great as $B = 2\sqrt{2}$. In our experimental test, we used light whose statistical behavior (field second-order statistics) is indistinguishable from classical, viz., the light from a broadband laser diode operating below threshold (Qian, Little, Howell, Eberly, 2015).

Our detections of whole-beam intensity are free of the heralding requirements familiar in paired-photon CHSH experiments. Repeated tests confirmed that such a field can strongly violate the CHSH Bell inequality and can attain Bell-violating levels of correlation similar to those found in tests of maximally entangled quantum systems.

One naturally asks, how are these results possible? We know that a field with classically random statistics is a local real field, and we also know that Bell inequalities prevent local physics from containing correlations as strong as what quantum states provide. But the experimental results directly contradict this. The resolution of the apparent contradiction is not complicated, but does mandate a shift in the conventional understanding of the role of Bell inequalities, particularly as markers of a classical-quantum border. Bell himself came close to addressing this point. He pointed out that even adding classical indeterminism would still not be enough for any type of hidden variable system to overcome the restriction imposed by his inequalities. This is correct as far as it goes, but fails to engage the point that local fields can be statistically classical and exhibit entanglement at the same time (Qian, Little, Howell, Eberly, 2015).

For the fields under study, the entanglement is a strong correlation that is intrinsically present between the amplitude and polarization DOFs, and it is embedded in the field from the start (as it also is embedded ab initio in any quantum states that violate a Bell inequality). The possibility of such pre-existing structural correlation is bypassed in a CHSH derivation.

Thus one sees that Bell violation is a result of entanglement due to tensor product structure (Qian, Little, Howell, Eberly, 2015).

Criterion: Bell-Kochen-Specker inequality (1967)

Firstly, we present the requirements of this criterion. The generalized Bell inequality, due to Kochen and Specker, namely, the Bell–Kochen–Specker (BKS) inequality, analyzes under what condition a single degree of freedom exhibits the

inequivalence between explanations based on classical and quantum probabilities. There is no need to consider entangled states, which requires at least two or more degrees of freedom, and neither is it necessary to have spins with space-like separation (so that the operators operating on the different spins commute). All that is needed for the BKS theorem is the existence of a certain collection of Hermitian operators.

BKS inequality for a single Spin 1 System:

The BKS inequality can be derived for a spin 1 system that is located at a single point. Since there is only one degree of freedom, the quantum state cannot be an entangled state (Kochen, Specker, 1967).

Consider the case of P_1, P_2, P_3, P_4, P_5 , namely, five commuting and non-commuting operators that are arranged in Figure 6. Let the operators be numbered periodically with $P_6 \equiv P_1$; then the commutation equations are given by the following:

$$[P_n, P_{n+1}] = 0, [P_n, P_{n+2}] \neq 0, \text{ where } n = 1, 2, \dots, 5$$

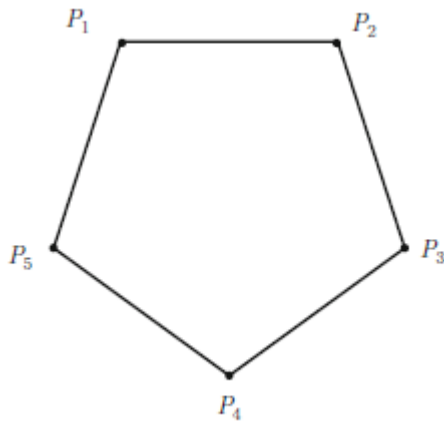


Figure 6 (Baaquie, 2013)

We assume the quantum state is described by the pure density matrix ρ given by: $\rho = |\psi\rangle\langle\psi|$ and $\text{tr}(\rho) = 1$. Then the quantum expectation value of a (Hermitian) operator B is given by: $E_q[B] = \text{tr}(\rho B)$.

We assume $P(X = x, Y = y, Z = z)$ yields the probability for the simultaneous occurrence of the sample values x, y and z of the random variables X, Y and Z respectively. We consider a function H that depends on the random variables X, Y, Z . Its (average) classical expectation value is given by:

$$E_c[H] = \int dx dy dz H(x, y, z) P(x, y, z) .$$

It can be shown that the BKS inequality for this case is given by classical probability theory and yields:

$$\sum_i E_c [P_i] \leq 2 \quad : \quad \text{BKS inequality} \quad (2.11)$$

The “contextual” inequality, obtained by evaluating the expectation value of P_i in a quantum state, is given by:

$$\sum_i E_q [P_i] \leq \sqrt{5} \quad (2.12)$$

and violates the BKS inequality given in (2.11).

BKS inequality for two Spin 1/2 System:

Consider the case of three Hermitian operators A, B , and C such that $[A, B] = 0 = [A, C]$ but $[B, C] \neq 0$ and constructed from the two spin 1/2 degrees of freedom. Since A can be simultaneously measured with other operators that commute with it, the joint probability distribution functions $p_1(A, B)$ and $p_2(A, C)$ can be measured, and which are theoretically also obtainable from quantum mechanics (Kochen, Specker, 1967).

Although not within the framework of quantum mechanics, a classical joint probability distribution function does in fact exist for A, B, C considered as classical random variables and is given as follows:

$$p(A, B, C) = \frac{p_1(A, B) p_2(A, C)}{p(A)} \quad (2.13)$$

where $\sum_B p_1(A, B) = p(A) = \sum_C p_2(A, C)$.

This construction reproduces the experimentally measurable marginal probability distribution function. One recovers, for instance, the experimentally observed $p_1(A, B)$ by summing over the outcomes for C in $p(A, B, C)$ and which

results in a cancellation of $p(A)$ on the right-hand side of (2.13), leading to the required probability $p_1(A, B)$ for the inequality (2.11) (Kochen, Specker, 1967).

Criterion: Bipartite Bell inequalities – Clauser-Horne-Shimony-Holt inequality (1969)

We first assume two particles controlled by Alice and Bob, respectively. In the framework of hidden variables, the probability that Alice obtains the outcome α and Bob the outcome β when Alice is measuring observable A and Bob is measuring observables B given that the hidden variable is λ is denoted by $p_\lambda(\alpha, \beta|A, B)$. The expectation value of the observable AB given that the hidden variable is λ is then calculated according to

$$\langle AB \rangle_\lambda = \sum_{\alpha, \beta} \alpha \beta p_\lambda(\alpha, \beta|A, B) \quad (2.14)$$

We measure the expectation value in experiment

$$\langle AB \rangle = \int d\lambda \rho(\lambda) \sum_{\alpha, \beta} \alpha \beta p_\lambda(\alpha, \beta|A, B) \quad (2.15)$$

where $\rho(\lambda)$ is a probability density on the hidden variable λ (Clauser, Horne, Shimony, Holt, 1969).

Besides the existence of hidden variables, Bell theories also assume locality. In other words, one considers hidden-variable theories in which the expectation values of Equation (2.15) can always be written in terms of probability distributions that factorize, i.e. which obey

$$p_\lambda(\alpha, \beta|A, B) = p_\lambda(\alpha|A) p_\lambda(\beta|B) \quad (2.16)$$

for any two observables A_1 and B_1 and fixed λ . Hidden-variable theories that obey Equation (2.16) are called local hidden-variable theories (LHV theories). Note that locality implies that Alice's choice of observable cannot affect Bob's outcome probabilities (no-signalling), i.e., that for any two observables B and B' that Bob measures (and any observable A and outcome α of Alice), the probabilities obey

$$p_\lambda(\alpha|A, B) = p_\lambda(\alpha|A, B')$$

where $p_{\lambda}(\alpha|A, B) = \sum_b p_{\lambda}(\alpha, \beta|A, B)$
(Holt, 1969).

(2.17) (Clauser, Horne, Shimony,

The most commonly used Bell inequality for two particles is the Clauser-Horne-Shimony-Holt (CHSH) inequality. It states that in any LHV theory, the inequality

$$\langle AB \rangle + \langle AB' \rangle + \langle A'B \rangle - \langle A'B' \rangle \leq 2 \quad (2.18)$$

holds for any observables A, A', B and B' . In quantum mechanics, this inequality can be violated by choosing $A = -X, A' = -Y, B = (X - Y)/\sqrt{2}, B' = (X + Y)/\sqrt{2}$ and the singlet state of Equation $|\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. For these choices, the left-hand side of Equation (2.18) equals $2\sqrt{2}$.

Bell inequalities can also be used for entanglement detection, as any state that violates a Bell inequality must be entangled. This can be seen by noting that for any separable state $\rho_{sep} = \sum_i p_i \rho_i^A \otimes \rho_i^B$, one has

$$\langle AB \rangle = \sum_i p_i \text{tr}(A \rho_i^A) \text{tr}(B \rho_i^B) \quad (2.19)$$

This defines an LHV model, where the locality can be seen by Equation (2.19) and therefore, it cannot violate a Bell inequality. However, the converse is not true: There are some entangled states that do not violate any Bell inequality. Finally, it is worth mentioning that Bell inequalities are, in contrast to entanglement witnesses, tools to detect entanglement independent from the observables actually measured in experiment (Clauser, Horne, Shimony, Holt, 1969).

Criterion: Kochen-Specker non-contextuality (1969)

As Bell's theorem relies on locality in order to prevent influences between different measurements, so does the theorem by Kochen and Specker (1969) rely on the notion of noncontextuality: roughly, the idea that the value of an observable A , measured simultaneously with observables B or C (with which it hence must be jointly measurable), does not depend on whether it is measured simultaneously with B or C . This is a reasonable expectation in the classical world—for instance, we do

not observe an object's color changing, depending on whether we measure it simultaneously with its shape, or with its mass.

It is again clear that this assumption holds whenever we have a joint probability distribution—as in this case, we can think of a population in which elements simply carry certain values for all observables within experimental interest, which do not mutually influence one another, and are simply revealed upon measurement (Kochen, Specker, 1967).

We consider four observables $\{A, B, C, D\}$ on a four-dimensional Hilbert space \mathcal{H}_4 . Among these observables, we have the following commutation (and hence, joint measurability) relations:

$$\begin{aligned} [A, B] &= 0 & [C, B] &= 0 \\ [A, D] &= 0 & [C, D] &= 0 \\ [A, C] &\neq 0 & [B, D] &\neq 0 \end{aligned} \quad (2.20)$$

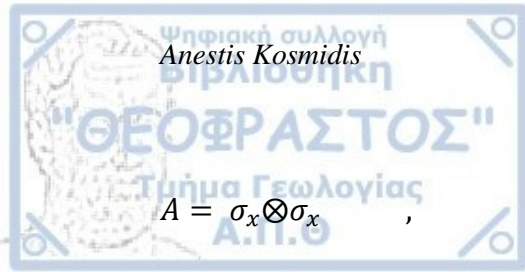
Thus, in the expression:

$$\langle C_{CHSH}^{KS} \rangle = \langle AB \rangle + \langle BC \rangle + \langle CD \rangle - \langle DA \rangle \quad (2.21)$$

only jointly measurable quantities enter in pairs. If we now assume that the value of each observable is independent of the context—that, for instance, the value of A does not depend on whether it is measured simultaneously with B or D —we again assume the presence of a joint probability distribution for all observables, and consequently, again obtain the bound $|\langle C_{CHSH}^{KS} \rangle| \leq 2$.

Now, with the identifications $A_1 \otimes \mathbb{1} = C$, $A_2 \otimes \mathbb{1} = A$, $\mathbb{1} \otimes B_1 = D$ and $\mathbb{1} \otimes B_2 = B$, the observables in Equation 1.2.4 fulfill exactly these relations. Consequently, we cannot assume that each of them yields its value independently of its context—and hence, any test of Bell's theorem is also a test of the Kochen-Specker theorem.

However, we need not appeal to entanglement, or indeed the bipartite Hilbert-space structure in order to test the Kochen-Specker theorem. For instance, we may take the observables (which are related to the observables in Equation (2.4) by a unitary rotation):



$$A = \sigma_x \otimes \sigma_x$$

$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$C = \sigma_x \otimes 1, \quad B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \quad (2.22)$$

and the (product) state :

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |10\rangle) \quad (2.23)$$

to again obtain the value $|\langle C_{CHSH}^{KS} \rangle| \leq 2$. Hence, we can view the noncontextuality of Kochen and Specker as a relaxation of Bell's locality: for any set of local observables, the commutation relations in Equation (2.20) will be automatically fulfilled, but not every set of observables fulfilling them consists of local observables on a bipartite Hilbert space (Kochen, Specker, 1967).

Criterion: Roy-Singh local-realist inequalities (1979)

Roy and Singh were the first to derive from the local-realist condition testable inequalities (RS inequalities) different from the Bell-CHSH-type inequalities. The Roy-Singh method provides an elegant method to derive local-realist inequalities. Suppose two qubits in an entangled state are intercepted by two measuring devices geographically separated from each other. The first device randomly measures property either X_1, X_2, \dots on particle 1, and the other device either property Y_1, Y_2, \dots on particle 2. Experimentally, one measures bi-partite correlations of the type $P(x_j, y_k)$, where $x_j = \pm 1$ and $y_k = \pm 1$ are measurement outcomes of measuring X_j and Y_k respectively. The assumption of local-realism (LR) entails that for each such pair of these variables, there is a deterministic hidden variable (DHV) theory whereby:

$$\langle X_j Y_k \rangle = \int_{\lambda} d\lambda \rho(\lambda) X_j(\lambda) Y_k(\lambda) \quad (2.24)$$

where λ is a “complete” or “dispersion-free” specification of the state described by underlying probability distribution $\rho(\lambda)$. Roy and Singh consider quantities of the form:

$$\left(X_1^{(1)} \pm X_2^{(1)} \pm \dots \pm X_{m_1}^{(1)} + Y_1^{(1)} \pm Y_2^{(1)} \pm \dots \pm Y_{n_1}^{(1)}\right)^2 + \left(X_1^{(2)} \pm X_2^{(2)} \pm \dots \pm X_{m_2}^{(2)} + Y_1^{(2)} \pm Y_2^{(2)} \pm \dots \pm Y_{n_2}^{(2)}\right)^2 + \dots + \left(X_1^{(q)} \pm X_2^{(q)} \pm \dots \pm X_{m_q}^{(q)} + Y_1^{(q)} \pm Y_2^{(q)} \pm \dots \pm Y_{n_q}^{(q)}\right)^2 \geq q \quad (2.25)$$

where $m_j + n_j = \text{odd}$ and the self-correlation terms, i.e., correlations between the same particle, are so arranged as to cancel out. Here $X_k^{(j)} \in \{X_1, X_2, \dots, X_n\}$ and $Y_k^{(j)} \in \{Y_1, Y_2, \dots, Y_m\}$, where m, n are positive integers and $X_j, Y_k \in \{\pm 1\}$ (Roy, Singh, 1979).

As a particular example, we consider:

$$(X_1 - Y_1 - Y_2)^2 + (X_2 - Y_1 + Y_2)^2 \geq 2 \quad (2.26)$$

Expanding the left hand side of (2.26), and using the DHV assumption Equation (2.24), we obtain:

$$\langle B \rangle \equiv \langle X_1 Y_1 \rangle + \langle X_1 Y_2 \rangle + \langle X_2 Y_1 \rangle - \langle X_2 Y_2 \rangle \leq 2 \quad (2.27)$$

which just is the CHSH inequality (2.18) (Roy, Singh, 1979).

Criterion: Mermin inequality (3-qubits) (1981)

We first assume two particles controlled by Alice and Bob, respectively. In the framework of hidden variables, the probability that Alice obtains the outcome α and Bob the outcome β , when Alice is measuring observable A and Bob is measuring observables B given that the hidden variable λ is denoted by $p_\lambda(\alpha, \beta | A, B)$. The expectation value of the observable AB given that the hidden variable is λ is then calculated according to:

$$\langle AB \rangle_\lambda = \sum_{\alpha, \beta} \alpha \beta p_\lambda(\alpha, \beta | A, B) \quad (2.28)$$

Besides the existence of hidden variables, Bell theories also assume locality. So the expectation values of Equation (2.28) can always be written in terms of probability distributions that factorize, i.e. which obey:

$$p_{\lambda}(\alpha, \beta | A, B) = p_{\lambda}(\alpha | A) p_{\lambda}(\beta | B)$$

for any two observables A_1 and B_1 and fixed λ .

Analogously, we assume three particles controlled by Alice, Bob and Charlie, respectively. In the framework of hidden variables, the probability that Alice obtains the outcome α , Bob the outcome β and Charlie the outcome γ , when Alice is measuring observable A, Bob is measuring observables B and Charlie is measuring observables C given that the hidden variable λ is denoted by $p_{\lambda}(\alpha, \beta, \gamma | A, B, C)$. We consider probability distributions which factorize fully and have the form:

$$p_{\lambda}(\alpha, \beta, \gamma | A, B, C) = p_{\lambda}(\alpha | A) p_{\lambda}(\beta | B) p_{\lambda}(\gamma | C) \quad (2.29)$$

Probability distributions of this kind, which one might call “fully local”, obey the Mermin inequality. For three qubits, the Mermin inequality is given by

$$\langle ABC \rangle - \langle AB'C' \rangle - \langle A'BC' \rangle - \langle A'B'C \rangle \leq 2 \quad (2.30)$$

where A, A', B, B', C and C' are arbitrary observables. We will present Bell inequalities with the quantum mechanical observables that yield the largest violation already plugged in, since this allows for a more compact notation. The Mermin inequality (2.30) is maximally violated for the three-qubit Greenberger-Horne-Zeilinger (GHZ) state:

$$|GHZ_3\rangle = \frac{1}{\sqrt{2}} (|000\rangle + |111\rangle) \quad (2.31)$$

In this case, the left-hand side of (2.30) has a value of 4, since every term has an absolute value of 1 with the appropriate sign. Therefore, 4 is the value of maximal violation of the inequality (2.30). This observation was also the basis for the argument by Greenberger, Horne and Zeilinger who argue that the GHZ state contradicts realism in the sense of Einstein, Podolsky and Rosen (Kafatos, 1989).

Criterion: Leggett-Garg macroscopic realism (1985)

Finally, the third option to make the non-disturbance assumption plausible, after Bell locality and Kochen-Specker contextuality, is the macroscopic realism of Leggett and Garg. Macroscopic realism is the conjunction of two postulates:

- (i) Any macroscopic system that has available to it two or more distinguishable states, is at any given time in exactly one of those states.
- (ii) It is possible, in principle, to determine which of these states the system is in at a given time, without disturbing the system or its dynamics (Leggett, Garg, 1985).

Let us thus imagine a system that has exactly two states available to it, as well as a measurement Q (which we again assume to be ± 1 -valued) that is capable of differentiating between these states. Furthermore, we measure this observable at four different points in time t_1, t_2, t_3, t_4 . Then, we observe the correlation between measurements at different points in time, and calculate the quantity:

$$\langle C_{CHSH}^{LG} \rangle = \langle Q(t_1) Q(t_2) \rangle + \langle Q(t_2) Q(t_3) \rangle + \langle Q(t_3) Q(t_4) \rangle - \langle Q(t_1) Q(t_4) \rangle \quad (2.32)$$

The assumption of macroscopic realism serves to shield a measurement at a later time from the influence of an earlier one; thus, again, we can assume a joint probability distribution for the value of Q at different times, and conclude that:

$$|\langle C_{CHSH}^{LG} \rangle| \leq 2 \quad (\text{Leggett, Garg, 1985}).$$

A difference to the previous two cases (Bell locality and Kochen-Specker non-contextuality) is now that at first sight, there is no problem with 'joint' measurability, as we just re-measure the same observable Q at different points in time. However, in general, there will be a non-trivial time-evolution of the system in between measurements.

This time-evolution is mediated by some unitary $U(t)$, producing the transformation $|\psi(0)\rangle \rightarrow |\psi(t)\rangle = U(t) |\psi(0)\rangle$ (Leggett, Garg, 1985).

To calculate the expectation value of an operator at time t , we can equally well use a time-evolved operator and the state at $t = 0$:

$$\begin{aligned} \langle A \rangle_t &= \text{tr}(A \rho(t)) = \text{tr}(A U(t) \rho(0) U^\dagger(t)) = \text{tr}(U^\dagger(t) A U(t) \rho(0)) = \\ &= \text{tr}(A(t) \rho(0)) \end{aligned} \quad (2.33)$$

This is known as the Heisenberg picture, whereas the corresponding picture in which the time evolution acts on the states instead is the Schrödinger picture. Hence, we can keep the initial state fixed, and take

$$Q_i = U^\dagger(t_i - t_0) Q(t_0) U(t_i - t_0) \quad .$$

However, this yields ‘too much’ incommensurability: in general $[Q(t_i), Q(t_j)] \neq 0$ for any pair of indices, and consequently, we do not know how to define the correlator $\langle Q(t_i) Q(t_j) \rangle$, as the simple product of both operators will typically fail to be Hermitian.

Nevertheless, for projective qubit measurements, we can go back to the definition of the correlator:

$$\langle Q_1 Q_2 \rangle = \sum_{q_k, q_l} q_k q_l \Pr(Q_1^{q_k} Q_2^{q_l}) \quad (2.34)$$

where $q_k, q_l \in \{+1, -1\}$ are the outcomes of Q_1 and Q_2 , respectively. To calculate these probabilities, the projection postulate yields:

$$\Pr(Q_1^{q_k} Q_2^{q_l}) = \left\langle \frac{1 + q_k q_1 \cdot \sigma}{2} \cdot \frac{1 + q_l q_2 \cdot \sigma}{2} \cdot \frac{1 + q_k q_1 \cdot \sigma}{2} \right\rangle \quad (2.35)$$

where q_i is the Bloch vector associated to Q_i (Leggett, Garg, 1985).

Using this to compute the correlator, one arrives at the expression (Fritz 2010):

$$\sum_{q_k, q_l} q_k q_l \Pr(Q_1^{q_k} Q_2^{q_l}) = \langle Q_1 \circ Q_2 \rangle \quad (2.36)$$

for the appropriate quantum analogue to the classical correlation functions in Equation (2.32), where the symbol \circ denotes the symmetric (Jordan) product:

$$X \circ Y = \frac{XY + YX}{2} \quad (2.37)$$

With this framework, it can again be shown that in quantum mechanics, a maximum of $|\langle C_{CHSH}^{LG} \rangle| \leq 2\sqrt{2}$ is achievable (Leggett, Garg, 1985).

Criterion: Svetlichny inequality (1987)

We assume three particles controlled by Alice, Bob and Charlie, respectively. In the framework of hidden variables, the probability that Alice obtains the outcome α , Bob the outcome β and Charlie the outcome γ , when Alice is measuring observable

A, Bob is measuring observables B and Charlie is measuring observables C given that the hidden variable λ is denoted by $p_\lambda(\alpha, \beta, \gamma | A, B, C)$. As for entanglement, there is also a notion of genuine multipartite non-locality. For three qubits, any probability distribution that cannot be written as:

$$p_\lambda(\alpha, \beta, \gamma | A, B, C) = q_1 p_\lambda(\alpha | A) p_\lambda(\beta, \gamma | B, C) + q_2 p_\lambda(\beta | B) p_\lambda(\alpha, \gamma | A, C) + q_3 p_\lambda(\gamma | C) p_\lambda(\alpha, \beta | A, B) \quad (2.38)$$

where $\sum_i q_i = 1$ and $q_i \geq 0$, is called genuine multipartite non-local. Any probability distribution that is of the form of Equation (2.38) obeys the Svetlichny inequality:

$$\langle ABC \rangle + \langle AB'C \rangle + \langle ABC' \rangle - \langle AB'C' \rangle + \langle A'BC \rangle - \langle A'B'C \rangle - \langle A'BC' \rangle - \langle A'B'C' \rangle \leq 4 \quad (2.39)$$

where A, A', B, B', C and C' are arbitrary observables.

The quantum mechanical violation is maximal for the GHZ state (Equation 2.31) of three qubits and equals $4\sqrt{2}$ for the choice $A = -X$, $A' = Y$, $B = (X + Y)/\sqrt{2}$, $B' = (X - Y)/\sqrt{2}$, $C = -X$ and $C' = Y$, where X, Y are the Pauli matrices (Svetlichny, 1987).

One way to prove that Equation (2.39) holds for genuinely non-local models is based on the realization that the inequality is a sum of two CHSH inequalities. For example, all expectation values containing A form a CHSH inequality on parties two and three and the same for all terms that include A'. Moreover, since Equation (2.39) is invariant under any permutation of particles, it has this form on any two qubits. Also, Svetlichny's inequality has been generalized to an arbitrary number of qubits (Collins, Gisin, Popescu, Roberts, Scarani, 2002).

Criterion: Mermin polynomials and Mermin inequalities (1990)

Quantum probability can be discriminated from classical probability using Bell-type inequalities. An extension of Bell inequalities to a larger number of particles

corresponds to the set of Mermin inequalities. Such inequalities should be maximally violated by GHZ-type states (Mermin, 1990).

The Mermin polynomial for 3-qubits is:

$$M_3 = (\alpha_1 \alpha_2 \alpha'_3 + \alpha_1 \alpha'_2 \alpha_3 + \alpha'_1 \alpha_2 \alpha_3) - (\alpha'_1 \alpha'_2 \alpha'_3) \quad (2.40)$$

where α_i and α'_i correspond to two different settings for the measurement of each qubit i . Each measurement can take the values $\{-1, 1\}$. Classical theories obey local realism (LR) which translates into a bound for the expectation value of the Mermin polynomial, $\langle M_3 \rangle^{LR} \leq 2$ (Mermin, 1990).

In this case, the maximum possible eigenvalue, and therefore the quantum bound, is $\langle M_3 \rangle^{QM} < 4$ (Greenberger, Horne, Shimony, Zeilinger, 1990).

The Mermin polynomial for 4-qubits is:

$$\begin{aligned} M_4 = & -(a_1 a_2 a_3 a_4) + (a_1 a_2 a_3 a'_4 + a_1 a_2 a'_3 a_4 + a_1 a'_2 a_3 a_4 + a'_1 a_2 a_3 a_4) + \\ & + (a_1 a_2 a'_3 a'_4 + a_1 a'_2 a_3 a'_4 + a_1 a'_2 a'_3 a_4 + a'_1 a_2 a_3 a'_4 + a'_1 a_2 a'_3 a_4 + \\ & a'_1 a'_2 a_3 a_4) - \\ & - (a_1 a'_2 a'_3 a'_4 + a'_1 a_2 a'_3 a'_4 + a'_1 a'_2 a_3 a'_4 + a'_1 a'_2 a'_3 a_4) - \\ & - (a'_1 a'_2 a'_3 a'_4) \end{aligned} \quad (2.41)$$

with a classical bound of $\langle M_4 \rangle^{LR} \leq 4$ and a quantum bound of $\langle M_4 \rangle^{QM} \leq 8\sqrt{2}$.

The Mermin polynomial for 5-qubits is:

$$\begin{aligned} M_5 = & -(a_1 a_2 a_3 a_4 a_5) + (a_1 a_2 a_3 a'_4 a'_5 + a_1 a_2 a'_3 a_4 a'_5 + a_1 a'_2 a_3 a_4 a'_5 + \\ & + a'_1 a_2 a_3 a_4 a'_5 + a_1 a_2 a'_3 a'_4 a_5 + a_1 a'_2 a_3 a'_4 a_5 + a'_1 a_2 a_3 a'_4 a_5 + \\ & a_1 a'_2 a'_3 a_4 a_5 + a'_1 a'_2 a_3 a_4 a_5) - \\ & - (a_1 a'_2 a'_3 a'_4 a'_5 + a'_1 a_2 a'_3 a'_4 a'_5 + a'_1 a'_2 a_3 a'_4 a'_5 + a'_1 a'_2 a'_3 a_4 a'_5 + \\ & + a'_1 a'_2 a'_3 a'_4 a'_5) \end{aligned} \quad (2.42)$$

with a classical bound of $\langle M_5 \rangle^{LR} \leq 4$ and a quantum bound of $\langle M_5 \rangle^{QM} \leq 16$ (Greenberger, Horne, Shimony, Zeilinger, 1990).



	LR	QM	EXP
3 qubits	2	4	2.85 ± 0.02
4 qubits	4	$8\sqrt{2}$	4.81 ± 0.06
5 qubits	4	16	4.05 ± 0.06

Table 2: Table of results. LR corresponds to the Local Realism bound for each Mermin inequality, QM to the Quantum bound and EXP is the experimental result (Alsina, Latorre, 2016).

There exists an entire family of n-qubit inequalities first discovered by Mermin. We present Mermin operators. Let us change the notation of observables $\{a, b, c, \dots\} \equiv \{a_1, a_2, a_3, \dots\}$, which is more convenient to treat the multipartite case. Defining $M_1 \equiv a_1$, the Mermin polynomials are obtained recursively as:

$$M_n = \frac{1}{2} M_{n-1}(a_n + a'_n) + \frac{1}{2} M'_{n-1}(a_n - a'_n) \quad (2.43)$$

where M'_k is obtained from M_k by interchanging primed and non-primed observables a_n . In particular, M_3 corresponds to the three-qubit Mermin operator:

$$M_3 = (a \otimes b \otimes c' + a \otimes b' \otimes c + a' \otimes b \otimes c) - (a' \otimes b' \otimes c') \quad (2.44)$$

where \otimes denotes the Kronecker product and the variables a, a' and b, b' are represented by Hermitian operators acting on Hilbert spaces \mathcal{H}_a and \mathcal{H}_b , respectively. For dichotomic variables the operators satisfy $a^2 = a'^2 = b^2 = b'^2 = \mathbb{I}$, because the measurement operators a, a', b and b' have eigenvalues ± 1 . M_2 corresponds to the two-qubit Mermin operator:

$$M_2 = (a \otimes b + a \otimes b' + a' \otimes b) - (a' \otimes b') \quad (2.45) \quad (\text{Mermin, 1990}).$$

For n qubits, the Mermin inequality is given by:

$$\begin{aligned} & \langle X_1 X_2 X_3 X_4 X_5 \dots X_n \rangle - \sum_{perms} \langle Y_1 Y_2 X_3 X_4 X_5 \dots X_n \rangle + \sum_{perms} \langle Y_1 Y_2 Y_3 Y_4 X_5 \dots X_n \rangle - \\ & \dots \leq \\ & \leq \begin{cases} 2^{n/2}, & \text{for even } n \\ 2^{(n-1)/2}, & \text{for odd } n \end{cases} \end{aligned} \quad (2.46)$$

where \sum_{perms} indicates a sum over all permutations of all qubits that lead to distinct terms. This Mermin inequality holds also for an arbitrary choice of observables. The maximal violation is obtained for the n-qubit GHZ state:

$$|GHZ_n\rangle = \frac{1}{\sqrt{2}} (|0 \dots 0\rangle + |1 \dots 1\rangle) \quad (2.47)$$

for which the left-hand side of the inequality (2.46) reaches a value of 2^{n-1} .

Therefore, 2^{n-1} is the maximal violation of the inequality (2.46) (Mermin, 1990).

Criterion: Ardehali inequality (1992)

The Ardehali inequality holds for the same kind of non-locality. We denote X_i the Pauli matrix σ_X acting on the i^{th} qubit, Y_i the Pauli matrix σ_Y and Z_i the Pauli matrix σ_Z acting on the i^{th} qubit. Then, the Ardehali inequality is given by:

$$\begin{aligned} & [\langle A_1 X_2 X_3 X_4 X_5 X_6 X_7 \dots X_n \rangle + \langle B_1 X_2 X_3 X_4 X_5 X_6 X_7 \dots X_n \rangle - \\ & \sum_{perms(2,\dots,n)} (\langle A_1 Y_2 X_3 X_4 X_5 X_6 X_7 \dots X_n \rangle - \langle B_1 Y_2 X_3 X_4 X_5 X_6 X_7 \dots X_n \rangle) - \\ & \sum_{perms(2,\dots,n)} (\langle A_1 Y_2 Y_3 X_4 X_5 X_6 X_7 \dots X_n \rangle + \langle B_1 Y_2 Y_3 X_4 X_5 X_6 X_7 \dots X_n \rangle) + \\ & \sum_{perms(2,\dots,n)} (\langle A_1 Y_2 Y_3 Y_4 X_5 X_6 X_7 \dots X_n \rangle - \langle B_1 Y_2 Y_3 Y_4 X_5 X_6 X_7 \dots X_n \rangle) + \\ & \sum_{perms(2,\dots,n)} (\langle A_1 Y_2 Y_3 Y_4 Y_5 X_6 X_7 \dots X_n \rangle + \langle B_1 Y_2 Y_3 Y_4 Y_5 X_6 X_7 \dots X_n \rangle) - \\ & \sum_{perms(2,\dots,n)} (\langle A_1 Y_2 Y_3 X_4 Y_5 Y_6 X_7 \dots X_n \rangle - \langle B_1 Y_2 Y_3 Y_4 Y_5 Y_6 X_7 \dots X_n \rangle) - \dots] / \sqrt{2} \leq \\ & \begin{cases} 2^{n/2}, & \text{for odd } n \\ 2^{(n-1)/2}, & \text{for even } n \end{cases} \quad (2.48) \end{aligned}$$

where $\sum_{perms(2,\dots,n)}$ denotes a sum over all permutations of qubits 2 to n that yield distinct observables. Moreover, $A_1 = (X_1 + Y_1) / \sqrt{2}$ and $B_1 = (X_1 - Y_1) / \sqrt{2}$.

The Ardehali inequality holds for arbitrary observables, but with the above observables and the GHZ state, the quantum mechanical violation is maximal and equals 2^{n-1} (Ardehali, 1992).

Criterion: Bell inequality for two qutrits (2002)

We first assume there are two parties, A and B, are allowed to perform two different three-outcome measurements, A_1 and A_2 for A, and B_1 and B_2 for B. Denoting by $P(A_i = B_j + k)$ the probability that the outcomes for parties A and B, measuring A_i and B_j , differ by k modulo d (in this case $d=3$), one can consider the following Bell inequality:

$$I_3 = P(A_1 = B_1) + P(B_1 = A_2 + 1) + P(A_2 = B_2) + P(B_2 = A_1) - P(A_1 = B_1 - 1) - P(B_1 = A_2) - P(A_2 = B_2 - 1) - P(B_2 = A_1 - 1) \leq 2 \quad (2.49)$$

The maximally entangled state of a bipartite system $|\Psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$ reads:

$$|\Psi\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |jj\rangle \quad (2.50)$$

where $|j\rangle$ are the orthonormal bases in each subsystem (Acin, Durt, Gisin, Latorre, 2002).

Criterion: Bell inequality for three qutrits (2004)

We construct a Bell inequality for coincidence probabilities on a three three-dimensional (qutrit) system. We studied above the Bell inequalities and the Clauser-Horne-Shimony-Holt (CHSH) inequality, the latter being cast into a form more amenable for experimental verification, were formulated on the simplest composite quantum system, namely, a system of two two-dimensional particles (or two qubits). Since then, Bell arguments have been generalized to more complicated situations, either for a larger number of particles or for two particles of dimension greater than two. For three two-dimensional particles, Greenberger, Horne, and Zeilinger presented an elegant argument, also known as GHZ paradox, where the conflict between classical theories and quantum mechanics was shown to be qualitatively stronger in this case than for two qubits. For N ($N > 3$) two-dimensional particles, Mermin, Belinskii, and Klyshko separately generalized the CHSH inequality and proved that the quantum violation of this inequality increases exponentially with the number of particles (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

For two particles of dimension greater than two, it was found that the CHSH inequality can be maximally violated in higher dimensional systems and this violation continues to survive in the limit of infinite dimension. Moving to higher dimension, very little is known for N-qudit systems, with $N, d > 2$. GHZ paradoxes have been generalized, and some numerical results have been presented for three- and four-qutrit systems. We present an interesting coincidence Bell inequality for three qutrits in the case for which each observer measures two non-commuting observables (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

We consider the following Bell-type scenario: three space-separated observers, denoted by A, B, and C (or Alice, Bob, and Charlie), can measure two different local observables of three outcomes, labeled by 0, 1, and 2. We denote by X_i the observable measured by party X and by x_i the outcome with $X = A, B, C$ ($x = a, b, c$). If the observers decide to measure A_1, B_1 and C_2 , the result is (0, 2, 1) with probability $p(a_1 = 0, b_1 = 2, c_2 = 1)$ (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

The set of these 8×27 probabilities gives a complete description of any statistical quantity that can be observed in such Gedanken experiment. We denote by $p(a_i + b_j + c_k = r)$ the coincidence probability:

$$p(a_i + b_j + c_k = r) = \sum_{a,b=0,1,2} p(a_i = a, b_j = b, c_k = r - a - b) \quad (2.51)$$

where all the equalities are modulo three (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

Any Local Reality description of the Gedanken experiment, must satisfy some constraints, known as Bell inequalities. The following condition is satisfied by all Local Reality theories:

$$p(a_1 + b_1 + c_1 = 0) + p(a_1 + b_2 + c_2 = 1) + p(a_2 + b_1 + c_2 = 1) + \\ + p(a_2 + b_2 + c_1 = 1) + p(a_2 + b_2 + c_2 = 0) - p(a_2 + b_1 + c_1 = 2) - \\ p(a_1 + b_2 + c_1 = 2) - p(a_1 + b_1 + c_2 = 2) \leq 3 \quad (2.52)$$

These are considerations to deterministic local models. This is because any probabilistic model can be transformed into a deterministic one by simply adding some additional variables:

$$p(a_1 + b_1 + c_1 = 0) + p(a_1 + b_2 + c_2 = 1) + p(a_2 + b_1 + c_2 = 1) + \\ + p(a_2 + b_2 + c_1 = 1) + 2p(a_2 + b_2 + c_2 = 0) - p(a_2 + b_1 + c_1 = 2) - \\ p(a_1 + b_2 + c_1 = 2) - p(a_1 + b_1 + c_2 = 2) \leq 3 \quad (2.53)$$

This is the final form for our three-qutrit Bell inequality (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

Taking $c_1 = c_2 = 0$ in Equation (2.53), one derives the two-qutrit inequality:

$$p(a_1 + b_1 = 0) + p(a_1 + b_2 = 1) + p(a_2 + b_1 = 1) + p(a_2 + b_2 = 0) - \\ - p(a_1 + b_1 = 2) - p(a_2 + b_1 = 2) - p(a_1 + b_2 = 2) - p(a_2 + b_2 = 2) \leq 2 \quad (2.54)$$

After deriving the Bell inequality, our next step will be to look for quantum states and measurements violating it. First, as initial state, we take:

$$|\psi\rangle = \frac{1}{\sqrt{3}} (|000\rangle + |111\rangle + |222\rangle) \quad (2.55)$$

which can be regarded as a generalization of the maximally entangled state of two qutrits.

For this choice of setting, and the state (2.55), all the probabilities terms with a positive sign are equal to $7/9$, while the terms with negative sign are equal to $1/9$, so the inequality gives $6 \times 7/9 - 3 \times 1/9 = 39/9 \simeq 4.33 > 3$ (Acin, Chen, Gisin, Kaszlikowski, Kwek, Oh, Zukowski, 2004).

Criterion: Bell inequality for n qutrits (2004)

The Bell inequality (2.49) is also extended to arbitrary dimension. It is shown that the combination of joint probabilities:

$$\begin{aligned}
I_d = \sum_{k=0}^{\left[\frac{d}{2}\right]-1} \left(1 - \frac{2k}{d-1}\right) [& P(A_1 = B_1 + k) + P(B_1 = A_2 + k + 1) + P(A_2 = B_2 + k) \\
& + P(B_2 = A_1 + k) - P(A_1 = B_1 - k - 1) - P(B_1 = A_2 - k) \\
& - P(A_2 = B_2 - k - 1) - P(B_2 = A_1 - k - 1)] \leq 2
\end{aligned}
\tag{2.56}$$

for Local Variable Theory Models.

Starting from Equation (2.56) we can derive the corresponding Bell operator and a larger violation is again found partially entangled states of two qudits. Table 3 summarizes these results up to $d = 8$. Note that the difference between the violation for $|\Psi\rangle$ and $|\Psi_{mv}\rangle$ increases with increase in the dimension (Acin, Durt, Gisin, Latorre, 2002).

Dimension	Violation for $ \Psi\rangle$	Maximal violation (for $ \Psi_{mv}\rangle$)	Difference (%)
3	2.8729	2.9149	1.4591
4	2.8962	2.9727	2.6398
5	2.9105	3.0157	3.6133
6	2.9202	3.0497	4.4345
7	2.9272	3.0776	5.1411
8	2.9324	3.1013	5.7588

Table 3: Violation of the inequality (2.56) for two qudits, $\mathbb{C}^d \otimes \mathbb{C}^d$, up to $d=8$. The values obtained for the maximally entangled state (2.50) and the maximal violation of the inequality corresponding to the largest eigenvalue of the Bell operator are shown (Acin, Durt, Gisin, Latorre, 2002).

Criterion: Bell inequality for qutrits by Acin (2004)

We consider the following Bell-type scenario. Three space-separated observers A, B and C can measure two different local observables of three outcomes, labeled by 0, 1 and 2. We denote by X_i the observable measured by party X and by x_i the outcome with $X = A, B, C$ ($x = a, b, c$). For example, if the observers decide to

measure A_1, B_1 and C_2 , the results is $(0, 2, 1)$ with probability $p(a_1 = 0, b_1 = 2, c_2 = 1)$. Acin proposed the following three parties Bell inequality, which in the probability formalism it reads:

$$\begin{aligned}
 & p(a_1 + b_1 + c_1 = 0) + p(a_1 + b_2 + c_2 = 1) + p(a_2 + b_1 + c_1 = 1) \\
 & + p(a_2 + b_2 + c_1 = 1) + 2p(a_2 + b_2 + c_2 = 0) \\
 & - p(a_2 + b_1 + c_1 = 2) - p(a_1 + b_2 + c_1 = 2) \\
 & - p(a_1 + b_1 + c_2 = 2) \leq 3
 \end{aligned}
 \tag{2.57}$$

The analysis here is very similar to the CGLMP case: the maximal violation is given by a quasi maximally entangled state $|\psi\rangle = (|000\rangle + \gamma |111\rangle + |222\rangle) / \sqrt{2 + \gamma^2}$ where now $\gamma \approx 1.186$. The quantum value is 4.37 (Acín, Chen, Gisin, Kaszlikowski, Kwek, Oh, Żukowski, 2004).

Criterion: Multi-setting tight Bell Inequality for 2 qubits from Collins, Gisin (2004)

Most of the inequalities mentioned above belong to the two setting Bell inequalities, i.e., they are based on the standard Bell experiment, in which each local observer is given a choice between two dichotomic observables. However, we could extend the number of measurement settings. Actually, multi-setting Bell inequalities may have many advantages in many protocols in quantum information theory (Collins, Gisin, 2004).

We focus on Bell inequality for two-particle systems. The Bell-type scenario involves only two observers and each of them measures M different local observables of two outcomes ± 1 . We denote A_i and B_j the observables on the A and B party respectively, with $i, j = 1, \dots, M$. The correlation function $Q(A_i B_j)$, in the case of a local realistic theory, is then the average values of the products $A_i B_j$ over many runs of the experiment. We also denote $Q(A_i B_j)$, $Q(A_i)$ and $Q(B_j)$ as Q_{ij} , Q_{i0} and Q_{0j} , respectively. Then the famous CHSH inequality:

$$I_{CHSH} = Q_{11} + Q_{12} + Q_{21} - Q_{22} < 2 \quad (2.58)$$

holds in any local realistic theory. The CHSH inequality is almost always the most efficient one to prove a quantum state to be nonlocal. The first Bell inequality relevant to the CHSH inequality was proposed by Collins and Gisin (2004). In the form of joint probability, their inequality for 3-setting Bell inequality for 2 qubits reads:

$$\begin{aligned} I_{CG} = & P(a_1 = 0, b_1 = 0) + P(a_1 = 0, b_2 = 0) + P(a_1 = 0, b_3 = 0) + \\ & + P(a_2 = 0, b_1 = 0) + P(a_3 = 0, b_1 = 0) - P(b_2 = 0) + \\ & P(a_2 = 0, b_2 = 0) - P(a_2 = 0, b_3 = 0) - P(a_3 = 0, b_2 = 0) - P(a_1 = 0) - \\ & 2P(b_1 = 0) \leq 0 \end{aligned} \quad (2.59)$$

After some calculations, our new two three-setting Bell inequalities are:

$$\begin{aligned} -8 \leq I_3 = & Q_{21} + Q_{12} + Q_{31} + Q_{13} + Q_{32} + Q_{23} - Q_{11} - Q_{22} + \\ & + Q_{10} + Q_{01} - Q_{20} - Q_{02} \leq 4 \end{aligned} \quad (2.60)$$

We also obtain two four-setting Bell inequalities:

$$\begin{aligned} -6 \leq I_4 = & Q_{11} + Q_{22} + Q_{12} + Q_{21} + Q_{14} + Q_{41} - Q_{24} - Q_{42} - \\ & - 2Q_{33} + Q_{31} + Q_{13} + Q_{32} + Q_{23} \leq 6 \end{aligned} \quad (2.61)$$

(Collins, Gisin, 2004).

Criterion: Multi-setting tight Bell Inequality for 2 qutrits (2004)

For two-qutrit system, the CGLMP inequality reduces to:

$$\begin{aligned} I_{CGLMP} = & [P(A_1 = B_1) + P(B_1 = A_2 + 1) + P(A_2 = B_2) + P(B_2 = A_1)] - \\ & - [P(A_1 = B_1 - 1) + P(B_1 = A_2) + P(A_2 = B_2 - 1) + P(B_2 = A_1 - 1)] \leq 2 \end{aligned} \quad (2.62)$$

where all the equalities in the probabilities are modulo 3 (Collins, Gisin, Linden, Massar, Popescu, 2002).

We can find a new three setting Bell inequality for two qutrits, which is relevant to the CGLMP inequality for two qutrits:

$$\begin{aligned}
I_3 = & -2P(a_1 + b_1 = 0) + P(a_1 + b_1 = 1) + P(a_1 + b_1 = 2) + P(a_1 + b_2 = 0) - \\
& - P(a_1 + b_2 = 2) + P(a_2 + b_1 = 0) - P(a_2 + b_1 = 2) + P(a_1 + b_3 = 1) - \\
& - P(a_1 + b_3 = 2) + P(a_3 + b_1 = 1) - P(a_3 + b_1 = 2) + P(a_2 + b_3 = 1) - \\
& - P(a_2 + b_3 = 2) + P(a_3 + b_2 = 1) - P(a_3 + b_2 = 2) + P(a_3 + b_3 = 0) - \\
& - P(a_3 + b_3 = 1) \leq 4
\end{aligned} \tag{2.63}$$

where all the equalities in the probabilities are modulo 3 (Deng, Zhou, Chen, 2009).

Criterion: CGLMP inequality for qudits (2004)

The CHSH inequality, which belongs to a class of mathematical formulations broadly termed as Bell's inequalities, allow us to distinguish the predictions of local hidden variable theories (LHVs) and theories involving non-classical correlations, specifically quantum theory in our case. In that experiment, we worked with a maximally-entangled state (Equations 2.64, 2.65) with a dimensionality of $d = 2$, i.e. the number of independent outcomes of the measurements. The maximum violation of the CHSH inequality expected for such a bipartite, two-outcome experiment is $S = 2\sqrt{2}$ (Collins, Gisin, Linden, Massar, Popescu, 2002).

Bell states:

$$\begin{aligned}
|\psi^-\rangle &= \frac{1}{\sqrt{2}} (|H\rangle_A |V\rangle_B - |V\rangle_A |H\rangle_B) \\
|\psi^+\rangle &= \frac{1}{\sqrt{2}} (|H\rangle_A |V\rangle_B + |V\rangle_A |H\rangle_B)
\end{aligned} \tag{2.64}$$

$$\begin{aligned}
|\varphi^-\rangle &= \frac{1}{\sqrt{2}} (|H\rangle_A |H\rangle_B - |V\rangle_A |V\rangle_B) \\
|\varphi^+\rangle &= \frac{1}{\sqrt{2}} (|H\rangle_A |H\rangle_B + |V\rangle_A |V\rangle_B)
\end{aligned} \tag{2.65}$$

The CHSH inequality can however be also generalized to experiments with $d > 2$ given suitable modification to correlation function E (Equation 2.66). This inequality has the feature that the expected theoretical maximum violation decreases with the increasing d and converges to the classical limit of $S = 2$ for $d = \infty$. This feature may suggests somekind of an approach to classicality with large particle

counts. A violation of less than $S = 2\sqrt{2}$ may indicate the system has $d > 2$ or it might due to the non-maximally entangled nature of the state describing the quantum system under test (Collins, Gisin, Linden, Massar, Popescu, 2002).

The correlation function is given by:

$$E(a_i, b_j) = P(a_i = +1, b_j = +1) - P(a_i = +1, b_j = -1) - P(a_i = -1, b_j = +1) + P(a_i = -1, b_j = -1) \quad (2.66)$$

In 2002, Daniel Collins, Nicolas Gisin, Noah Linden, Serge Massar, and Sandu Popescu came up with a set of Bell's inequalities which came to be known collectively as the CGLMP inequality. The inequality is generalized for arbitrary high-dimensional bipartite systems with two measurement settings and d outcomes on each side. What this means is that the violation increases with the dimensionality d of the system.

In a bipartite system, suppose that both parties, Alice (A) and Bob (B) each can carry out two possible measurements, A_1 or A_2 , and B_1 or B_2 , respectively. Each measurement may have d possible outcomes denoted by $0, \dots, d-1$ (Figure 7). The expression for the CGLMP expression can then be written as:

$$I_d \equiv \sum_{k=0}^{\lfloor \frac{d}{2} \rfloor - 1} \left(1 - \frac{2k}{d-1}\right) \{ [P(A_1 = B_1 + k) + P(B_1 = A_2 + k + 1) + P(A_2 = B_2 + k) + P(B_2 = A_1 + k)] - [P(A_1 = B_1 - k - 1) + P(B_1 = A_2 - k) + P(A_2 = B_2 - k - 1) + P(B_2 = A_1 - k - 1)] \} \quad (2.67)$$

where $d \geq 2$. For any dimensionality d , the CGLMP inequality has the classical limit of $I_d \leq 2$ (Collins, Gisin, Linden, Massar, Popescu, 2002).

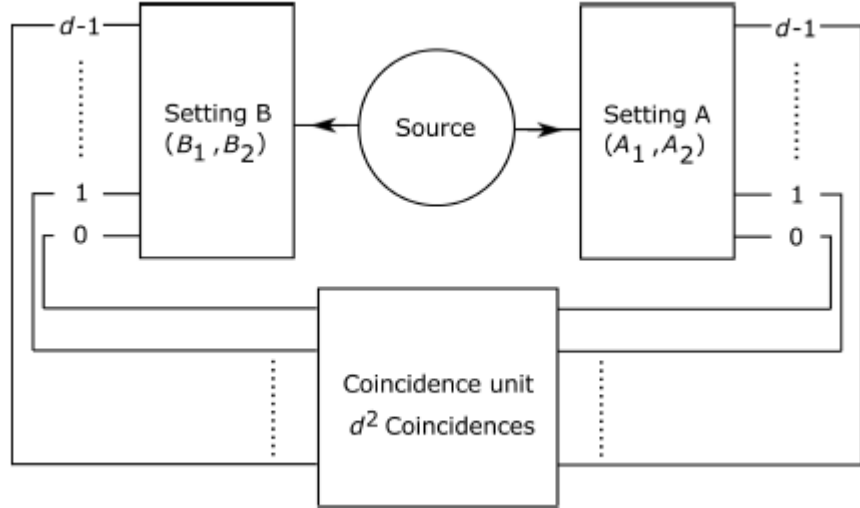


Figure 7: A d-dimensional quantum system with two measurement settings A_1 and A_2 or B_1 and B_2 , and d outcomes on each side. The four different combinations of settings give in total of $4d^2$ possible outcome of coincidence patterns which can be used for calculating the CGLMP inequality (Shun, 2015).

For bipartite system with two outcomes on each side, i.e. $d = 2$, the CGLMP inequality expression I_2 can be written as:

$$I_2 = [P(A_1 = B_1) + P(B_1 = A_2 + 1) + P(A_2 = B_2) + P(B_2 = A_1)] - \\ - [P(A_1 = B_1 - 1) + P(B_1 = A_2) + P(A_2 = B_2 - 1) + P(B_2 = A_1 - 1)] \quad (2.68)$$

which further expands to:

$$I_2 = P(A_1 = 0, B_1 = 0) + P(A_1 = 1, B_1 = 1) + P(A_2 = 0, B_1 = 1) + \\ + P(A_2 = 1, B_1 = 0) + P(A_2 = 0, B_2 = 0) + P(A_2 = 1, B_2 = 1) + \\ + P(A_1 = 0, B_2 = 0) + P(A_1 = 1, B_2 = 1) - P(A_1 = 0, B_1 = 1) - \\ - P(A_1 = 1, B_1 = 0) - P(A_2 = 0, B_1 = 0) - P(A_2 = 1, B_1 = 1) - \\ - P(A_2 = 0, B_2 = 1) - P(A_2 = 1, B_2 = 0) - P(A_1 = 0, B_2 = 1) - \\ - P(A_1 = 1, B_2 = 0) = \\ = E(A_1, B_1) + E(A_2, B_2) + E(A_1, B_2) - E(A_2, B_1) = S \quad (2.69)$$

thus recovering the original expression for the CHSH inequality for the same dimensionality (Collins, Gisin, Linden, Massar, Popescu, 2002).

Maximal Violation:

As the dimensionality of the Hilbert space increases, the maximal violation for a maximally-entangled state:

$$|\Phi_d^+\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |j\rangle_A \otimes |j\rangle_B \quad (2.70)$$

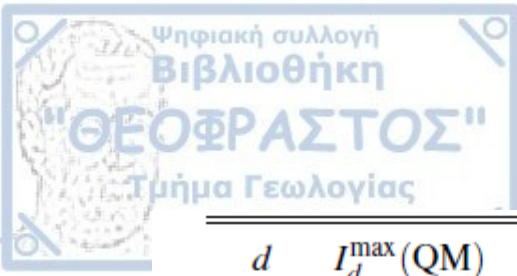
also increases.

We mention that for $d > 2$, the inequality I_d does not give the maximum violation for a maximally-entangled state. Paradoxically, a maximum violation for I_d only occurs for the case of a non-maximally entangled state (Collins, Gisin, Linden, Massar, Popescu, 2002).

We assess the CGLMP inequality on the case of a bipartite system with dimensionality $d = 4$, with two measurement settings at each party. This is the minimum dimensionality where the behaviors of CHSH and CGLMP inequality diverges. A and B can each perform two possible measurements, A_1 or A_2 and B_1 or B_2 , respectively. Each measurement will yield 4 possible outcomes, giving a total of 64 joint outcomes when all 4 possible combinations of A and B settings are considered.

The CGLMP expression I_d in Equation (2.67) is equivalent to the I_{22dd} expression:

$$I_{22dd} = \frac{d-1}{2d} (I_d - 2) \quad (2.71) \text{ (Collins, Gisin, Linden, Massar, Popescu, 2002)}$$



d	$I_d^{\max}(\text{QM})$	$I_d^{ \Phi_d^+\rangle}$	$I_{22dd}^{ \Phi_d^+\rangle}$
2	2.8284	2.8284	0.20711
3	2.9149	2.8729	0.29098
4	2.9727	2.8962	0.33609
5	3.0157	2.9105	0.36422
6	3.0497	2.9202	0.38342
7	3.0776	2.9272	0.39736
8	3.1013	2.9324	0.40793
9	3.1217	2.9365	0.41622
10	-	2.9398	0.42291
100	-	2.9668	0.47856
1000	-	2.9695	0.48427
∞	-	2.9698	0.48491

Table 4: The summary of different types of violation with two measurement settings and d outcomes. It has been shown that the maximum CGLMP violation $I_d^{\max}(\text{QM})$ does not correspond to maximally entangled input states. $I_d^{|\Phi_d^+\rangle}$ is the maximum violation for a maximally entangled input state $|\Phi_d^+\rangle$. $I_{22dd}^{|\Phi_d^+\rangle}$ is the corresponding best known I_{22dd} violation given in Equation (2.71) (Shun, 2015).

Criterion: Bell function – Buhrman and Massar inequality – Bell operator (2005)

We propose a generalized Bell inequality for two three-dimensional systems with three settings in each local measurement. It is shown that this inequality is maximally violated if local measurements are configured to be mutually unbiased and a composite state is maximally entangled. This feature is similar to Clauser-Horne-Shimony-Holt inequality for two qubits but is in contrast with the two types of inequalities, Collins-Gisin-Linden-Massar-Popescu and Son-Lee-Kim, for high-dimensional systems (Ji, Lee, Lim, Nagata, Lee, 2008).

Now we derive a three-setting Bell inequality for two qutrits. Alice and Bob now have three sets of measuring apparatus each, from which they each choose one and perform a measurement. The three variables whose values are determined by the measurements using Alice's (Bob's) three sets are referred to as A_0, A_1 and A_2 (B_0, B_1 and B_2), respectively. We assign three possible values of 1, ω , and ω^2 , where $\omega = e^{i2\pi/3}$ is a primitive third root of unity, to the outcome of the measurement on each variable. As discussed for the CHSH inequality, the local realistic description implies that the values of the variables are predetermined by the local hidden variables λ : $A_i = A_i(\lambda)$ and $B_j = B_j(\lambda)$, and a statistical average of their correlations is given as:

$$\langle A_i B_j \rangle = \int d\lambda \rho(\lambda) A_i(\lambda) B_j(\lambda)$$

where $\rho(\lambda)$ is the probability density distribution over λ :

$$\rho(\lambda) \geq 0 \text{ and } \int d\lambda \rho(\lambda) = 1.$$

We consider the following Bell function:

$$B(\lambda) = \frac{1}{2} \sum_{n=1}^2 \sum_{i=0}^2 \sum_{j=0}^2 \omega^{nij} A_i^n(\lambda) B_j^n(\lambda) \quad (2.72)$$

where A_i^n, B_j^n is the n-th power of A_i and B_j respectively and λ are the local hidden variables like the CHSH inequality (Ji, Lee, Lim, Nagata, Lee, 2008).

The constraint for the classical correlations using the above Bell function is:

$$-\frac{9}{2} \leq B(\lambda) \leq \frac{9}{2} \quad (2.73)$$

We examine the quantum violation of the three-setting Bell inequality for two qutrits. The Bell operator corresponding to the classical Bell function in Equation (2.72) is given as:

$$\hat{B} = \frac{1}{2} \sum_{n=1}^2 \sum_{i=0}^2 \sum_{j=0}^2 \omega^{nij} \hat{A}_i^n \otimes \hat{B}_j^n \quad (2.74)$$

where each operator \hat{A}_i (\hat{B}_j) represents a measurement for A_i (B_j) on Alice's (Bob's) qutrit. An orthogonal measurement of $M \in \{A_i, B_j\}$ is described by a complete set of orthonormal basis vectors $\{|k\rangle_M\}$. Distinguishing the measurement outcomes is indicated by a set of eigenvalues. Let the set of eigenvalues be $\{1, \omega, \omega^2\}$, as the trichotomic variable M takes an element in the set by definition. The

measurement operator is then represented by $\hat{M} = \sum_{k=0}^2 \omega^k |k\rangle_M \langle k|$. In this representation each trichotomic operator $\hat{M} \in \{\hat{A}_i, \hat{B}_j\}$ is unitary, satisfying $\hat{M}^3 = \mathbb{1}$, where $\mathbb{1}$ is the identity operator. We note the unitary operator \hat{M} and its second power \hat{M}^2 have the same measurement basis just with different orderings of eigenvalues so that the introduction of higher powers does not alter the number of measurement settings in this section.

We remark the previous work by Buhrman and Massar, in which the authors introduced a Bell function and determined its quantum upper bound allowed for the general case of d-dimensional systems and d measurements settings when local measurements on quantum entangled states are made. The quantum upper bound they determined is “non-tight” in the sense that their Bell function cannot take on a value greater than that, but it has not been proven that this upper bound can actually be attained. Applying their result to our Bell operator of Equation (2.74), the quantum upper bound is $3\sqrt{3} \approx 5.196$. On the other hand, it is proven that $3\sqrt{3} \cos(\frac{\pi}{18}) \approx 5.117$ is the maximum value actually attainable (Buhrman, Massar, 2005).

We want to generalize the Bell function for qudits. So we generalize the Bell inequality for qutrits to d-dimensional systems, namely qudits, with d a prime integer. A measurement on a qudit produces one of d possible outcomes. For a generalized Bell inequality for qudits, two observers are allowed each to choose one of d variables.

We consider a classical Bell function for qudits:

$$B(\lambda) = \frac{1}{d-1} \sum_{n=1}^{d-1} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} \omega^{nij} A_i^n(\lambda) B_j^n(\lambda) \quad (2.75)$$

where ω is now a primitive d-th root of unity, i.e. $\omega = \exp\left(\frac{i2\pi}{d}\right)$ and $A_i(\lambda) = \omega^{a_i(\lambda)}$ and $B_j(\lambda) = \omega^{b_j(\lambda)}$ with $a_i(\lambda)$ and $b_j(\lambda)$ integer-valued functions of hidden variables λ (Ji, Lee, Lim, Nagata, Lee, 2008).

The statistical average of the Bell function, namely $\langle B \rangle$, satisfies the following inequality:

$$-\frac{d^2}{d-1} \leq \langle B \rangle \leq \frac{d(2d-3)}{d-1} \quad (2.76)$$

where $\langle B \rangle = \int d\lambda \rho(\lambda) B(\lambda)$, with a probability density distribution $\rho(\lambda)$ and λ is a collection of local hidden variables.

The quantum Bell operator, corresponding to the classical Bell function, is given as:

$$\hat{B} = \frac{1}{d-1} \sum_{n=1}^{d-1} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} \omega^{nij} \hat{A}_i^n \otimes \hat{B}_j^n \quad (2.77)$$

where \hat{A}_i and \hat{B}_j are local unitary operators with eigenvalues $\{1, \omega, \omega^2, \dots, \omega^{d-1}\}$ (Ji, Lee, Lim, Nagata, Lee, 2008).

Our Bell inequalities show relatively small degrees of violations. Ratios of quantum to classical maxima are given for $d = 3, 5, 17$ as:

$$\frac{\langle \psi | \hat{B} | \psi \rangle}{\langle B \rangle} \approx \begin{cases} 1.137 & , \text{ for } d = 3 \\ 1.156 & , \text{ for } d = 5 \\ 1.229 & , \text{ for } d = 17 \end{cases}$$

These ratios are smaller than 1.414 and 1.436, those of CHSH inequality for qubits and CGLMP inequality for qutrits, respectively. However, it is interesting to observe that the ratios increase with respect to the dimension once the nonlocality appears (Ji, Lee, Lim, Nagata, Lee, 2008).

Criterion: Tight Bell Inequalities for many qubits (2006)

We present a family of tight Bell inequalities involving only two measurement settings of each party for $N > 2$ qubits. Remarkably our new inequalities are violated by some states which do satisfy all the standard Bell inequalities. Furthermore the inequalities automatically recover all the standard ones for systems with less than N qubits (Chen, Albeverio, Fei, 2006).

Their implementations are not only favorably within the reach of well-established technology of linear optics, but also can provide stronger nonlocality tests and contribute significantly to the reduction of experimental efforts (Chen, Albeverio, Fei, 2006).

We consider N parties and allow each of them to choose independently between two dichotomic observables A_j, A_j' for the j -th observer, specified by some

local parameters, each measurement having two possible outcomes -1 and 1 . We define:

$$B = B_{N-1} \otimes \frac{1}{2} (A_N + A'_N) + \mathbb{1}_{N-1} \otimes \frac{1}{2} (A_N - A'_N) \quad (2.78)$$

$$B_{N-1} =$$

$$=$$

$$\frac{1}{2^{N-1}} \sum_{s_1, \dots, s_{N-1} = \pm 1} S(s_1, \dots, s_{N-1}) \sum_{k_1, \dots, k_{N-1} = 1, 2} s_1^{k_1-1} \dots s_{N-1}^{k_{N-1}-1} E_{QM}(k_1, \dots, k_{N-1}) \quad (2.79)$$

where B_{N-1} is the quantum mechanical (QM) Bell operator of WWZB (Werner, Wolf, Zukowski and Brukner) inequalities for $N-1$ particles and $S(s_1, \dots, s_{N-1})$ can be arbitrary function taking only ± 1 as values. Here $E_{QM}(k_1, \dots, k_{N-1}) \equiv \langle \bigotimes_{j=1}^{N-1} O_j(k_j) \rangle$ denotes the expectation value of the correlation function $\bigotimes_{j=1}^{N-1} O_j(k_j)$, where $O_j(1) = A_j$ and $O_j(2) = A'_j$ with $k_j = 1, 2$. The notation $\mathbb{1}_{N-1}$ represents an identity matrix of dimension 2^{N-1} , with the meaning of “not measuring” the first $N-1$ parties (Werner, Wolf, Żukowski, Brukner, 2002).

The WWZB Bell operator is defined by:

$$B_N^{WWZB} = \frac{1}{2^N} \sum_{s_1, \dots, s_N = \pm 1} S(s_1, \dots, s_N) \sum_{k_1, \dots, k_N = 1, 2} s_1^{k_1-1} \dots s_N^{k_N-1} \bigotimes_{j=1}^N O_j(k_j) \quad (2.80)$$

where $S(s_1, \dots, s_N)$ is an arbitrary function of $s_i (= \pm 1)$, $i = 1, \dots, N$, taking values ± 1 , $O_j(1) = A_j$ and $O_j(2) = A'_j$ with $k_j = 1, 2$. It is shown that local realism requires: $|\langle B_N \rangle| \leq 1$ (Werner, Wolf, Żukowski, Brukner, 2002).

Noting that local realism requires $|\langle B_{N-1} \rangle_{LHV}| < 1$, we obtain:

$$|\langle B \rangle_{LHV}| = \frac{1}{2} |\langle B_{N-1} (A_N + A'_N) + (A_N - A'_N) \rangle_{LHV}| \leq 1 \quad (2.81)$$

In fact $A_N = \pm 1$ and $A'_N = \pm 1$ for the observer N , one has either $|A_N + A'_N| = 2$ and $|A_N - A'_N| = 0$, or vice versa. This implies that Equation (2.81) holds. For a given function of $S(s_1, \dots, s_{N-1})$, one can generate the full set of members of a family by simply permuting different locations, or the measurement orientations A_i and A'_i (Werner, Wolf, Żukowski, Brukner, 2002).

Maximal Violation:

We presented the first family of tight Bell inequalities with all the advantages above. It is shown that all the generalized GHZ entangled states, given by equation:

$$|\psi\rangle = \cos \alpha |0, \dots, 0\rangle + \sin \alpha |1, \dots, 1\rangle \quad (2.82)$$

where $0 < \alpha < \pi/4$, can violate the inequalities. All the Greenberger-Horne-Zeilinger states of N qubits (up to local unitary transformations) are shown to violate the inequalities maximally, by an amount that grows exponentially with N . Though the new inequalities involve only the same setup as the standard Bell inequalities (with only two measurement settings per site), they can reveal the nonlocality of the states (2.82) with an extraordinary power. Remarkably all the GHZ states of N qubits (up to local unitary transformations) are shown to violate the inequalities maximally, with a violation factor that grows exponentially as $2^{(N-2)/2}$ (Chen, Albeverio, Fei, 2006).

Theorem 2.83: All generalized GHZ states of many qubits violate a Bell inequality, where the generalized Greenberger-Horne-Zeilinger (GHZ) states given by Equation (2.82) (Chen, Albeverio, Fei, 2006).

Theorem 2.84: All the GHZ states violate the Bell inequality (Equation 2.81) maximally (Chen, Albeverio, Fei, 2006).

The result of Theorem (2.83) is remarkable. Our inequalities only involve the same setup as the standard nonlocality testing experiment by using only two measurement settings per site, and are immediately feasible due to rapidly developing technology for generation and manipulation of multiparticle entangled states in linear optical, atomic or trapped ions systems (Chen, Albeverio, Fei, 2006).

For N even, the corresponding Mermin inequalities presented above, are combinations of all the correlation functions and have a total of 2^N terms. These tight inequalities (Equation 2.78) require only $2^{N-1} + 2$ terms, as B_{N-1} is a combination of 2^{N-2} correlation functions in this case. Therefore these tight inequalities demand asymptotically only half of the experimental efforts, as compared with the standard ones, for N even (Chen, Albeverio, Fei, 2006).

It is shown that these tight inequalities reveal violation of local realism for some class of states, while the standard Bell's inequalities fail to detect. Furthermore the inequalities can be maximally violated by GHZ states with an amount that grows exponentially as $2^{(N-2)/2}$ and all the standard inequalities for less than N parties can be automatically recovered (Chen, Albeverio, Fei, 2006).

Maximal Violation for two qudits:

We investigate the maximal violation of Bell inequalities for two d -dimensional systems by using the method of Bell operator. The maximal violation corresponds to the maximal eigenvalue of the Bell operator matrix (Chen, Wu, Kwek, Oh, Ge, 2006).

The eigenvectors corresponding to these eigenvalues are described by asymmetric entangled states. We estimate the maximum value of the eigenvalue for large dimension. A family of elegant entangled states $|\psi\rangle_{app}$ that violate Bell inequality more strongly than the maximally entangled state but are somewhat close to these eigenvectors is presented (Chen, Wu, Kwek, Oh, Ge, 2006).

These approximate states can potentially be useful for quantum cryptography as well as many other important fields of quantum information (Chen, Wu, Kwek, Oh, Ge, 2006).

d	2	3	4	5	6	7	8	9	10	20	30	40	50	60
$I_d(\Psi\rangle_{\text{eig}})$	2.82843	2.9149	2.9727	3.0157	3.0497	3.0777	3.1013	3.1217	3.1396	3.2492	3.3068	3.3449	3.3728	3.3946
$I_d(\Psi\rangle_{\text{app}})$	2.82843	2.90909	2.96466	3.00689	3.04078	3.06895	3.09296	3.10639	3.13219	3.24551	3.30496	3.34389	3.37222	3.39417
$I_d(\Psi\rangle_{\text{mes}})$	2.82843	2.87293	2.89624	2.91054	2.92020	2.92716	2.93241	2.93651	2.93980	2.95472	2.95974	2.96225	2.96376	2.96477
d	70	80	90	100	150	200	250	300	350	400	450	500	550	600
$I_d(\Psi\rangle_{\text{eig}})$	3.4124	3.4273	3.4400	3.4511	3.4914	3.5178	3.5370	3.552	3.5641	3.5743	3.583	3.5906	3.5973	3.6033
$I_d(\Psi\rangle_{\text{app}})$	3.41192	3.4267	3.4393	3.45022	3.48933	3.51446	3.53256	3.54651	3.55775	3.5671	3.57505	3.58194	3.588	3.59339
$I_d(\Psi\rangle_{\text{mes}})$	2.96549	2.96603	2.96645	2.96678	2.96779	2.96830	2.96860	2.96880	2.96895	2.96906	2.96914	2.96921	2.96926	2.96931
d	650	700	750	800	850	900	950	1000	1500	2000	2500	3000	3500	4000
$I_d(\Psi\rangle_{\text{eig}})$	3.6087	3.6136	3.6181	3.6222	3.6260	3.6296	3.6329	3.6360	3.6594	3.6747	3.6859	3.6946	3.7017	3.7077
$I_d(\Psi\rangle_{\text{app}})$	3.59824	3.60263	3.60664	3.61032	3.61372	3.61687	3.61981	3.62256	3.64299	3.65623	3.66584	3.67330	3.67936	3.68442
$I_d(\Psi\rangle_{\text{mes}})$	2.96935	2.96938	2.96941	2.96944	2.96946	2.96948	2.96950	2.96951	2.96961	2.96966	2.96969	2.96971	2.96973	2.96974
d	5000	6000	7000	8000	50000	70000	80000	90000	100000	200000	300000	400000	500000	600000
$I_d(\Psi\rangle_{\text{eig}})$	3.7174	3.7250	3.7311	3.7362										
$I_d(\Psi\rangle_{\text{app}})$	3.69253	3.69884	3.70398	3.70829	3.75659	3.76372	3.76644	3.76878	3.77083	3.78345	3.79019	3.79472	3.79810	3.80080
$I_d(\Psi\rangle_{\text{mes}})$	2.96975	2.96976	2.96977	2.96978	2.96981	2.96981	2.96981	2.96981	2.96981	2.96981	2.96981	2.96981	2.96981	2.96981

Table 5: Bell expressions, where I_d = Tsirelson's bound , $|\Psi\rangle_{\text{eig}}$ maximal violation = maximal eigenvalue of Bell operator matrix , $|\Psi\rangle_{\text{app}}$ = approximate states , $|\Psi\rangle_{\text{mes}}$ = maximally entangled state (Chen, Wu, Kwek, Oh, Ge, 2006) .

Criterion: Experimentally testable state-independent quantum contextuality (2008)

Local hidden variable theories are a special type of non-contextual hidden variable (NCHV) theories, defined as those where the expectation value of an observable A is the same whether A is measured with a compatible observable B, or with a compatible observable C, even though B and C are incompatible. The Kochen-Specker (KS) theorem states that no NCHV theory can reproduce QM (Cabello, 2008).

These proofs apply to systems described by Hilbert spaces of dimension $d > 3$ and are state-independent (i.e., valid for any state). Quantum contextuality is related to quantum error correction, random access codes, quantum key distribution, one-location quantum games, and entanglement detection between internal degrees of freedom. There are “KS inequalities”, which are based on the assumption of contextuality and on some QM predictions, and therefore are not independent of QM (Cabello, 2008).

A natural question is the following: Given a physical system described in QM by a Hilbert space of dimension d , is it possible to derive experimentally testable

inequalities using only the assumption of noncontextuality, such that any quantum state violates them (Cabello, 2008)?

Given a physical system described in Quantum Mechanics by a Hilbert space of dimension d , we suppose that A_{ij} is an observable with two possible results: -1 or $+1$, and two observables A_{ij} and A_{kl} are compatible if they share a subindex (i.e. $i = k$, or $i = l$, or $j = k$, or $j = l$). When we prepare an ensemble of systems and measure 4 compatible observables A_{ij}, A_{ik}, A_{il} and A_{im} in each system, $\langle A_{ij}A_{ik}A_{il}A_{im} \rangle$ denotes the average of the products of their results. In any theory of NCHV (Non-Contextual Hidden Variable) in which the observables A_{ij} have definite results, the following inequality must be satisfied:

$$\begin{aligned} & -\langle A_{12}A_{16}A_{17}A_{18} \rangle - \langle A_{12}A_{23}A_{28}A_{29} \rangle - \langle A_{23}A_{34}A_{37}A_{39} \rangle - \\ & -\langle A_{34}A_{45}A_{47}A_{48} \rangle - \langle A_{45}A_{56}A_{58}A_{59} \rangle - \langle A_{16}A_{56}A_{67}A_{69} \rangle - \\ & -\langle A_{17}A_{37}A_{47}A_{67} \rangle - \langle A_{18}A_{28}A_{48}A_{58} \rangle - \langle A_{29}A_{39}A_{59}A_{69} \rangle \leq 7 \end{aligned} \quad (2.85)$$

This inequality can be proven as follows. We define $\alpha = -A_{12}A_{16}A_{17}A_{18} - \dots - A_{29}A_{39}A_{59}A_{69}$. If we generate all the 2^{18} possible values of α , we will find that $\alpha = 7$ is the maximum. Therefore, if we can measure α on different systems, the average satisfies $\langle \alpha \rangle \leq 7$. We cannot measure α on a single system, because α contains incompatible observables. However, since we assume that each A_{ij} would give the same result in any context, we can measure subsets of compatible observables on different subensembles prepared in the same state and then inequality (2.85) is valid for the averages over each subensemble. This derivation is similar to a standard derivation of a Bell inequality. The only difference is that in a Bell inequality we assume that the result of a measurement of A_{12} is independent of spacelike separated measurements, while here we assume that it is independent of compatible measurements.

Now consider a physical system described by a Hilbert space of dimension $d = 4$ (e.g., two qubits or a single spin-3/2 particle), and the observables represented by the operators:

$$A_{ij} = 2|v_{ij}\rangle\langle v_{ij}| - \mathbb{1}$$

where v_{ij} is a unit vector and $\mathbb{1}$ denotes the identity. Each observable A_{ij} has two possible results: -1 or +1. If v_{ij} is orthogonal to v_{ik} , then A_{ij} and A_{ik} are compatible.

Therefore, 4 orthogonal vectors define 4 compatible observables. 18 vectors v_{ij} with the orthogonality relations assumed in inequality (2.85) are presented in Figure 8 below.

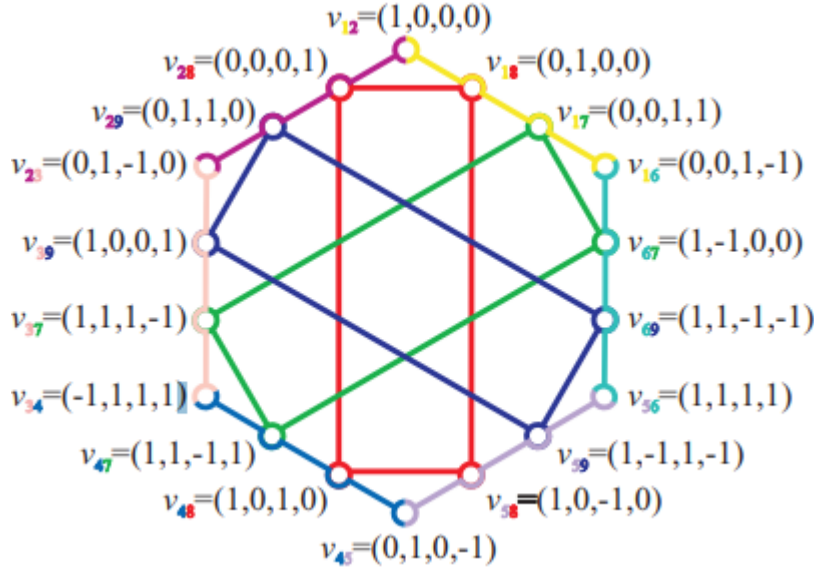


Figure 8: Each dot represents a unit-vector v_{ij} . Each of the 6 sides of the regular hexagon and each of the 3 rectangles contains only orthogonal vectors. Note that, for clarity's sake, most labels have no unit length (Cabello, 2008).

Now we prove that for $d=4$ Quantum Mechanics violates (2.85) for any state.

According to QM, if one measures on the same system 4 compatible observables A_{ij} corresponding to 4 orthogonal vectors v_{ij} , the product of their 4 results will always be -1, because $A_{ij}A_{ik}A_{il}A_{im} = -\mathbb{1}$. Therefore, using the vectors of Figure 8, QM predicts that the experimental value of the left-hand side of inequality (2.85) must be 9 in any state, which is clearly beyond the bound for any description based on noncontextual hidden variables (Cabello, 2008).

Suppose that P_{ij} with $i \in \{1, 2, 3\}$ and $j \in \{4, 5, 6\}$ is an observable with two possible results: -1 or +1, and two observables P_{ij} and P_{kl} are compatible if they share a subindex. Using the method described before, it can be easily proved that

any NCHV theory in which the observables P_{ij} have definite results satisfies the following inequality:

$$\begin{aligned} & \langle P_{14}P_{15}P_{16} \rangle + \langle P_{24}P_{25}P_{26} \rangle + \langle P_{34}P_{35}P_{36} \rangle + \langle P_{14}P_{24}P_{34} \rangle + \\ & + \langle P_{15}P_{25}P_{35} \rangle - \langle P_{16}P_{26}P_{36} \rangle \leq 4 \end{aligned} \quad (2.86)$$

However, if we consider a two-qubit system and choose the following observables:

$$\begin{aligned} P_{14} &= Z_1, \quad P_{15} = Z_2, \quad P_{16} = Z_1 \otimes Z_2, \\ P_{24} &= X_2, \quad P_{25} = X_1, \quad P_{26} = X_1 \otimes X_2, \\ P_{34} &= Z_1 \otimes X_2, \quad P_{35} = X_1 \otimes Z_2, \quad P_{36} = Y_1 \otimes Y_2 \end{aligned} \quad (2.87)$$

where e.g. Z_1 denotes $\sigma_Z^{(1)}$, the Pauli matrix Z of qubit 1, then, according to QM, the left-hand side of Equation (2.86) must be 6, since $P_{14}P_{15}P_{16} = P_{24}P_{25}P_{26} = P_{34}P_{35}P_{36} = P_{14}P_{24}P_{34} = P_{15}P_{25}P_{35} = -P_{16}P_{26}P_{36} = \mathbb{1}$, where $\mathbb{1}$ denotes the identity. Therefore, QM violates inequality (2.86) for any two-qubit state (Cabello, 2008).

Suppose that the $4+2n$ observables $A_1, \dots, A_4, B_1, \dots, B_n, C_1, \dots, C_n$ with n (odd) ≥ 3 , have only two possible results: -1 or +1. Assuming that each of the following averages contains only compatible observables, using the method described before, it can be easily seen that any NCHV theory satisfies the following inequality:

$$\begin{aligned} & \langle A_1B_1B_2 \prod_{i=3}^n B_i \rangle + \langle A_2B_1C_2 \prod_{i=3}^n C_i \rangle + \langle A_3C_1B_2 \prod_{i=3}^n C_i \rangle + \langle A_4C_1C_2 \prod_{i=3}^n B_i \rangle - \\ & - \langle A_1A_2A_3A_4 \rangle \leq 3 \end{aligned} \quad (2.88)$$

However, if we consider an n -qubit system, with n (odd) ≥ 3 , and choose the following observables:

$$A_1 = Z_1 \otimes Z_2 \otimes Z_3 \otimes \dots \otimes Z_n$$

$$A_2 = Z_1 \otimes X_2 \otimes X_3 \otimes \dots \otimes X_n$$

$$A_3 = X_1 \otimes Z_2 \otimes X_3 \otimes \dots \otimes X_n$$

$$A_4 = X_1 \otimes X_2 \otimes Z_3 \otimes \dots \otimes Z_n$$

$$B_i = Z_i$$

$$C_i = X_i \quad (2.89)$$

then according to QM, the left-hand side of inequality (2.88) must be 5, since $A_1 B_1 B_2 \prod_{i=3}^n B_i = A_2 B_1 C_2 \prod_{i=3}^n C_i = A_3 C_1 B_2 \prod_{i=3}^n C_i = A_4 C_1 C_2 \prod_{i=3}^n B_i = -A_1 A_2 A_3 A_4 = 1$. Therefore, QM violates inequality (2.88) for any n-qubit state with $n \text{ (odd)} \geq 3$ (Cabello, 2008).

The above 3 experimentally testable inequalities are valid for any NCHV theory and violated by any quantum state. They combine the most celebrated properties of the Bell inequalities, independence of QM and experimental testability, with state independence, the most celebrated property of the KS theorem. One of these inequalities seems particularly suitable to experimentally test the state-independent violation predicted by QM (Cabello, 2008).

Criterion: Zohren and Gill inequality for 2x2xd Bell scenario (2008)

Entanglement and nonlocality in general, there have been many studies of generalized Bell inequalities. These include the Mermin inequality for multiple qubits ($n > 2$), the Collins-Gisin inequality for multiple measurements ($m > 2$), and the Collins-Gisin-Linden-Massar-Popescu (CGLMP) inequality for higher-dimensional systems known as qudits ($d > 2$). For $n = 2$ and $m = d > 2$ exhibits the potential to reduce the requirements to close the detection loophole. Here, n denotes the number of parties (e.g. Alice, Bob). Each party performs one of m measurement choices, with each measurement registers one of d outcomes.

The CGLMP inequality (for $n = m = 2$ and $d > 2$), which takes the form:

$$I_d = \sum_{k=0}^{\lfloor \frac{d}{2} \rfloor - 1} \left(1 - \frac{2k}{d-1} \right) [P(k) - P(-k-1)] \leq 2 \quad (2.90)$$

where $P(k) = P(A_1 = B_1 + k) + P(B_1 = A_2 + k + 1) + P(A_2 = B_2 + k) + P(B_2 = A_1 + k)$ (2.91) and the measurement settings are labeled by 1 and 2 for Alice's (or Bob's) choice of measurement, with outcomes A_1 and A_2 for Alice (and B_1 and B_2 for Bob) and in a slight abuse of notation, the joint probability $p(A_0 = B_0)$ indicates the probability that Alice and Bob's measurement outcomes are identical.

Here the joint probabilities are defined for outcomes $A_a = 0, 1, \dots, d-1$ and the addition is performed modulo d . These can expressed as:

$$P(A_a = B_b + k) = \sum_{j=0}^{d-1} P(A_a = j, B_b = j + k \mod d) \quad (2.92)$$

It is also studied a closely related inequality proposed by Zohren and Gill:

$$P_L(A_2 < B_2) + P_L(B_2 < A_1) + P_L(A_1 < B_1) + P_L(B_1 \leq A_2) \geq 1 \quad (2.93)$$

where $P_L(A_a < B_b) = \sum_{i < j} P_L(i, j | a, b)$ and $P_L(i, j | a, b) = \sum_{\lambda} p(\lambda) P(i | a, \lambda) P(j | b, \lambda)$ (local realistic theory) (Zohren, Gill, 2008).

These two inequalities (2.92, 2.93) have the remarkable property that the violation increases with increasing d (Polozova, Strauch, 2016).

Criterion: Bell ratio – Bell operator (2016)

We reproduce some known results in a novel way and find some multipartite Bell inequalities for systems having three settings and three outcomes per party. We construct Bell inequalities for systems composed of several subsystems composed by more than two levels each. In particular, we focus our attention on quantum systems consisting on qutrits. Inequalities for three outcomes have been written more often in terms of probabilities but they can also be treated with expectation values.

We extend this formalism in order to build new inequalities for three outcomes and a different number of parties and find its classical and quantum bounds for qutrits in a semi-systematic way. We find some regular patterns for the coefficients of the inequalities and for the settings and states that maximally violate these inequalities. This mechanism is potentially generalizable to other dimensions (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Bell Inequalities for two Outcomes and two parties:

In the case of two parties the only relevant Bell inequality is the one of Clauser, Horne, Shimony and Holt. It is obtained out of the following Bell polynomial:

$$BCHSH = ab + ab' + a'b - a'b' \quad (2.94)$$

where $a, a' = \pm 1$ and $b, b' = \pm 1$ are the possible outcomes detected by observers Alice and Bob, respectively. Note that equation (2.94) can be factorized as:

$$BCHSH = a(b + b') + a'(b - b') \quad (2.95)$$

so one of the terms is ± 2 , while the other one is equal to zero, which means that the maximum value that can be obtained with a local realistic theory is $\langle B_{CHSH} \rangle_{LR} = 2$. In a more general case, this classical bound can be obtained by computing the value of the Bell polynomial with all possible outcomes for a, a', b and b' and selecting its maximum (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

In quantum mechanics, the variables a, a' and b, b' are represented by Hermitian operators acting on the Hilbert spaces \mathcal{H}_a and \mathcal{H}_b , respectively. For dichotomic variables the operators satisfy $a^2 = a'^2 = b^2 = b'^2 = \mathbb{1}$, because the measurement operators a, a', b and b' have eigenvalues ± 1 . The quantum Bell operator reads then:

$$BCHSH = a \otimes b + a \otimes b' + a' \otimes b - a' \otimes b' \quad (2.96)$$

where \otimes denotes the Kronecker product. The quantum bound $\langle B_{CHSH} \rangle_{QM}$ corresponds to the maximal eigenvalue of all possible Bell operators (2.96) satisfying the previously stated conditions. A Bell operator B defines a Bell inequality if $\langle B \rangle_{LR} < \langle B \rangle_{QM}$. In the case of CHSH inequality, it is proven by Tsirelson that the maximum quantum value is $\langle B_{CHSH} \rangle_{QM} = 2\sqrt{2}$ (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

We study the ratio associated to a Bell polynomial:

$$R(B) = \frac{\langle B \rangle_{QM}}{\langle B \rangle_{LR}} \quad (2.97)$$

as it quantifies the strength of the inequality generated by the Bell operator B . Note that a Bell inequality is characterized by the ratio $R(B) > 1$. For example, for the CHSH inequality we have $R(B_{CHSH}) = \sqrt{2}$ (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Quantum states producing $R(B) > 1$ are non-local in the sense that those ratios cannot be reproduced by considering a local hidden variable theory. As consequence,

non-local quantum states cannot be fully separable. However, entanglement and non-locality are different concepts. Indeed, some entangled states do not violate any Bell inequality. Furthermore, states producing the maximal ratio are typically highly entangled (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Three parties:

In the case of three qubits (Alice, Bob, Charlie) the most general symmetric Bell operator can be written as:

$$B_3 = z_0(a \otimes b \otimes c) + z_3(a' \otimes b' \otimes c') + z_1(a \otimes b \otimes c' + a \otimes b' \otimes c + a' \otimes b \otimes c) + z_2(a \otimes b' \otimes c' + a' \otimes b \otimes c' + a' \otimes b' \otimes c) \quad (2.98)$$

where $z_0, \dots, z_3 \in \mathbb{R}$. The following values for z_i :

$$z_i^M = \{z_0, z_1, z_2, z_3\}^M = \{0, 1, 0, -1\} \quad (2.99)$$

lead us to the 3-qubit Mermin operator:

$$M_3 = (a \otimes b \otimes c' + a \otimes b' \otimes c + a' \otimes b \otimes c) - (a' \otimes b' \otimes c') \quad (2.100)$$

having a square:

$$M_3^2 = 4\mathbb{1}_{ABC} - ([a, a'] \otimes [b, b'] \otimes \mathbb{1}_C + [a, a'] \otimes \mathbb{1}_B \otimes [c, c'] + \mathbb{1}_A \otimes [b, b'] \otimes [c, c']) \quad (2.101)$$

Bell inequalities generated by operators like M_3 are called multipartite Bell inequalities (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Criterion: CGLMP inequality – Bell operator C_{nsd} (2016)

Two parties with hermitian operators and 3 Outcomes:

A Bell inequality for two parties, two settings and d outcomes is known as CGLMP inequality. In the case of three outcomes, as we have seen above, the inequality is given by:

$$p(a = b) + p(b = a' + 1) + p(a' = b) + p(b' = a) - p(a = b - 1) - p(b = a') - p(a' = b' - 1) - p(b' = a - 1) \leq 2 \quad (2.102)$$

where the possible outcomes are $\{0, 1, 2\}$ and the sum inside probabilities is modulo $d = 3$. This Bell inequality can be associated with the following Bell operator:

$$C_{223} = 2 - 3(a^2 + b'^2) + \frac{3}{4} (ab + a^2b - a'b - a'^2b - ab^2 + a'b^2 + ab' - a^2b' + a'b' + a'^2b' + ab'^2 - a'b'^2) + \frac{9}{4} (a^2b^2 - a'^2b^2 + a^2b'^2 + a'^2b'^2) \quad (2.103)$$

where the notation C_{nsd} stands for n parties, s settings and d outcomes (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

The quantum value is given by $\langle C_{223} \rangle_{QM} = \frac{2(5-\gamma^2)}{3} \approx 2.9149$ for the optimal state:

$$|\psi\rangle = \frac{(|00\rangle + \gamma|11\rangle + |22\rangle)}{\sqrt{2+\gamma^2}}$$

where $\gamma = \frac{\sqrt{11}-\sqrt{3}}{2} \approx 0.7923$. The violation rate for this quasi Bell state reads:

$$R(C_{223}) = \frac{5-\gamma^2}{3} \approx 1.4547 \quad (\text{Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016}).$$

Larger number of parties:

In the case of four parties, two settings and three outcomes we have found the following symmetric Bell operator:

$$C_{423} = [2(abcd) + (a'bcd + ab'cd + abc'd + abcd') + w(a'b'cd + a'bc'd + a'bcd' + ab'c'd + ab'cd' + abc'd') + (a'b'c'd + a'bc'd' + a'b'cd' + ab'c'd') + 2(a'b'c'd')] \quad (2.104)$$

which produces $\langle C_{423} \rangle_{LR} = 3\sqrt{3} \approx 5.19$, $\langle C_{423} \rangle_{QM} \approx 9.766$ and $R(C_{423}) \approx 1.879$ for the optimal settings:

$$A = B = C = D = X$$

$$A' = B' = C' = D' = Z \quad (2.105)$$

which are again mutually unbiased settings. The optimal state has entanglement properties equivalent to those of the GHZ of four parties and three settings:

$$|GHZ_{4,3}\rangle = (|0000\rangle + |1111\rangle + |2222\rangle)/\sqrt{3} .$$

It is noted X and Z are the generators of the Weyl-Heisenberg group which is formed by the generalized unitary Pauli matrices:

$$X = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} , Z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & w & 0 \\ 0 & 0 & w^2 \end{pmatrix} \quad (2.106)$$

where $w = e^{2\pi i/3}$ (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

For 6 parties we have also found a symmetric Bell operator. To simplify the notation, the polynomials having terms with the same number of primes are denoted by its number of primes in parenthesis, for example:

$$(1') \equiv a'bcdef + ab'cdef + abc'def + abcd'ef + abcde'f + abcdef' .$$

In this notation, the 6 parties operator reads:

$$C_{623} = -w(0') + (1') - (2') + w(3') - (4') + (5') - w(6') \quad (2.107)$$

For this inequality, $\langle C_{623} \rangle_{LR} = 9\sqrt{3} \approx 15.589$, $\langle C_{623} \rangle_{QM} \approx 32.817$ and

$R(C_{623}) \approx 2.105$, with optimal settings. The maximal violation is given by a quasi GHZ state, as for the case of 2 and 3 qutrits (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Qutrits	2	3	4	5	6
$\langle [B]_A \rangle_{LR}$	$\sqrt{3}$	$3\sqrt{3}$	$3\sqrt{3}$	$9\sqrt{3}$	$9\sqrt{3}$
$\langle [B]_A \rangle_{LR}^{(-)}$	$-2\sqrt{3}$	$-3\sqrt{3}$	$-6\sqrt{3}$	$-9\sqrt{3}$	$-18\sqrt{3}$
$\langle [B]_H \rangle_{LR}$	3	3	9	9	27
$\langle [B]_H \rangle_{LR}^{(-)}$	-3	-6	-9	-18	-27
$\langle [B]_x \rangle_{QM}$	2.524	5.058	9.766	15.575	32.817
R	1.457	1.686	1.879	1.731	2.105
Settings	MOS	MUB	MUB	Num.	MOS
P	0.347	0.342	1/3	0.351	0.334

Table 7: Main results for inequalities from 2 to 6 qutrits, where it can be seen that the classical patterns match perfectly, while the 5-qutrit inequality appears not to follow the

quantum pattern. Here, $\langle B \rangle_{LR}$ and $\langle B \rangle_{LR}^{(-)}$ denote the maximum and minimum classical value for the optimizations of anti-Hermitian or Hermitian part of the operator, respectively. The quantity that we take as the extremal classical bound is marked in bold, and $\langle [B]_x \rangle_{QM}$ stands for its corresponding quantum value, where $x = A$ for an even number of qutrits and $x = H$ for an odd number of qutrits. $R = \langle B \rangle_{QM} / \langle B \rangle_{LR}$ and Settings denotes the optimal settings. P denotes the purity of the $[n/2]$ party reductions of the optimal state and Num. means numerical approximate solution and italic font in the 5-qutrits case is written to note that this case does not follow the same patterns of the others. We remark that optimal values appearing in this table have been achieved by optimizing over qutrit systems (Alsina, Cervera, Goyeneche, Latorre, Zyczkowski, 2016).

Criterion: 3-qubits set Bell inequalities (2017)

We consider a three-qubit system, with a qubit each with Alice, Bob and Charlie. In the Bell inequalities that we introduce, two of the parties will make two measurements, while the third party will make only one measurement. This third party can be either Alice, Bob, or Charlie. A general state need not have any symmetry, therefore we will be considering a set of Bell inequalities, rather than one inequality. The one measurement by one of the parties is necessary. We note that one of the two parties makes only one measurement. We list the set of six inequalities. In this list, the left-hand side should be thought of as the expectation value of the observables. In the first and third inequalities, Alice makes one measurement given by observable A_1 , Bob measures the observables B_1 and B_2 , and Charlie measures observables C_1 and C_2 . These are dichotomic observables, with values $\{-1, 1\}$. In the inequalities (2.108b) and (2.108f), Bob measures only one observable, B_1 , while in the inequalities (2.108d) and (2.108e), Charlie measures only one observable, C_1 . Other parties measure two observables:

$$A_1 B_1 (C_1 + C_2) + B_2 (C_1 - C_2) \leq 2 \quad (2.108a)$$

$$A_1 B_1 (C_1 + C_2) + A_2 (C_1 - C_2) \leq 2 \quad (2.108b)$$

$$(B_1 + B_2) C_1 + A_1 (B_1 - B_2) C_2 \leq 2 \quad (2.108c)$$

$$A_1 (B_1 + B_2) + A_2 (B_1 - B_2) C_1 \leq 2 \quad (2.108d)$$

$$(A_1 + A_2)B_1 + (A_1 - A_2)B_2C_1 \leq 2 \quad (2.108e)$$

$$(A_1 + A_2)C_1 + (A_1 - A_2)B_1C_2 \leq 2 \quad (2.108f)$$

We obtain the bound for the first inequality (2.108a) and the analysis is similar for the others. Let us call the corresponding Bell operator for the first inequality (2.108a) as:

$$B_3 = A_1B_1(C_1 + C_2) + B_2(C_1 - C_2) \quad (2.109)$$

If we take the square of this expression we get:

$$B_3^2 = 4\mathbb{1} + A_1[C_1, C_2][B_1, B_2] \quad (2.110)$$

Here, we have used $A_1^2 = B_1^2 = B_2^2 = C_1^2 = C_2^2 = \mathbb{1}$. We know that, for two bounded operators X and Y :

$$\|[X, Y]\| \leq 2 \|X\| \|Y\| \quad (2.111)$$

where “ $\|$ ” is the sup norm of a bounded operator. Using this relation, we notice that the maximum value will be obtained when B_3^2 is $8\mathbb{1}$ and hence $\|B_3\| \leq 2\sqrt{2}$ (Das, Datta, Agrawal, 2017).

Proposition 1: All generalized GHZ states violate all six inequalities (2.108a – 2.108f) of this set (Das, Datta, Agrawal, 2017).

Proposition 2: Any separable pure three-qubit state obeys all the inequalities within the set (Das, Datta, Agrawal, 2017).

Proposition 3: All biseparable pure three-qubit states violate exactly two inequalities within the set and the amount of maximal violation are same for both (Das, Datta, Agrawal, 2017).

Proposition 4: For all genuine tripartite entangled states, we have violation within the set (Das, Datta, Agrawal, 2017).

It is shown that the more entangled a generalized GHZ state is, the more will be the violation. This establishes a relation between nonlocality and entanglement for this class of states (Das, Datta, Agrawal, 2017).

We have established that our set of inequalities are violated by any entangled three-qubit state. We can generalize this set of inequalities to n -qubit states. We have to distinguish between two cases, odd number of qubits and even number of qubits. Starting from the operator, of which GHZ state is an eigenstate, one can construct different Bell inequalities. For even n , there will be a set of n inequalities, while for odd n , the number will rise to $n(n-1)$. The set is larger for odd number of qubits, because we have choice of making one measurement on any of n qubits, while in the case of even n , two measurements are made on all qubits. Therefore, we have to construct different types of inequalities for even and odd number of particles. We have already seen that the GHZ state of three qubits is eigenstate of the operator: $\sqrt{2} (\sigma_x \otimes \sigma_x \otimes \sigma_x + \sigma_z \otimes \sigma_z \otimes \mathbb{1})$ with the highest eigenvalue $2\sqrt{2}$.

This form of the operator can be generalized for any n -qubit GHZ state, when n is odd. It is noted that n -qubit GHZ states is the eigenstate of the operator:

$$\sqrt{2} (\sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \dots \otimes \sigma_x^{n \text{th}} + \sigma_z \otimes \sigma_z \otimes \dots \otimes \sigma_z^{(n-1) \text{th}} \otimes \mathbb{1}) \quad (2.112)$$

with the highest eigenvalue $2\sqrt{2}$. So, like the three-qubit case, we have to consider non-correlation Bell inequalities, when n is odd. The first two Bell inequalities (2.108a) and (2.108b) can be easily generalized for n -qubit pure states as:

$$A_1 A_2 A_3 A_4 A_5 \dots (A_n + A'_n) + A'_2 A'_3 A'_4 A'_5 \dots (A_n - A'_n) \leq 2 \quad (2.113)$$

and

$$A_2 A_3 A_4 A_5 \dots (A_n + A'_n) + A_1 A'_2 A'_3 A'_4 A'_5 \dots (A_n - A'_n) \leq 2 \quad (2.114)$$

Here, A_i and A'_i are two dichotomic observable for i -th party. In these inequalities, one measurement has been made on first qubit. Similarly, one can make single measurement on $(n-2)$ other qubits. This will lead to $(n-1)$ inequalities. We can write n such $(n-1)$ inequalities with $(A_i \pm A'_i)$ for i -th qubit, giving a set of total $n(n-1)$ inequalities. For three qubits the number of inequalities in the set is twelve (Das, Datta, Agrawal, 2017).

For finding maximal violation, we consider all allowed A_i and A'_i therefore their positions in the inequalities can be interchanged. The above set of inequalities

can be used to characterize the entanglement of n -qubit states for odd n . In the case of generalized n -qubit GHZ states, any one of these generalized inequalities is enough. One can show that for odd number of qubits these non-correlation Bell inequalities are violated by all generalized GHZ states with maximum violation of $2\sqrt{2}$ for the conventional GHZ state. Situation changes when one considers GHZ like states with even number of qubits. Because now, like the Bell states, the conventional GHZ state of n qubits (n is even) is the eigenstate of the operator:

$\sqrt{2} (\sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \dots \otimes \sigma_x^{n \text{th}} + \sigma_z \otimes \sigma_z \otimes \dots \otimes \sigma_z^{(n-1) \text{th}} \otimes \sigma_z)$ with highest eigenvalue $2\sqrt{2}$. This suggests that correlation Bell inequalities are required in this case. For example, we can generalize the first correlation Bell inequality as:

$$(A_1 + A'_1)A_2A_3A_4A_5 \dots A_n + (A_1 - A'_1)A'_2A'_3A'_4A'_5 \dots A'_n \leq 2 \quad (2.115)$$

Proposition 5: Multiqubit extension of the inequalities are violated by multiqubit generalized GHZ states (Das, Datta, Agrawal, 2017).

Criterion: Coefficient matrix for Bell inequalities (2018)

Multi-setting Tight Bell Inequality for 2 Qubits:

We focus on Bell inequality for two-particle systems. The Bell-type scenario involves only two observers and each of them measures M different local observables of two outcomes ± 1 . For simplicity and convenience, we denote $X_{1,1_k}, X_{2,2_k}$ as A_k and B_k ($k = 1, \dots, M$) respectively. The correlation function $Q(A_i B_j)$, in the case of a local realistic theory, is then the average values of the products $A_i B_j$ over many runs of the experiment. We also denote $Q(A_i B_j)$, $Q(A_i)$ and $Q(B_j)$ as Q_{ij} , Q_{i0} and Q_{0j} , respectively. Then the famous CHSH inequality, as we have seen above from the inequality (2.59) :

$$I_{CHSH} = Q_{11} + Q_{12} + Q_{21} - Q_{22} < 2$$

and in the form of joint probability, the inequality (2.60) reads:

$$I_{CG} = P(a_1 = 0, b_1 = 0) + P(a_1 = 0, b_2 = 0) + P(a_1 = 0, b_3 = 0) + \\ + P(a_2 = 0, b_1 = 0) + P(a_3 = 0, b_1 = 0) - P(b_2 = 0) +$$

$$P(a_2 = 0, b_2 = 0) - P(a_2 = 0, b_3 = 0) - P(a_3 = 0, b_2 = 0) - P(a_1 = 0) - 2P(b_1 = 0) \leq 0$$

The four-setting Bell inequality (2.62):

$$-6 \leq I_4 = Q_{11} + Q_{22} + Q_{12} + Q_{21} + Q_{14} + Q_{41} - Q_{24} - Q_{42} - 2Q_{33} + Q_{31} + Q_{13} + Q_{32} + Q_{23} \leq 6$$

The last inequality (2.62) can be written in the following way:

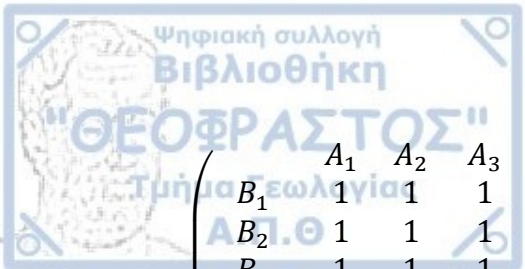
$$I_4 = \begin{pmatrix} & A_1 & A_2 & A_3 & A_4 \\ B_1 & 1 & 1 & 1 & 1 \\ B_2 & 1 & 1 & 1 & -1 \\ B_3 & 1 & 1 & -2 & 0 \\ B_4 & 1 & -1 & 0 & 0 \end{pmatrix} \quad (2.116)$$

Here, the coefficient in the matrix indicate the coefficients of the corresponding expectation values. Using the same method, we find many six-setting Bell inequalities for two qubits. We present only one of them whose correlation coefficients are regular with respect to the CHSH inequality and inequality (2.62). In the matrix form, it reads:

$$I_6 = \begin{pmatrix} & A_1 & A_2 & A_3 & A_4 & A_5 & A_6 \\ B_1 & 1 & 1 & 1 & 1 & 1 & 1 \\ B_2 & 1 & 1 & 1 & 1 & 1 & -1 \\ B_3 & 1 & 1 & 1 & 1 & -2 & 0 \\ B_4 & 1 & 1 & 1 & -3 & 0 & 0 \\ B_5 & 1 & 1 & -2 & 0 & 0 & 0 \\ B_6 & 1 & -1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (2.117)$$

Inspired by previous inequalities, it is not difficult to guess the general form of a set of even setting Bell inequalities:

$$I_{2n} =$$



$$\begin{pmatrix}
 & A_1 & A_2 & A_3 & \dots & A_n & A_{n+1} & A_{n+2} & \dots & A_{2n-2} & A_{2n-1} & A_{2n} \\
 B_1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & 1 \\
 B_2 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & -1 \\
 B_3 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & -2 & 0 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 B_n & 1 & 1 & 1 & \dots & 1 & 1 & -(n-1) & \dots & 0 & 0 & 0 \\
 B_{n+1} & 1 & 1 & 1 & \dots & 1 & -n & 0 & \dots & 0 & 0 & 0 \\
 B_{n+2} & 1 & 1 & 1 & \dots & -(n-1) & 0 & 0 & \dots & 0 & 0 & 0 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 B_{2n-2} & 1 & 1 & 1 & \ddots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
 B_{2n-1} & 1 & 1 & -2 & \ddots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
 B_{2n} & 1 & -1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0
 \end{pmatrix}$$

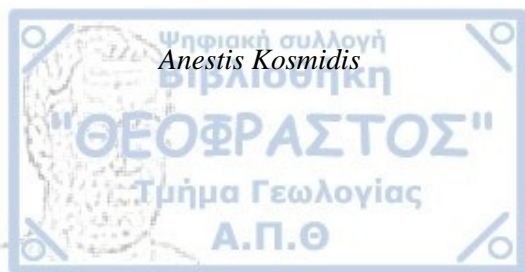
$$\leq n(n+1) \quad (2.118)$$

More generally, if we set A_{2n} and B_{2n} in inequality (2.118) equal to 1, then most of, not all, the reduced inequality: $I_{2n-1}^{reduced} \leq n(n+1)$ is tight (Deng, Zhou, Chen, 2009).

Comments on the criteria:

The tests become more and more stringent.

- 1) The fundamental test is **contextuality** which is applicable for any quantum system. There is always an appropriate collection of operators that can decide at the most basic level whether a system can be described by quantum or classical probability theory.
- 2) The second test is **entanglement** that requires two or more couples of systems.
- 3) The third test is **non-locality** expressed as violation of the **Bell inequality** which is a special case of the second test for many applications. Non locality and entanglement are more useful.



Anestis Kosmidis

CHAPTER 2: SELECTED APPLICATIONS OF QUANTUM STATISTICS FOR THE ANALYSIS OF DATA SETS

2.1 Example: 3-Qubits Wigner-d'Espagnat inequality

Theorem 1 (Wigner-d'Espagnat inequality):

We consider three events A, B, Γ (measurable subsets) of the sample space Ω and we denote by A^c the complementary event of A , then the following inequality holds for the probability measure P (Wigner 1970, d'Espagnat 1979, Bell 1981, Khrennikov 2016):

$$P(A \cap B) + P(B^c \cap \Gamma) \geq P(A \cap \Gamma) \quad (1.1).$$

Proof:

$$\begin{aligned} P(A \cap B) + P(B^c \cap \Gamma) &= \\ &= P((A \cap B) \cap \Omega) + P((B^c \cap \Gamma) \cap \Omega) = \\ &= P((A \cap B) \cap (\Gamma \cup \Gamma^c)) + P((B^c \cap \Gamma) \cap (A \cup A^c)) = \\ &= P((A \cap B \cap \Gamma) \cup (A \cap B \cap \Gamma^c)) + P((B^c \cap \Gamma \cap A) \cup (B^c \cap \Gamma \cap A^c)) = \\ &= P(A \cap B \cap \Gamma) + P(A \cap B \cap \Gamma^c) + P(B^c \cap \Gamma \cap A) + P(B^c \cap \Gamma \cap A^c) = \\ &= P((A \cap \Gamma) \cap B) + P(A \cap B \cap \Gamma^c) + P((A \cap \Gamma) \cap B^c) + P(B^c \cap \Gamma \cap A^c) = \\ &= P(A \cap \Gamma) + P(A \cap B \cap \Gamma^c) + P(B^c \cap \Gamma \cap A^c) \geq P(A \cap \Gamma) \\ &\text{because } P(A \cap B \cap \Gamma^c) \geq 0 \text{ and } P(B^c \cap \Gamma \cap A^c) \geq 0. \blacksquare \end{aligned}$$

Remark 1: Importance of the Wigner-d'Espagnat inequality

The violation of Wigner-d'Espagnat inequality indicates that the observed phenomenon can not be modeled by the Kolmogorov probability and it has to be modeled by quantum probability (Wigner 1970, d'Espagnat 1979, Bell 1981, 1987, Khrennikov 2016). The key point here is that quantum systems can exhibit correlations that are not analogous to classical theories. If Wigner-d'Espagnat inequality holds for a data set, we shall employ classical probability models. On the contrary, if for a given data set Wigner-d'Espagnat inequality fails, we have to employ non-Kolmogorov probability models, like quantum probability models.

Example 1:

We shall the validity of the Wigner-d’Espagnat inequality for 5226 observations of 3 binary variables R, S and E, which correspond to confirmation of substance use (R), existence of phenotypic trait (S) and employment relationship (E). We compute the binary correlations between (R,S), (R,E), (S,E). To simplify the notation and relate directly with previous notation of events, we denote the events as follows:

$$\{R = 0\} = A, \{R = 1\} = A^c, \{S = 0\} = B, \{S = 1\} = B^c, \{E = 0\} = \Gamma, \{E = 1\} = \Gamma^c.$$

The contingency matrices and the joint empirical probability matrices are:

Contingency table of Variables R, S		Variable S		Marginal events of R
		$\{S = 0\} = B$	$\{S = 1\} = B^c$	
Variable R	$\{R = 0\} = A$	333	559	892
	$\{R = 1\} = A^c$	955	3379	4334
Marginal events of S		1288	3938	5226

Table 8

Joint Probability Distribution of Variables R, S		Variable S		Marginal Probabilities of R
		$\{S = 0\} = B$	$\{S = 1\} = B^c$	
Variable R	$\{R = 0\} = A$	0.0637	0.1070	0.1707
	$\{R = 1\} = A^c$	0.1827	0.6466	0.8293
Marginal Probabilities of S		0.2464	0.7536	1

Table 9

Contingency table of Variables R, E		Variable E		Marginal events of R
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable R	$\{R = 0\} = A$	349	543	892
	$\{R = 1\} = A^c$	766	3568	4334
Marginal events of E		1115	4111	5226

Table 10

Joint Probability Distribution of Variables R, E		Variable E		Marginal Probabilities of R
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable R	$\{R = 0\} = A$	0.0668	0.1039	0.1707
	$\{R = 1\} = A^c$	0.1466	0.6827	0.8293
Marginal Probabilities of E		0.2134	0.7866	1

Table 11

Contingency table of Variables S, E		Variable E		Marginal events of S
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable S	$\{S = 0\} = B$	377	911	1288
	$\{S = 1\} = B^c$	738	3200	3938
Marginal events of E		1115	4111	5226

Table 12

Joint Probability Distribution of Variables S, E		Variable E		Marginal Probabilities of S
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable S	$\{S = 0\} = B$	0.0721	0.1743	0.2464
	$\{S = 1\} = B^c$	0.1412	0.6123	0.7535
Marginal Probabilities of E		0.2134	0.7866	1

Table 13

We check the validity of the Law of Total Probability:

$$\begin{aligned}
 P(A) &= P(B)P(A|B) + P(B^c)P(A|B^c) = \\
 &= 0.2464 \cdot \frac{0.0637}{0.2464} + 0.7536 \cdot \frac{0.107}{0.7536} = 0.0637 + 0.107 = 0.1707
 \end{aligned}$$

$$\begin{aligned}
 P(A) &= P(\Gamma)P(A|\Gamma) + P(\Gamma^c)P(A|\Gamma^c) = \\
 &= 0.2134 \cdot \frac{0.0668}{0.2134} + 0.7866 \cdot \frac{0.1039}{0.7866} = 0.0668 + 0.1039 = 0.1707
 \end{aligned}$$

$$\begin{aligned} P(A^c) &= P(B)P(A^c|B) + P(B^c)P(A^c|B^c) = \\ &= 0.2464 \cdot \frac{0.1827}{0.2464} + 0.7536 \cdot \frac{0.6466}{0.7536} = 0.1827 + 0.6466 = 0.8293 \end{aligned}$$

$$\begin{aligned} P(A^c) &= P(\Gamma)P(A^c|\Gamma) + P(\Gamma^c)P(A^c|\Gamma^c) = \\ &= 0.2134 \cdot \frac{0.1466}{0.2134} + 0.7866 \cdot \frac{0.6827}{0.7866} = 0.1466 + 0.6827 = 0.8293 \end{aligned}$$

$$\begin{aligned} P(B) &= P(A)P(B|A) + P(A^c)P(B|A^c) = \\ &= 0.1707 \cdot \frac{0.0637}{0.1707} + 0.8293 \cdot \frac{0.1827}{0.8293} = 0.0637 + 0.1827 = 0.2464 \end{aligned}$$

$$\begin{aligned} P(B) &= P(\Gamma)P(B|\Gamma) + P(\Gamma^c)P(B|\Gamma^c) = \\ &= 0.2134 \cdot \frac{0.0721}{0.2134} + 0.7866 \cdot \frac{0.1743}{0.7866} = 0.0721 + 0.1743 = 0.2464 \end{aligned}$$

$$\begin{aligned} P(B^c) &= P(A)P(B^c|A) + P(A^c)P(B^c|A^c) = \\ &= 0.1707 \cdot \frac{0.107}{0.1707} + 0.8293 \cdot \frac{0.6466}{0.8293} = 0.107 + 0.6466 = 0.7536 \end{aligned}$$

$$\begin{aligned} P(B^c) &= P(\Gamma)P(B^c|\Gamma) + P(\Gamma^c)P(B^c|\Gamma^c) = \\ &= 0.2134 \cdot \frac{0.1412}{0.2134} + 0.7866 \cdot \frac{0.6123}{0.7866} = 0.1412 + 0.6123 = 0.7535 \end{aligned}$$

$$\begin{aligned} P(\Gamma) &= P(B)P(\Gamma|B) + P(B^c)P(\Gamma|B^c) = \\ &= 0.2464 \cdot \frac{0.0721}{0.2464} + 0.7535 \cdot \frac{0.1412}{0.7535} = 0.0721 + 0.1412 = 0.2133 \end{aligned}$$

$$\begin{aligned} P(\Gamma) &= P(A)P(\Gamma|A) + P(A^c)P(\Gamma|A^c) = \\ &= 0.1707 \cdot \frac{0.0668}{0.1707} + 0.8293 \cdot \frac{0.1466}{0.8293} = 0.0668 + 0.1466 = 0.2134 \end{aligned}$$

$$\begin{aligned} P(\Gamma^c) &= P(B)P(\Gamma^c|B) + P(B^c)P(\Gamma^c|B^c) = \\ &= 0.2464 \cdot \frac{0.1743}{0.2464} + 0.7535 \cdot \frac{0.6123}{0.7535} = 0.1743 + 0.6123 = 0.7866 \end{aligned}$$

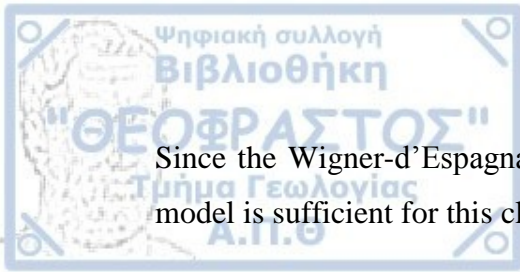
$$\begin{aligned} P(\Gamma^c) &= P(A)P(\Gamma^c|A) + P(A^c)P(\Gamma^c|A^c) = \\ &= 0.1707 \cdot \frac{0.1039}{0.1707} + 0.8293 \cdot \frac{0.6827}{0.8293} = 0.1039 + 0.6827 = 0.7866 \end{aligned}$$

The Law of Total Probability is valid in all cases. ■

We also check the validity of Wigner-d'Espagnat inequality:

$$P(A, B) + P(B^c, \Gamma) = 6.37\% + 14.12\% = 20.49\% \geq 6.68\% = P(A, \Gamma)$$

Since the Wigner-d’Espagnat inequality is not violated, the Kolmogorov probability model is sufficient for this class of data.



Construction of a counterexample:

We construct a data set, which violates the Wigner-d'Espagnat inequality. The constructed data set is given in the following tables for 3 binary variables R, S, E:

Contingency table of Variables R, S		Variable S		Marginal events of R
		$\{S = 0\} = B$	$\{S = 1\} = B^c$	
Variable R	$\{R = 0\} = A$	30	1270	1300
	$\{R = 1\} = A^c$	3746	180	3926
Marginal events of S		3776	1450	5226

Table 14

Joint Probability Distribution of Variables R, S		Variable S		Marginal Probabilities of R
		$\{S = 0\} = B$	$\{S = 1\} = B^c$	
Variable R	$\{R = 0\} = A$	0.0057	0.243	0.2488
	$\{R = 1\} = A^c$	0.7168	0.0344	0.7512
Marginal Probabilities of S		0.7225	0.2774	1

Table 15

Contingency table of Variables R, E		Variable E		Marginal events of R
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable R	$\{R = 0\} = A$	1000	300	1300
	$\{R = 1\} = A^c$	2876	1050	3926
Marginal events of E		3876	1350	5226

Table 16

Joint Probability Distribution of Variables R, E		Variable E		Marginal Probabilities of R
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable R	$\{R = 0\} = A$	0.1914	0.0574	0.2488
	$\{R = 1\} = A^c$	0.5503	0.2009	0.7512
Marginal Probabilities of E		0.7417	0.2583	1

Table 17

Contingency table of Variables S, E		Variable E		Marginal events of S
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable S	$\{S = 0\} = B$	3300	476	3776
	$\{S = 1\} = B^c$	576	874	1450
Marginal events of E		3876	1350	5226

Table 18

Joint Probability Distribution of Variables S, E		Variable E		Marginal Probabilities of S
		$\{E = 0\} = \Gamma$	$\{E = 1\} = \Gamma^c$	
Variable S	$\{S = 0\} = B$	0.6315	0.0911	0.7226
	$\{S = 1\} = B^c$	0.1102	0.1672	0.2774
Marginal Probabilities of E		0.7417	0.2583	1

Table 19

We check the validity of the Law of Total Probability:

$$\begin{aligned}
 P(A) &= P(B)P(A|B) + P(B^c)P(A|B^c) = \\
 &= 0.7225 \cdot \frac{0.0057}{0.7225} + 0.2774 \cdot \frac{0.243}{0.2774} = 0.0057 + 0.243 = 0.2487 \\
 P(A) &= P(\Gamma)P(A|\Gamma) + P(\Gamma^c)P(A|\Gamma^c) = \\
 &= 0.7417 \cdot \frac{0.1914}{0.7417} + 0.2583 \cdot \frac{0.0574}{0.2583} = 0.1914 + 0.0574 = 0.2488 \\
 P(A^c) &= P(B)P(A^c|B) + P(B^c)P(A^c|B^c) = \\
 &= 0.7225 \cdot \frac{0.7168}{0.7225} + 0.2774 \cdot \frac{0.0344}{0.2774} = 0.7168 + 0.0344 = 0.7512 \\
 P(A^c) &= P(\Gamma)P(A^c|\Gamma) + P(\Gamma^c)P(A^c|\Gamma^c) = \\
 &= 0.7417 \cdot \frac{0.5503}{0.7417} + 0.2583 \cdot \frac{0.2009}{0.2583} = 0.5503 + 0.2009 = 0.7512 \\
 P(B) &= P(A)P(B|A) + P(A^c)P(B|A^c) = \\
 &= 0.2488 \cdot \frac{0.0057}{0.2488} + 0.7512 \cdot \frac{0.7168}{0.7512} = 0.0057 + 0.7168 = 0.7225 \\
 P(B) &= P(\Gamma)P(B|\Gamma) + P(\Gamma^c)P(B|\Gamma^c) = \\
 &= 0.7417 \cdot \frac{0.6315}{0.7417} + 0.2583 \cdot \frac{0.0911}{0.2583} = 0.6315 + 0.0911 = 0.7226 \\
 P(B^c) &= P(A)P(B^c|A) + P(A^c)P(B^c|A^c) = \\
 &= 0.2488 \cdot \frac{0.243}{0.2488} + 0.7512 \cdot \frac{0.0344}{0.7512} = 0.243 + 0.0344 = 0.2774
 \end{aligned}$$

$$P(B^c) = P(\Gamma)P(B^c|\Gamma) + P(\Gamma^c)P(B^c|\Gamma^c) =$$

$$= 0.7417 \cdot \frac{0.1102}{0.7417} + 0.2583 \cdot \frac{0.1672}{0.2583} = 0.1102 + 0.1672 = 0.2774$$

$$P(\Gamma) = P(B)P(\Gamma|B) + P(B^c)P(\Gamma|B^c) =$$

$$= 0.7226 \cdot \frac{0.6315}{0.7226} + 0.2774 \cdot \frac{0.1102}{0.2774} = 0.6315 + 0.1102 = 0.7417$$

$$P(\Gamma) = P(A)P(\Gamma|A) + P(A^c)P(\Gamma|A^c) =$$

$$= 0.2488 \cdot \frac{0.1914}{0.2488} + 0.7512 \cdot \frac{0.5503}{0.7512} = 0.1914 + 0.5503 = 0.7417$$

$$P(\Gamma^c) = P(B)P(\Gamma^c|B) + P(B^c)P(\Gamma^c|B^c) =$$

$$= 0.7226 \cdot \frac{0.0911}{0.7226} + 0.2774 \cdot \frac{0.1672}{0.2774} = 0.0911 + 0.1672 = 0.2583$$

$$P(\Gamma^c) = P(A)P(\Gamma^c|A) + P(A^c)P(\Gamma^c|A^c) =$$

$$= 0.2488 \cdot \frac{0.0574}{0.2488} + 0.7512 \cdot \frac{0.2009}{0.7512} = 0.0574 + 0.2009 = 0.2583$$

The Law of Total Probability is valid in all cases. ■

However, the Wigner-d'Espagnat inequality does not hold:

$$P(A, B) + P(B^c, \Gamma) = 0.57\% + 11.02\% = 11.59\% < 19.14\% = P(A, \Gamma).$$

We observe that the Wigner-d'Espagnat inequality is clearly violated for the data set we constructed. Therefore, according to the [Remark 1](#) above, the system of three binary variables R, S and E should be modeled with quantum probability.

We conclude that if we have a data set with three binary variables and we want to correlate the variables two by two in the frame of joint probability, we distinguish two cases. In the first case, the Wigner-d'Espagnat inequality is confirmed, so we model the system with joint probability according to Kolmogorov and examine Pearson correlations between the variables. In the second case, the Wigner-d'Espagnat inequality is violated, so we model the system with quantum probability according to Heisenberg and we examine correlations between the density operators of the variables.

In statistical analysis, we collect data without knowing its true origin nor the type of modeling we should follow. However, the Wigner-d'Espagnat inequality is a criterion for selecting classical or quantum modeling.

2.2 Example: Quantum probability in macroworld. Vessels of water (Aerts, Aerts, Broekaert, Gabora, 2000)

It is shown that Bell inequalities can be violated in the macroscopic world. The macroworld violation is illustrated using an example involving connected vessels of water. It is shown that whether the violation of inequalities occurs in the microworld or in the macroworld, it is the identification of nonidentical events that plays a crucial role. We investigate the violation of Bell inequalities in macroscopic situations and analyze how this indicates the presence of genuine quantum structure. We explicitly challenge the common belief that quantum structure is present only in micro-physical reality (and macroscopic coherent systems), and present evidence that quantum structure can be present in the macro-physical reality.

Specifically, we note that if nonidentical events are consistently differentiated, Bell-type Pitowsky inequalities are no longer violated, even for Bohm's example of two entangled spin 1/2 quantum particles.

Bell inequalities are defined with the following experimental situation in macroworld. We consider a physical entity S , and four experiments e_1, e_2, e_3 and e_4 that can be performed on the physical entity S . Each of the experiments $e_i, i \in \{1, 2, 3, 4\}$ has two possible outcomes, respectively denoted $o_i(up)$ and $o_i(down)$. Some of the experiments can be performed together, which in principle leads to 'coincidence' experiments $e_{ij}, i, j \in \{1, 2, 3, 4\}$.

For example e_i and e_j together will be denoted e_{ij} . Such a coincidence experiment e_{ij} has four possible outcomes, namely $(o_i(up), o_j(up))$, $(o_i(up), o_j(down))$, $(o_i(down), o_j(up))$ and $(o_i(down), o_j(down))$. Following Bell, we introduce the expectation values $E_{ij}, i, j \in \{1, 2, 3, 4\}$ for these coincidence experiments, as:

$$E_{ij} = +1 \cdot P(o_i(up), o_j(up)) + 1 \cdot P(o_i(down), o_j(down)) - 1 \cdot P(o_i(up), o_j(down)) - 1 \cdot P(o_i(down), o_j(up)) \quad (2.1)$$

From the assumption that the outcomes are either +1 or -1, and that the correlation E_{ij} can be written as an integral over some hidden variable of a product of the two local outcome assignments, one derives Bell inequalities:

$$|E_{13} - E_{14}| + |E_{23} + E_{24}| \leq 2 \quad (2.2)$$

Hence we have the coincidence experiments e_{13}, e_{14}, e_{23} and e_{24} , but instead of concentrating on the expectation values they introduce the coincidence probabilities p_{13}, p_{14}, p_{23} and p_{24} , together with the probabilities p_2 and p_4 . Concretely, p_{ij} means the probability that the coincidence experiment p_{ij} gives the outcome $(o_i(up), o_j(down))$, while p_i means the probability that the experiment e_i gives the outcome $o_i(up)$. The Clauser Horne inequalities then read:

$$-1 \leq p_{14} - p_{13} + p_{23} + p_{24} - p_2 - p_4 \leq 0 \quad (2.3)$$

Although the Clauser Horne inequalities are thought to be equivalent to Bell inequalities, they are of a slightly more general theoretical nature, and lend themselves to Pitowsky's generalization, which will play an important role in our theoretical analysis.

We review an example of a macroscopic situation where Bell inequalities and Clauser Horne inequalities are violated. Consider an entity S which is a container with 20 liters of transparent water (Figure 9), in a state s such that the container is placed in the gravitational field of the earth, with its bottom horizontal.

We introduce the experiment e_1 that consists of putting a siphon K_1 in the container of water at the left, taking out water using the siphon, and collecting this water in a reference vessel R_1 placed to the left of the container. If we collect more than 10 liters of water, we call the outcome $o_1(up)$, and if we collect less or equal to 10 liters, we call the outcome $o_1(down)$. We introduce another experiment e_2 that consists of taking with a little spoon, from the left, a bit of the water, and determining whether it is transparent. We call the outcome $o_2(up)$ when the water is transparent and the outcome $o_2(down)$ when it is not. We introduce the experiment e_3 that consists of putting a siphon K_3 in the container of water at the right, taking out water using the siphon, and collecting this water in a reference vessel R_3 to the right of the

container. If we collect more or equal to 10 liters of water, we call the outcome $o_3(up)$, and if we collect less than 10 liters, we call the outcome $o_3(down)$. We also introduce the experiment e_4 which is analogous to experiment e_2 , except that we perform it to the right of the container.

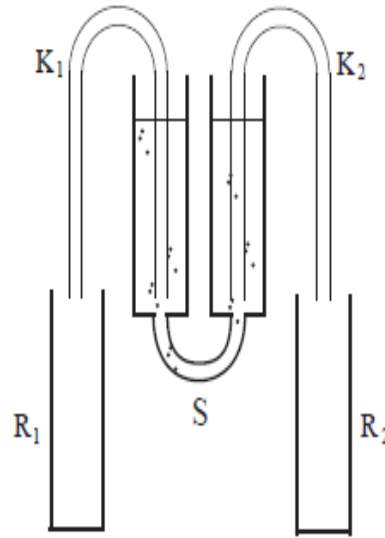


Figure 9: The vessels of water example violating Bell inequalities. The entity S consists of two vessels containing 20 liters of water that are connected by a tube. Experiments are performed on both sides of the entity S by introducing syphons K_1 and K_2 in the respective vessels and pouring out the water and collecting it in reference vessels R_1 and R_2 . Carefully chosen experiments reveal that Bell inequalities are violated by this entity S (Aerts, Aerts, Broekaert, Gabora, 2000).

Clearly, for the container of water being in state s , experiments e_1 and e_3 give with certainty the outcome $o_1(up)$ and $o_3(up)$, which shows that $p_1 = p_3 = 1$. Experiments e_2 and e_4 give with certainty the outcome $o_2(up)$ and $o_4(up)$, which shows that $p_2 = p_4 = 1$.

The experiment e_1 can be performed together with experiments e_3 and e_4 , and we denote the coincidence experiments e_{13} and e_{14} . Also, experiment e_2 can be performed together with experiments e_3 and e_4 , and we denote the coincidence

experiments e_{23} and e_{24} . For the container in state s , the coincidence experiment e_{13} always gives one of the outcomes $(o_1(up), o_3(down))$ or $(o_1(down), o_3(up))$, since more than 10 liters of water can never come out of the vessel at both sides. This shows that $E_{13} = -1$ and $p_{13} = 0$. The coincidence experiment e_{14} always gives the outcome $(o_1(up), o_4(up))$ which shows that $E_{14} = +1$ and $p_{14} = +1$, and the coincidence experiment e_{23} always gives the outcome $(o_2(up), o_3(up))$ which shows that $E_{23} = +1$ and $p_{23} = +1$. Clearly experiment e_{24} always gives the outcome $(o_2(up), o_4(up))$ which shows that $E_{24} = +1$ and $p_{24} = +1$. Let us now calculate the terms of Bell inequalities:

$$|E_{13} - E_{14}| + |E_{23} + E_{24}| = |-1 - 1| + |1 + 1| = 2 + 2 = 4 \quad (2.4)$$

and of the Clauser Horne inequalities:

$$p_{14} - p_{13} + p_{23} + p_{24} - p_2 - p_4 = 1 - 0 + 1 + 1 - 1 - 1 = 1 \quad (2.5)$$

This shows that Bell inequalities and Clauser Horne inequalities can be violated in macroscopic reality. It is even so that the example violates the inequalities more than the original quantum example of the two coupled spin- 1/2 entities.

If we get more than four events, and unfortunately the original Bell inequalities are out of their domain of applicability. However Pitowsky has developed a generalization of Bell inequalities where any number of experiments and events can be taken into account, and as a consequence we can check whether the new situation violates Pitowsky inequalities.

If Pitowsky inequalities would not be violated in the vessels of water model, while for the microscopic Bohm example they would, then this would 'prove' the different status of the two examples, the macroscopic being 'false', due to lack of correctly distinguishing between events, and the microscopic being genuine.

More specifically, let S be a set of pairs of integers from $\{1, 2, \dots, n\}$ that is:

$$S \subseteq \{\{i, j\} \mid 1 \leq i < j \leq n\} \quad (2.6)$$

Let $R(n, S)$ denote the real space of all functions $f : \{1, 2, \dots, n\} \cup S \mapsto \mathbb{R}$. We denote vectors in $R(n, S)$ by $f = (f_1, f_2, \dots, f_n, \dots, f_{ij}, \dots)$, where the f_{ij}

appear in a lexicographic order on the i, j 's. Let $\{0, 1\}^n$ be the set of all n -tuples of zeroes and one's. We denote elements of $\{0, 1\}^n$ by $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ where $\varepsilon_j \in \{0, 1\}$. For each $\varepsilon \in \{0, 1\}^n$ let u^ε be the following vector in $R(n, S)$:

$$u_j^\varepsilon = \varepsilon_j, \quad 1 \leq j \leq n$$

$$u_{ij}^\varepsilon = \varepsilon_i \varepsilon_j, \quad \{i, j\} \in S$$

The classical correlation polytope $C(n, S)$ is the closed convex hull in $R(n, S)$ of all 2^n possible vectors $u^\varepsilon, \varepsilon \in \{0, 1\}^n$.

Let $p = (p_1, \dots, p_n, \dots, p_{ij}, \dots)$ be a vector in $R(n, S)$. Then $p \in C(n, S)$ if there is a Kolmogorovian probability space (X, M, μ) and (not necessarily distinct) events $A_1, A_2, \dots, A_n \in M$ such that:

$$p_i = \mu(A_i), \quad 1 \leq i \leq n, \quad p_{ij} = \mu(A_i \cap A_j), \quad \{i, j\} \in S$$

where X is the space of events and μ the probability measure (Pitowsky, 1989).

To illustrate the above theorem and at the same time the connection with Bell inequalities and the Clauser Horne inequalities, we consider some specific examples of Pitowsky's theorem. The case $n = 4$ and $S = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}\}$. The condition $p \in C(n, S)$ is then equivalent to the Clauser-Horne inequalities:

$$0 \leq p_{ij} \leq p_i \leq 1$$

$$0 \leq p_{ij} \leq p_j \leq 1, \quad i = 1, 2 \text{ and } j = 3, 4 \quad (2.7)$$

$$p_i + p_j - p_{ij} \leq 1$$

$$-1 \leq p_{13} + p_{14} + p_{24} - p_{23} - p_1 - p_4 \leq 0$$

$$-1 \leq p_{23} + p_{24} + p_{14} - p_{13} - p_2 - p_4 \leq 0$$

$$-1 \leq p_{14} + p_{13} + p_{23} - p_{24} - p_1 - p_3 \leq 0 \quad (2.8)$$

$$-1 \leq p_{24} + p_{23} + p_{13} - p_{14} - p_2 - p_3 \leq 0$$

The case $n = 3$ and $S = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. We find then the following inequalities equivalent to the condition $p \in C(n, S)$:

$$0 \leq p_{ij} \leq p_i \leq 1$$

$$0 \leq p_{ij} \leq p_j \leq 1, \quad 1 \leq i < j \leq 3$$

(2.9)

$$p_i + p_j - p_{ij} \leq 1$$

$$p_1 + p_2 + p_3 - p_{12} - p_{13} - p_{23} \leq 0$$

$$p_1 - p_{12} - p_{13} + p_{23} \leq 0$$

$$p_2 - p_{12} - p_{23} + p_{13} \leq 0$$

(2.10)

$$p_3 - p_{13} - p_{23} + p_{12} \leq 0$$

2.3 Example: Quantum probability in cognition. Cats (Aerts, Aerts, Broekaert, Gabora, 2000)

We study how Bell inequalities can be violated in cognition, specifically in the relationship between abstract concepts and specific instances of these concepts. This supports the hypothesis that genuine quantum may represent mental processing and cognition. We introduce a model where the amount of nonlocality and the degree of quantum uncertainty are parameterized and demonstrate that increasing nonlocality increases the degree of violation, while increasing quantum uncertainty decreases the degree of violation. So we show how Bell inequalities are violated in the mind in virtue of the relationship between abstract concepts and specific instances of them. We investigate how concepts violate Bell inequalities.

As in macroworld, Bell inequalities are defined with the following experimental situation in mind. We consider a physical entity S , and four experiments e_1, e_2, e_3 and e_4 that can be performed on the physical entity S . Each of the experiments $e_i, i \in \{1, 2, 3, 4\}$ has two possible outcomes, respectively denoted $o_i(up)$ and $o_i(down)$. Some of the experiments can be performed together, which in principle leads to ‘coincidence’ experiments $e_{ij}, i, j \in \{1, 2, 3, 4\}$.

For example e_i and e_j together will be denoted e_{ij} . Such a coincidence experiment e_{ij} has four possible outcomes, namely $(o_i(up), o_j(up))$, $(o_i(up), o_j(down))$, $(o_i(down), o_j(up))$ and $(o_i(down), o_j(down))$. Following Bell, we introduce the expectation values $E_{ij}, i, j \in \{1, 2, 3, 4\}$ for these coincidence experiments, as:

$$E_{ij} = +1 \cdot P(o_i(up), o_j(up)) + 1 \cdot P(o_i(down), o_j(down)) - 1 \cdot P(o_i(up), o_j(down)) - 1 \cdot P(o_i(down), o_j(up)) \quad (3.1)$$

From the assumption that the outcomes are either +1 or -1, and that the correlation E_{ij} can be written as an integral over some hidden variable of a product of the two local outcome assignments, one derives Bell inequalities:

$$|E_{13} - E_{14}| + |E_{23} + E_{24}| \leq 2 \quad (3.2)$$

Hence we have the coincidence experiments e_{13}, e_{14}, e_{23} and e_{24} , but instead of concentrating on the expectation values they introduce the coincidence probabilities p_{13}, p_{14}, p_{23} and p_{24} , together with the probabilities p_2 and p_4 .

Concretely, p_{ij} means the probability that the coincidence experiment p_{ij} gives the outcome $(o_i(up), o_j(down))$, while p_i means the probability that the experiment e_i gives the outcome $o_i(up)$. The Clauser Horne inequalities then read:

$$-1 \leq p_{14} - p_{13} + p_{23} + p_{24} - p_2 - p_4 \leq 0 \quad (3.3)$$

To make things more concrete we present an example. Keynote players in this example are the two cats, Glimmer and Inkling, that live at our research center. The experimental situation has been set up by one of the authors (Diederik) to show that the mind of another of the authors (Liane) violates Bell inequalities. The situation is as follows. On the table where Liane prepares the food for the cats is a little note that says: 'Think of one of the cats now'. To show that Bell inequalities are violated we must introduce four experiments e_1, e_2, e_3 and e_4 . Experiment e_1 consists of Glimmer showing up at the instant Liane reads the note.

If, as a result of the appearance of Glimmer and Liane reading the note, the state of her mind is changed from the more general concept 'cat' to the instance 'Glimmer', we call the outcome $o_1(up)$, and if it is changed to the instance 'Inkling', we call the outcome $o_1(down)$.

Experiment e_3 consists of Inkling showing up at the instant that Liane reads the note. We call the outcome $o_3(up)$ if the state of her mind is changed to the instance 'Inkling', and $o_3(down)$ if it is changed to the instance 'Glimmer', as a result of the appearance of Inkling and Liane reading the note. The coincidence experiment e_{13} consists of Glimmer and Inkling both showing up when Liane reads the note. The outcome is $(o_1(up), o_3(down))$ if the state of her mind is changed to the instance 'Glimmer', and $(o_1(down), o_3(up))$ if it changes to the instance 'Inkling' as a consequence of their appearance and the reading of the note.

Now it is necessary to know that occasionally the secretary puts bells on the cats' necks, and occasionally she takes the bells off. Thus, when Liane comes to work, she does not know whether or not the cats will be wearing bells, and she is always curious to know. Whenever she sees one of the cats, she eagerly both looks and listens

for the bell. Experiment e_2 consists of Liane seeing Inkling and noticing that she hears a bell ring or doesn't. We give the outcome $o_2(up)$ to the experiment e_2 when Liane hears the bell, and $o_2(down)$ when she does not. Experiment e_4 is identical to experiment e_2 except that Inkling is interchanged with Glimmer.

The coincidence experiment e_{14} consists of Liane reading the note, and Glimmer showing up, and her listening to whether a bell is ringing or not. It has four possible outcomes: $(o_1(up), o_4(up))$ when the state of Liane's mind is changed to the instance 'Glimmer' and she hears a bell; $(o_1(up), o_4(down))$ when the state of her mind is changed to the instance 'Glimmer' and she does not hear a bell; $(o_1(down), o_4(up))$ when the state of her mind is changed to the instance 'Inkling' and she hears a bell and $(o_1(down), o_4(down))$ when the state of her mind is changed to the instance 'Inkling' and she does not hear a bell. The coincidence experiment e_{23} is defined analogously. It consists of Liane reading the note and Inkling showing up and her listening to whether a bell is ringing or not. It too has four possible outcomes: $(o_2(up), o_3(up))$ when she hears a bell and the state of her mind is changed to the instance 'Inkling'; $(o_2(up), o_3(down))$ when she hears a bell and the state of her mind is changed to the instance 'Glimmer'; $(o_2(down), o_3(up))$ when she does not hear a bell and the state of her mind is changed to the instance 'Inkling' and $(o_2(down), o_3(down))$ when she does not hear a bell and the state of her mind is changed to the instance 'Glimmer'.

The coincidence experiment e_{24} is the experiment where Glimmer and Inkling show up and Liane listens to see whether she hears the ringing of bells. It has outcome $(o_2(up), o_4(up))$ when both cats wear bells, $(o_2(up), o_4(down))$ when only Inkling wears a bell, $(o_2(down), o_4(up))$ when only Glimmer wears a bell and $(o_2(down), o_4(down))$ when neither cat wears a bell.

We now formulate the necessary conditions such that Bell inequalities are violated in this experiment:

- (1) The categorical concept 'cat' is activated in Liane's mind.
- (2) She does what is written on the note.

(3) When she sees Glimmer, there is a change of state, and the categorical concept 'cat' changes to the instance 'Glimmer', and when she sees Inkling it changes to the instance 'Inkling'.

(4) Both cats are wearing bells around their necks.

The coincidence experiment e_{13} gives outcome $(o_1(up), o_3(down))$ or $(o_1(down), o_3(up))$ because indeed from (2) it follows that Liane will think of Glimmer or Inkling. This means that $E_{13} = -1$. The coincidence experiment e_{14} gives outcome $(o_1(up), o_4(up))$, because from (3) and (4) it follows that she thinks of Glimmer and hears the bell. Hence $E_{14} = +1$. The coincidence experiment e_{23} also gives outcome $(o_2(up), o_3(up))$, because from (3) and (4) it follows that she thinks of Inkling and hears the bell. Hence $E_{23} = +1$. The coincidence experiment e_{24} gives $(o_2(up), o_4(up))$, because from (4) it follows that she hears two bells. Hence $E_{24} = +1$. As a consequence we have:

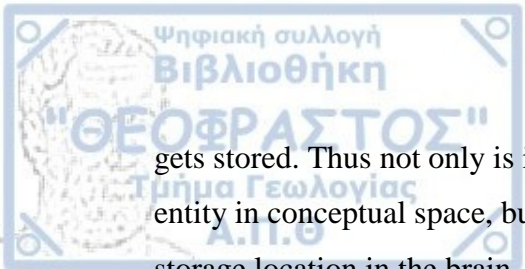
$$|E_{13} - E_{14}| + |E_{23} + E_{24}| = +4$$

The reason that Bell inequalities are violated is that Liane's state of mind changes from activation of the abstract categorical concept 'cat', to activation of either 'Glimmer' or 'Inkling'. We can thus view the state 'cat' as an entangled state of these two instances of it.

Our example shows that concepts in the mind violate Bell inequalities, and hence entail nonlocality in the sense that physicists use the concept.

As a first approximation, we can say that the nonlocality of stored experiences and concepts arises from their distributed nature. Each concept is stored in many memory locations; likewise, each location participates in the storage of many concepts. In order for the mind to be capable of generating a stream of meaningfully-related yet potentially creative reminders, the degree of this distribution must fall within an intermediate range. Thus, a given experience activates not just one location in memory, nor does it activate every memory location to an equal degree, but activation is distributed across many memory locations, with degree of activation falling with distance from the most activated one.

Memory is also content addressable, meaning that there is a systematic relationship between the content of an experience, and the place in memory where it



gets stored. Thus not only is it not localized as an episodic memory or conceptual entity in conceptual space, but it is also not localized with respect to its physical storage location in the brain.

Over the past several decades, numerous attempts have been made to forge a connection between quantum mechanics and the mind. In these approaches, it is generally assumed that the only way the two could be connected is through micro-level quantum events in the brain exerting macro-level effects on the judgements, decisions, interpretations of stimuli, and other cognitive functions of the conscious mind.

From the preceding arguments, it should now be clear that this is not the only possibility. If quantum structure can exist at the macro-level, then the process by which the mind arrives at judgements, decisions, and stimulus interpretations could itself be quantum in nature.

We should point out that we are not suggesting that the mind is entirely quantum. Clearly not all concepts and instances in the mind are entangled or violate Bell inequalities. Our claim is simply that the mind contains some degree of quantum structure.

On the other side, both these hidden variable models (Aerts and Pitowsky) are based on an observation that a structure of conditional probabilities characteristic for systems with spin is not a Kolmogorovian one. The problem is rooted in a non-Bayesian structure of such probabilities and is typically manifested by a violation of Bell's inequality. Both Aerts' and Pitowsky's models are not about simultaneous measurements as we have in the EPR-Bohm framework, but about conditional measurements. To define conditional measurements, we assume we have some state φ , perform a measurement of an observable α and the state φ is changed; in a new post α -measurement state we perform a measurement of another observable b which is incompatible with (or complementary to) α . We point out both Aerts' and Pitowsky's models were created in relation to Bell's inequality. However, there is no contradiction with the Bell Theorem, because it is impossible to derive the ordinary Bell's inequality for this model, because we cannot perform a simultaneous measurement of α and b . So instead of Bell's inequality for the simultaneous probability distributions, one can derive Bell's inequality for conditional probabilities,

as it has been demonstrated that this inequality can be applied to conditional measurements. Moreover, it has been demonstrated that it is violated by quantum model. We remark this conditional probability inequality is based only on the assumption that we can use Bayes' formula for conditional probabilities. Since both Aerts' and Pitowsky's models reproduce quantum probabilities, Bell's inequality for conditional probabilities is automatically violated for these models (Khrennikov 2003, 2016).

CHAPTER 3: QUANTUM MACHINE LEARNING VARIABLES AND CAVEATS

Machine learning algorithms construct and/or update their predictive model based on input data. A number of advances in the field of quantum information shows that particular quantum algorithms can offer a speedup over their classical counterparts (Jordan, 2018). It has been speculated that application of these techniques to the field of machine learning may produce similar results (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanisic, 2015).

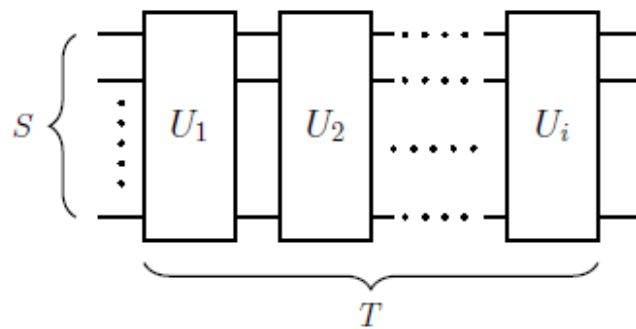


Figure 10: The roles of space S and time T in the circuit model for quantum computation (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanisic, 2015).

In order to understand the potential benefits of Quantum Machine Learning (QML), it must be possible to make comparisons between classical and quantum machine learning algorithms, in terms of speed and classifier performance. To compare algorithms, computer scientists consider two characteristic resources:

- Space, S : The amount of computational space needed to run the algorithm. Formally, 'space' refers to the number of qubits required. For S qubits, the dimension of the relevant Hilbert space is 2^S . It is important to distinguish between these two quantities, as there is an exponential factor between them.

- Time, T : The time taken to train and then classify within a specified error. Formally, 'time' refers to the number of operations required and, in the quantum circuit model, can be expressed as the number of consecutive gates applied to the qubits.

Figure 10 shows how time and space are represented in quantum circuit diagrams.

These are typically functions of the following variables:

- Size of training data set, n : The number of data points in the training set supplied to an algorithm.
- Size of input data set, N : The number of data points to be classified by an algorithm.
- Dimension of data points, m : The number of parameters for each data point. In machine learning, each data point is often treated as a vector, where the numeric value associated with each feature is represented as a component of the vector.
- Error, ϵ : The fraction of incorrect non-training classifications made by the algorithm (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanislav, 2015).

We outline some challenges for quantum machine learning that we believe should be taken into account when designing new algorithms and/or architectures. We start from some common early pitfalls in quantum algorithm design.

More specifically, an often overlooked aspect of quantum algorithms is state preparation. Arbitrary state preparation is exponentially hard in the number of qubits for discrete gate sets, providing a bound on the performance of all algorithms, and placing a restriction on the types of states used in initializing an algorithm. Moreover, there exist cases where this addition to the algorithm's complexity is ignored (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanislav, 2015).

We know measurement of a quantum mechanical system results in the collapse of the system's wave function to a single eigenstate of the measurement operator. Although it is possible to learn the pre-measurement state using a number of trials exponential in system size, this will kill any potential speedup. Therefore, any algorithm which outputs all of the amplitudes of the final state $|x\rangle$, suffers exponential costs. The only information that can be easily extracted from $|x\rangle$ is a global statistical property, such as the inner product, $\langle x|z\rangle$, with some fixed reference state $|z\rangle$, or the location of the dominant amplitudes of $|x\rangle$. This argues against the

existence of a useful quantum algorithm that stores output data in the exponentially large Hilbert space of a quantum state - the data would be impossible to retrieve (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanisic, 2015).

We consider how to encode classical data into quantum states. This procedure is an important part of any quantum algorithm. In terms of state preparation, information is typically encoded in state amplitudes. So given a vector $x \in \mathbb{R}^N$ stored in memory, we create copies of the state:

$$|x\rangle = \frac{1}{|x|} \sum_i x_i |i\rangle \quad (3.1) \text{ (Prakash, 2014)}$$

In the context of the analysis of classical data, we can exploit the encoding of quantum information to efficiently represent classical probability distributions with exponentially many points. For instance, when $v = (v_1, \dots, v_{2^n})$ is a probability vector of size 2^n , we can write an n-qubit state (register):

$$\psi = \sum_{i=1}^{2^n} \sqrt{v_i} e_i \quad (3.2)$$

Quantum Random Access Memory (QRAM) is a theoretical oracle that stores quantum states and allows queries to be made in superposition. The efficiency of the oracle removes any overheads for arbitrary state preparation, which could suppress the claimed quantum speedup of an algorithm.

It is possible to use QRAM to generate a quantum state from the n-dimensional vector x , in time $O(\sqrt{n})$. However, by pre-processing the vector, this can be improved to $O(\text{polylog}(n))$ (Prakash, 2014).

The inclusion of QRAM in QML proposals is troubling, both from a theoretical and an experimental perspective. However ruling out QRAM does not necessarily mean no data sets can be loaded into a quantum state efficiently. If the coefficients to be loaded into a state are given by an explicit formula, it may be possible for a quantum computer to prepare said state independently, without consulting a QRAM. This sample can be loaded into a superposition over n qubits efficiently, provided there is an efficient classical algorithm to integrate the function over an arbitrary interval. Therefore, it is a strong indication that a total dependence on QRAM is not necessary (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Stanisic, 2015).

The table below (Table 36) presents some algorithms and includes, where possible, the advantage the quantum algorithm gains over its classical counterpart and any conditions required for the speedup to be maintained.

Algorithm	Quantum Time Scaling	Quantum Space Scaling	Classical Time Scaling	Definition of Terms	Quantum Speedup?	Comments
Quantum Dot-based Artificial Neural Network [18, 24, 25, 26, 27, 28].	No rigorous bounds.	-	-	-	Not available.	Might provide spatial benefits because of capability to create temporal neural network, but no rigorous analysis found in papers.
Superposition-based Learning Algorithm (WNN) [32].	$O(\text{poly}(N_p))$	-	-	N_p : No. training patterns.	Unknown.	Assuming pyramidal QRAM network, and two inputs to every QRAM node.
Associative Memory using Stochastic Quantum Walk [33].	No rigorous bounds.	-	-	κ : Coherent weights. γ : Decoherent weights.	Not available.	Authors identify dependence of runtime on κ and γ with no detailed analysis offered.
Probabilistic Quantum Memories [38].	No rigorous bounds.	-	-	-	Not available.	Specifically notes no discussion of "possible quantum speedup" as "[...] the main point of the present Letter is the exponential storage capacity with retrieval of noisy inputs".
Quantum Deep Learning (Gibbs Sampling) [44].	$\tilde{O}(NE\sqrt{\kappa})$	$O(n_h + n_v + \log \epsilon^{-1})$	$\tilde{O}(NLEk)$ $O(NE\kappa)$ L : No. layers. k : No. sample sets.	N : No. training vectors. E : No. edges. κ : Scaling factor. x : Training vector. n_h : No. hidden nodes. n_v : No. visible nodes.	Asymptotic advantage.	The quantum advantage lies in the found solution being 'exact' up to ϵ . The classical algorithm converges to an approximate solution.
Quantum Deep Learning (Amplitude Estimation) [44].	$\tilde{O}(\sqrt{N}E^2\sqrt{\kappa})$	$O(n_h + n_v + \log \epsilon^{-1})$	$\tilde{O}(NLEk)$ $O(NE\kappa)$ L : No. layers. k : No. sample sets.	N : No. training vectors. E : No. edges. κ : Scaling factor. x : Training vector. n_h : No. hidden nodes. n_v : No. visible nodes.	Quadratic advantage in number of training vectors.	Assumes access to training data in quantum oracle. The dependence on E can be reduced quadratically in some cases.
Quantum Deep Learning (Amplitude Estimation and QRAM). [44]	$\tilde{O}(\sqrt{N}E^2\sqrt{\kappa})$	$O(N + n_h + n_v + \log \epsilon^{-1})$	$\tilde{O}(NLEk)$ $O(NE\kappa)$ L : No. layers. k : No. sample sets.	N : No. training vectors. E : No. edges. κ : Scaling factor. x : Training vector. n_h : No. hidden nodes. n_v : No. visible nodes.	Same as Quantum Deep Learning (Amplitude Estimation).	Assumes QRAM allows simultaneous operations on different qubits at unit cost.
Linear Systems Solving (HHL) [55, 56].	Quantum state output: $O(\text{poly} \log(N), \text{poly} \log(\epsilon^{-1}))$ Classical output: $O(\text{poly} \log N, \text{poly}(\epsilon^{-1}))$	$O(\log N)$	$O(\text{poly}(N) \log(\epsilon^{-1}))$	N : Dimension of the system. ϵ : Error.	Exponential if the desired output is a quantum state. Complicated otherwise.	As lots of classes of problems can be reduced to linear system solving, the quantum advantage is heavily dependent on context.
Quantum Principal Component Analysis [59].	$O(\log d)$	-	$O(d)$	d : Dimensions of data space.	Exponential in d .	Speedup only valid for spaces dominated by few principal components. Algorithm requires QRAM.
Quantum Nearest Centroid (sub-routine in k-means) [61, 62].	$O(\epsilon^{-1} \log nm)$	-	$O(nm)$	n : No. training vectors. m : Length of vectors.	Exponential speedup, however average classical runtime can be $\log(nm)/(\epsilon^2)$.	Assumes all vectors and amplitudes stored in QRAM, otherwise speedup vanishes. Quantum Support Vector Machine is Quantum Nearest Centroid with two clusters.
k-nearest Neighbours [64].	$\tilde{O}(\sqrt{n} \log n)$ (first order)	-	$O(nm)$	n : No. training vectors.	Quantum advantage for high-dimensional vector spaces.	Reduces to nearest centroid for $k=1$.
Minimum Spanning Tree [68].	$\Theta(N^{3/2})$	-	$\Omega(N^2)$	N : No. points in dataset.	Polynomial in the sub-routine used to find a minimum spanning tree.	Matrix model.
Quantum Perceptron (Data in States) [16, 39, 96].	No rigorous bounds.	-	$O(W + N \log(\epsilon^{-1}))$	W : No. of weights. N : Size of training dataset. ϵ : Allowed error.	Not available.	Producing a speedup appears to be difficult using these algorithms (see Appendix A, from which the classical bound has been taken).
Quantum Perceptron (Weights in States) [Appx. A].	$O(W + \log(N) \log(\epsilon^{-1}))$	-	$O(W + N \log(\epsilon^{-1}))$	W : No. of weights. N : Size of training dataset. ϵ : Allowed error.	Exponential in N .	-

Table 36: Table of Quantum Algorithms Advantages (Adcock, Allen, Day, Frick, Hinchliff, Johnson, Staniscic, 2015).

We recommend some cases where quantum theory helps Machine Learning (ML):

Example Principal Component Analysis (PCA):

Data: classical vectors $v_1, \dots, v_N \in \mathbb{R}^d$. For example:

- j_{th} entry of v_i counts number of times document i contains keyword j

- j_{th} entry of v_i indicates whether buyer i bought product j

PCA finds the principal components of correlation matrix:

$$A = \sum_{i=1}^N v_i v_i^T \quad (3.3)$$

Main eigenvectors describe patterns in the data. Can be used to summarize data, for prediction, etc (Lloyd, Mohseni, Rebentrost, 2014).

Idea for quantum speed-up :

If we can efficiently prepare the $|v_i\rangle$ as $\log_2(d) - \text{qubit}$ states, then doing this for random i gives mixed state : $\rho = \frac{1}{N} A$ (3.4)

where the equation (3.4) is the condition for Quantum Algorithm Implementation.

We want to sample (eigenvector, eigenvalue)-pairs from ρ (Lloyd, Mohseni, Rebentrost, 2014).

- Using few copies of ρ , we want to run $U = e^{-i\rho}$ on some σ

- Idea: start with $\sigma \otimes \rho$, apply SWAP, throw away 2nd register.

1st register now has $U^\varepsilon \sigma (U^\dagger)^\varepsilon$, up to error $O(\varepsilon^2)$.

Repeat this $1/\varepsilon$ times, using a fresh copy of ρ each time.


First register now contains $U \sigma U^\dagger$, up to error $\frac{1}{\varepsilon} O(\varepsilon^2) = O(\varepsilon)$.

- Suppose ρ has eigendecomposition: .

Phase estimation maps $|w_i\rangle|0\rangle \mapsto |w_i\rangle|\lambda'_i\rangle$, where $|\lambda_i - \lambda'_i| < \delta$, using $O(1/\delta)$ applications of U .

-Phase estimation on another fresh copy of ρ maps:

$$\rho \otimes |0\rangle\langle 0| \mapsto \sum_i \lambda_i |w_i\rangle\langle w_i| \otimes |\lambda'_i\rangle\langle \lambda'_i|$$



Measuring 2^{nd} register samples $|w_i\rangle|\lambda'_i\rangle$ with probability λ_i (Lloyd, Mohseni, Rebentrost, 2014).

Fast linear algebra with quantum mechanics:

A significant number of methods in the quantum machine learning literature is based on fast quantum algorithms for linear algebra. We discuss about the two main quantum sub routines for linear algebra: a quantum algorithm for matrix inversion and a quantum algorithm for singular value decomposition.

Fast matrix inversion: the quantum linear system algorithm

We know for a system of linear equations $Ax = b$ with $A \in \mathbb{R}^{N \times N}$ and $x, b \in \mathbb{R}^N$, the best classical algorithm has a runtime of $O(N^{2.373})$ (Coppersmith, Winograd, 1990). However, due to a large pre-factor, the algorithm is not used in practice. Standard methods, for example, based on QR-factorisation (Q is an orthogonal matrix, R is an upper triangular matrix) take $O(N^3)$ steps (Golub, Van Loan, 1996). The quantum linear system algorithm (QLSA) promises to solve the problem in $O(\log(N) \kappa^2 \frac{s^2}{\epsilon})$, where κ is the condition number, defined to be the ratio of the largest to the smallest eigenvalue, s is the sparsity or the maximum number of non-zero entries in a row and column of A and ϵ is the precision to which the solution is approximated (Harrow, Hassidim, Lloyd, 2009).

Although the QLSA algorithm solves matrix inversion in logarithmic time a number of caveats might limit its applicability to practical problems. First, the QLSA algorithm requires the matrix A to be sparse. Second, the classical data must be loaded in quantum super position in logarithmic time. Third, the output of the algorithm is not x itself but a quantum state that encodes the entries of x in superposition. Fourth, the condition number must scale at most sublinearly with N (Harrow, Hassidim, Lloyd, 2009).

We present a general comparison of the asymptotic scalings of classical, quantum and parallel algorithms for linear algebra and their major applications in machine learning in Table 37. With optimal learning rates we mean that any learning algorithm cannot achieve better prediction performance (uniformly) on the class of

problems considered. Interestingly, such assumptions also allow us to derive estimates for the condition number of the kernel matrix to be of order $\kappa = O(N^{1/2})$.

Problem	Scaling	Applications
Solving linear system of equations	C: $\tilde{O}(s\kappa N \log(1/\epsilon))$ [She94] ^a	Least-square-SVM [RML14]
	Q: $\tilde{O}(s^2\kappa^2 \log(N)/\epsilon)$ [HHL09] ^b	GP Regression [ZFF15]
	P: $\mathcal{O}(\log^2(N) \log(1/\epsilon))$ [Csa76] ^c	Kernel Least Squares [SSP16]
Singular value estimation	C: $\mathcal{O}(k^2 N \log(1/\delta)/\epsilon)$ [FKV04] ^d	Recommendation Systems [KP16]
	Q: $\mathcal{O}(\log(N)\epsilon^{-3})$ [LMR14]	Linear Regression [SSP16]
	P: $\mathcal{O}(\log^2(N) \log(1/\epsilon))$ [Csa76] ^e	Principal Component Analysis [LMR14] ^f

Quantum linear algebra algorithms and their machine learning applications. When carefully compared with classical versions that take into account the same caveats, quantum algorithms might lose their advantages. C,Q,P indicate, respectively, the asymptotic computational complexity for classical, quantum, and parallel computation. We remind the reader that, to date, memory and bandwidth limits in the communication between processors make the implementation of certain parallel algorithms unrealistic. Given an $N \times N$ dimensional matrix A , we denote by k the number of singular values that are computed by the algorithm, by s the sparsity, and by κ the condition number. For approximation algorithms ϵ is an approximation parameter. In other cases it denotes the numerical precision. Classical algorithms return the whole solution vector. Quantum algorithms return a quantum state; in order to extract the classical vector one needs $\mathcal{O}(N)$ copies on the state. **a)** is an approximate algorithm and can be applied to dense matrices; **b)** is exact but it does not output the solution vector and works only for sparse matrices (more details can be found in Section 6); **c)** requires $\mathcal{O}(N^4)$ parallel units and it is numerically unstable due to high sensitivity to rounding errors. Stable algorithms such as Gaussian elimination with pivoting or parallel QR-decomposition require $\mathcal{O}(N)$ time using $\mathcal{O}(N^2)$ computational units [CR86]; **d)** is an approximate algorithm which returns a rank k -approximation with probability $1 - \delta$, and has an additional error $\epsilon \|A\|_F$. Exact methods for an $N \times M$ matrix scale with $\min\{MN^2, NM^2\}$; **e)** calculates SVD by computing the eigenvalue decomposition of the symmetric matrix AA^T ; **f)** works on dense matrices that are low-rank approximable.

Table 37 (Ciliberto, Herbster, Ialongo, Pontil, Rocchetto, Severini, Wossnig, 2018).

Algorithms whose runtime is upper bounded by a polynomial function of N are said to be efficient. Problems for which there exists an efficient algorithm are easy. Conversely, hard problems are those where no polynomial algorithm is known.

The quantum algorithms surveyed here speed up efficient classical algorithms. Two types of speedups are obtained: polynomial or exponential. Polynomial speedups, although important from a practical point of view, do not prove that quantum computers are able to turn hard learning problems into easy ones. On the other hand, exponential speedups of algorithms that are already efficient face important challenges.

In order to achieve an exponential speedup despite the computational costs arising from accessing the memory we are restricted to hard algorithms. This is because, for these algorithms, the polynomial time construction of the quantum state that encodes the data set does not dominate over the speedup (Ciliberto, Herbster, Ialongo, Pontil, Rocchetto, Severini, Wossnig, 2018).

How to put classical data in superposition:

- Given vector $v \in \mathbb{R}^d$: how to prepare $|v\rangle = \frac{1}{\|v\|} \sum_{i=1}^d v_i |i\rangle$

- Assume quantum-addressable memory: $O_v : |i, 0\rangle \mapsto |i, v_i\rangle$

1. Find $\mu = \max_i |v_i|$ in $O(\sqrt{d})$ steps

2. $\frac{1}{\sqrt{d}} \sum_i |i\rangle \xrightarrow{O_v} \frac{1}{\sqrt{d}} \sum |i, v_i\rangle \mapsto \frac{1}{\sqrt{d}} \sum |i, v_i\rangle \left(\frac{v_i}{\mu} |0\rangle + \sqrt{1 - \frac{v_i^2}{\mu^2}} |1\rangle \right)$

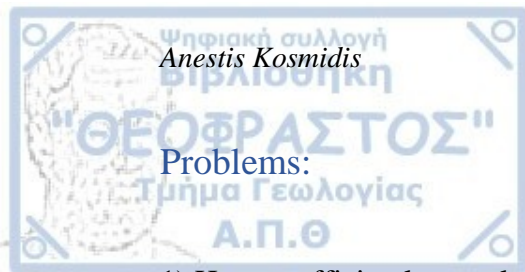
$\xrightarrow{O_v^{-1}} \frac{1}{\sqrt{d}} \sum_i |i\rangle \left(\frac{v_i}{\mu} |0\rangle + \sqrt{1 - \frac{v_i^2}{\mu^2}} |1\rangle \right) = \frac{\|v\|}{\mu \sqrt{d}} |v\rangle |0\rangle + |w\rangle |1\rangle$

3. Boost $|0\rangle$ by $O\left(\frac{\mu \sqrt{d}}{\|v\|}\right)$ rounds of amplitude amplification

-Expensive for “peaked” v ; cheap for “uniform” or “sparse” v (but there we can efficiently compute many things classically!) (Arunachalam, Wolf, 2017).

Many other attempts at using quantum for Machine Learning (ML):

- 1) k-means clustering
- 2) Support Vector Machines
- 3) Training perceptrons (depth-1 neural networks)
- 4) Quantum deep learning (=deep neural networks)
- 5) Training Boltzmann machines for sampling (Arunachalam, Wolf, 2017)



Problems:

- 1) How to efficiently put classical data in superposition?
- 2) How to use reasonable assumptions about the data (also in classical ML)
- 3) We don't have a large quantum computer yet!

- How to measure the efficiency of the learning algorithm:

- (i) Sample complexity: number of examples used
 - (ii) Time complexity: number of time-steps used
- A good learner has small time and sample complexity.

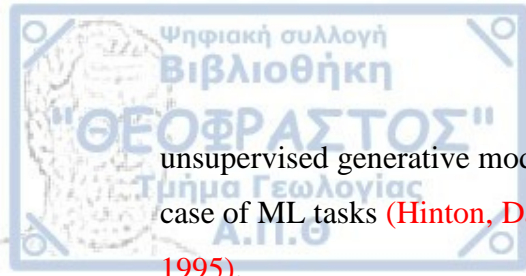
Quantum data:

We try to circumvent the problem of putting classical data in superposition, by assuming we start from quantum data: one or more copies of some quantum state, generated by natural process or experiment. However, it is observed that in distribution-independent learning, quantum examples are not significantly better than classical examples.

We can get quadratic speed-ups for some ML problems, while exponential speed-ups are under strong assumptions. The biggest issue is how to put big classical data in superposition. So in some scenarios, provably there is no quantum improvement (Arunachalam, Wolf, 2017).

Quantum Assisted Machine Learning:

We introduce the quantum-assisted Helmholtz machine (QAHM), an attempt to use near-term quantum devices to tackle high-dimensional data sets of continuous variables. Instead of using quantum computers to assist deep learning, the QAHM uses deep learning to extract a low-dimensional binary representation of data, suitable for relatively small quantum processors which can assist the training of an



unsupervised generative model. Figure 2 illustrates an example of this concept for the case of ML tasks (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

Research in the field of Quantum Assisted Machine Learning (QAML) has been focusing on tasks such as classification, regression, Gaussian models, vector quantization and principal component analysis. We do not think these approaches would be of practical use in near-term quantum computers. The same reasons that make these techniques so popular, e.g., their scalability and algorithmic efficiency in tackling huge data sets, make them less appealing to become top candidates as killer applications in QAML with devices in the range of 100–1000 qubits. In other words, regardless of the claims about polynomial and even exponential algorithmic speed-up, reaching interesting industrial-scale applications would require millions or even billions of qubits. Such an advantage is then moot when dealing with real-world data sets and with the quantum devices to become available in the next years in the few thousands-of-qubits regime. So, we believe that only a game changer such as the new developments in hybrid classical quantum algorithms might be able to make a dent in speeding up ML tasks (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

We propose and emphasize the following approach to maximize the possibility of finding killer applications on near-term quantum computers. More specifically, we focus on data sets with potentially intrinsic quantum-like correlations, making quantum computers indispensable. These will provide the most compact and efficient model representation, with the potential of a significant quantum advantage even at the level of 50–100 qubit devices. It is suggested the case of the cognitive sciences, as a research domain potentially yielding such data sets (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

However, the small number of qubits and the limitations of currently available hardware may impair the sampling process, making it useless for real ML applications. So, we argue that even noisy distributions could be used for generative modeling of real-life data sets. This requires working in settings where the operations implemented in hardware are only partially known. We call this scenario a gray-box. We also argue that hybrid classical-quantum architectures are suitable for near-term

applications where the classical part is used to bypass some of the limitations of the quantum hardware. We call this approach quantum-assisted (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

We wonder which type of real-life applications could benefit from quantum supremacy with near-term small devices. One of the main motivations underlying the research efforts described here is that quantum computers could speed up ML algorithms. This suggests that quantum models hold the potential to substantially reduce the amount of other type of computational resources, e.g., memory required to model a given data set.

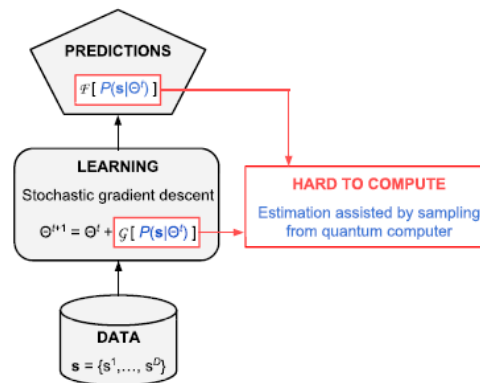
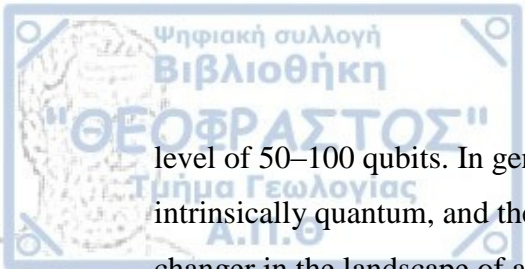


Figure 11: General scheme for hybrid quantum-classical algorithms as one of the most promising research directions to demonstrate quantum enhancement in ML tasks. A data set drives the fine tuning of model's parameters. In the case of generative models one can use stochastic gradient descent to update the parameters Θ from time t to $t+1$. The updates often require estimation of an intractable function G , which could be approximated by samples from a probability distribution $P(s|\Theta^t)$. This computationally hard sampling step could be assisted by a quantum computer. In some cases, making predictions out of the trained model is also an intractable task. The predictions F could be approximated by samples with the assistance of a quantum computer as well (Pedromo-Ortiz, Benedetti, Realpe-Gomez, Biswas, 2018).

Although it is emphasized the case of cognitive sciences, it would be interesting to explore what other relevant and commercial data sets exhibit quantum-like correlations, and where quantum computers can have an advantage even at the



level of 50–100 qubits. In general, the identification of characteristics that are intrinsically quantum, and therefore hard to simulate classically, could be a game changer in the landscape of applications for near-term quantum technologies.

Challenges in QAML:

We distinguish between two types of algorithms: those that operate on quantum data (i.e. data that is output of a quantum process, for example, a quantum chemistry problem) and those that seek to process data stored in a classical memory. The first case is ideal for QML. The data is ready to be analyzed and we do not have to spend computational resources to convert the data into quantum form. The second case is more elaborate as it requires a procedure that encodes the classical information into a quantum state. We know that the computational cost of this operation is particularly relevant to determine whether we can obtain quantum speedups in machine learning for classical data (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

We assume that we want to process N d -dimensional classical vectors with a quantum algorithm. The quantum random access memory (QRAM) is a quantum device that can encode in superposition N d -dimensional vectors into $\log(N d)$ qubits in $O(\log(N d))$ time by making use of the so called “bucket-brigade” architecture. The idea is to use a tree-structure where the $N d$ leaves contain the entries of the N vectors in \mathbb{R}^d . The QRAM, with a runtime of $O(\log(N d))$, can return a classical vector in quantum superposition efficiently. However, the number of physical resources it requires scales as $O(N d)$. This exponential scaling, with respect to the number of qubits, has been used to question whether the QRAM can be built in an experimental setting or whether it can provide a genuine computational advantage. Fundamentally the issue can be related to whether the exponential number of components needs to be continuously “active”. The proponents of the QRAM claim that only $O(\log(N d))$ components need to be active while the others can be considered as “non-active” and error free. Whether this assumption holds in an experimental setting is unclear (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

The first challenge that appears with QRAM is whether all the components require to be error corrected. Indeed, if the exponential physical resources required full error correction then it would be impractical to build the device in an experimental setting. We consider that for superpolynomial query algorithms, the QRAM requires error-corrected components.

A second problem is related to the exponential number of resources in an active memory. More specifically, the only fair comparison of a system which requires an exponential number of resources, is with a parallel architecture with a similar amount of processors. In this case many linear algebra routines, including solving linear systems and singular value decomposition, can be solved in logarithmic time.

A third challenge of the QRAM is the requirement of having data distributed in a relatively uniform manner over the quantum register. If that was not the case, the QRAM would violate the search lower bounds. In the case of non-uniformly distributed data, the QRAM is no longer efficient and take $O(\sqrt{N})$ to turn the classical data set into quantum superposition.

A fourth comment on the QRAM, the possibility of loading the data in logarithmic time, when the size of the data is considerable, can be controversial due to speed of communication arguments. We point out that latency can play a role in big memory structures. In particular, a lower bound on the distance which the information has to travel implies a lower bound on latency, due to considerations on the limits set by the speed of light. In a three dimensional space these are given by $O(\sqrt[3]{Nd})$. In practice these considerations will only dominate if the amount of memory is extremely large but, because in quantum machine learning we aim at data sets that surpass the current capability of classical computers, this bound is a potential caveat (Hinton, Dayan, Frey, Neal, 1995, Dayan, Hinton, Neal, Zemel, 1995).

There are additional challenges which will generally impact any QAML algorithm, such as the limited qubit connectivity, the finite dynamic range of the parameters dictated by the intrinsic energy scale of the interactions in the device, and intrinsic noise in the device leading to decoherence in the qubits and uncertainty in the programmable parameters (Pedromo-Ortiz, Benedetti, Realpe-Gomez, Biswas, 2018).

We conclude that the QRAM allows to upload data efficiently but might be hard to implement experimentally or might not allow a genuine quantum advantage if we take into account all the required resources. Noticeably the fast data access guaranteed by the QRAM is only required for QLM algorithm that run in sublinear time. Although many known QML algorithms run in sublinear time, quantum learning theory suggests that for some classically hard problems quantum resources might give exponential advantages. In this case, a memory structure that can prepare a quantum superposition in polynomial time (i.e. in $O(Nd)$) can still be sufficient to maintain a quantum speedup compared to the classical runtime.

One key strategy we propose towards the near-term demonstration of quantum advantage is the development of hybrid quantum–classical algorithms capable of exploiting the best of both worlds. Therefore, we put forward a new framework for such hybrid QAML algorithms (Pedromo-Ortiz, Benedetti, Realpe-Gomez, Biswas, 2018).

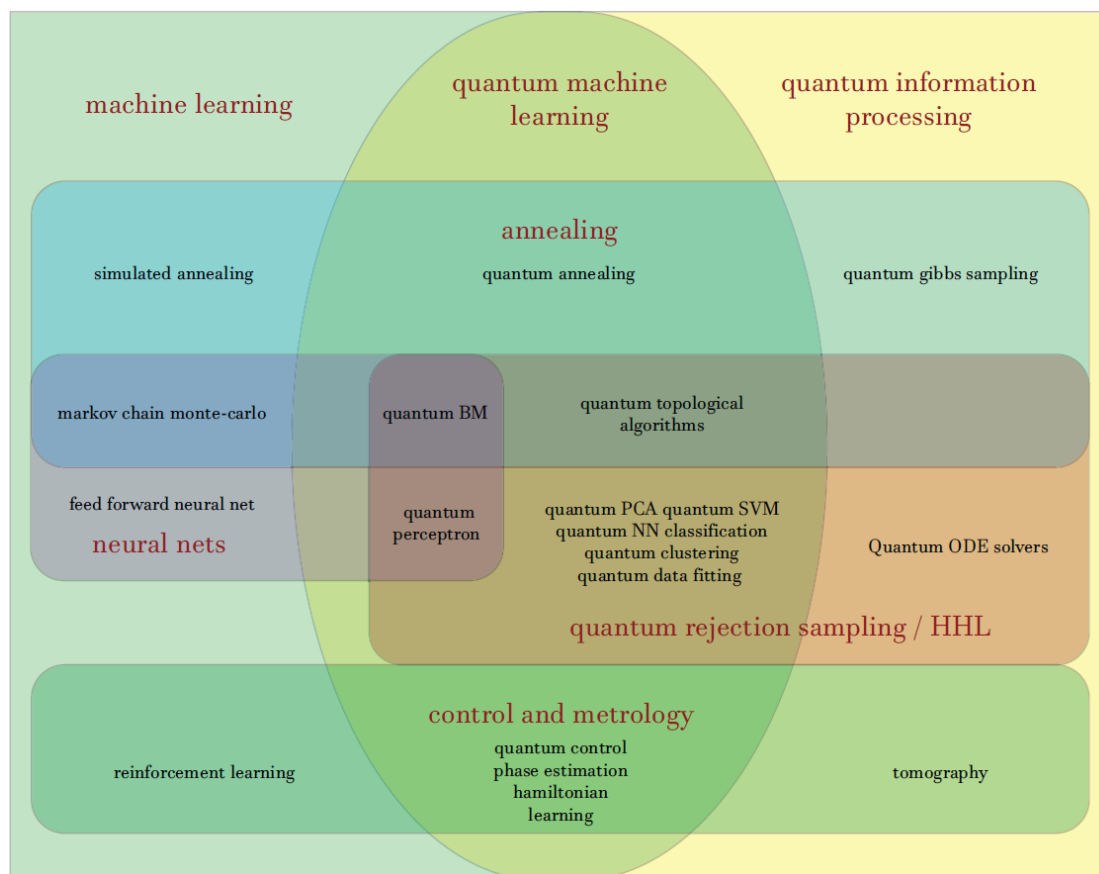


Figure 12: Quantum Machine Learning (Sekar, 2017).



Anestis Kosmidis



CHAPTER 4: QUANTUM CLUSTERING

Quantum Computing Shor's Algorithm and Hierarchical Clustering Technique:

It is observed that the detection procedure of Cancer Disease is very much time consuming and the results obtain by them are not so fast, so there is a need of more accurate, fast and efficient method through computing technologies. This can be accomplished with the collaboration between Quantum computing and the clustering algorithm, i.e. Shor Algorithm of Quantum Computing with various Hierarchical Clustering Technique. More specifically, the Hierarchical Clustering Technique helps in clustering of results to obtain an approach for Cancer Disease Detection, while the Shor Algorithm helps to increase the efficiency in term of accuracy (Jain, Chaturvedi, 2014).

We use quantum switching architecture for nearest neighbour coupling, namely an efficient quantum shear sorting (QSS) algorithm to reduce the number of time steps. For the QSS algorithm, the running complexity of the quantum switching architecture is polynomial in time with the nearest neighbour coupling and the implementation is less complex. The result shows that improved switching is extremely simple to implement using existing quantum computer candidates. The Quantum Computing technique can provide faster and efficient results with the use of different parameter in this system and the K-means clustering technique and Agglomerative clustering technique can provide a result analysis by clustering of results (Jain, Chaturvedi, 2014).

The quantum search algorithm is a technique for searching possibilities in only steps. In the first step we collect the information in the form of tumor size and node status as Data set, which has to be analyzed by both conventional pathological tests & our Quantum computing based technique which involves Shor Algorithm as an analyzing tool to analyze the data set. In the next step the clustering of obtained result by both techniques is done. In case of analyses of data set with conventional pathological tests, these tests analyze different aspects one by one in stages i.e. after completion of 1st they proceed towards 2nd and so on, whereas on the other hand in our Quantum computing based Analyses, Shor Algorithm analyzes the data set

peculiarly on the basis of range of parameters so there is no such time consumption, namely if the parameters are not in the range of define values, the algorithm automatically analyzes on next level and gives results accordingly. Then treatment will start within a short duration with more accuracy according to cancer type and stage of severity, because after getting results from Shor Algorithm and clustering an oncologist can get a clear cut idea about the stage of Cancer in which patient held (Figure 13) (Jain, Chaturvedi, 2014).

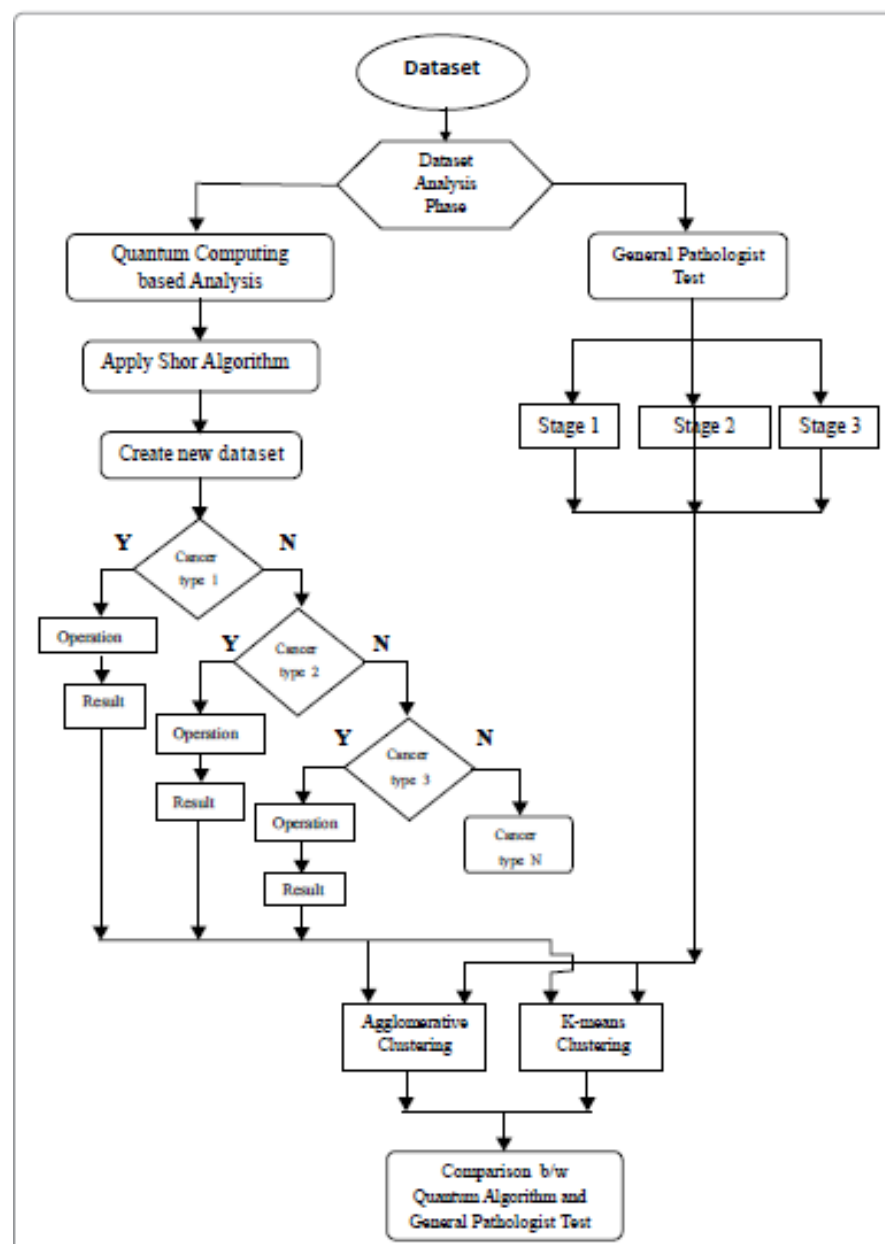


Figure 13: Designed System for Cancer Disease Detection (Jain, Chaturvedi, 2014).



Basically six steps are used to create a new data set which are given below:

Step 1: Apply the Shor Algorithm on data set providing tumor size denoted by 'A', Metastasis denoted by 'MS' and Node status 'NS'.

Step 2: Apply the number 'N' is the number wish to factorize. As the different stages of Cancer depending upon severity are here and the remainder from this operation is placed in a second 3 bit register.

Step 3: Apply random number X, where $1 < X < N-1$. This is the last stage of providing the initials to execute the Algorithm and there are chances of multiple answers so to reduce the error in operation these to be verified with different value of random number 'X'.

Step 4: Results provided from Shor Algorithm. The result which we get is in the form of stages of Cancer which are now ready for clustering by using statistical techniques.

Step 5: Clustering by K-means Clustering Technique and Agglomerative clustering technique, here top to bottom approach used. Clustered values are used to draw Dendrogram and graphical representations.


Step 6: Comparing the clustered values of results obtained by both the techniques i.e. from conventional Cancer Disease Detection and Shor Algorithm based Disease Detection technique (Jain, Chaturvedi, 2014).

Experiments performed on data set of cancer: In this Data set which have collected data for Cancer Disease which contain the Tumor Size, Node Status, Metastasis (Collected from Navodaya Cancer Hospital, Indrapuri, Bhopal). These three parameters have a great importance in Cancer Disease detection as Tumor size in any type of Cancer is the first Criteria for further analyses then Metastasis i.e. the spread of a cancer from one organ or part to another non-adjacent organ or part and Node status i.e. the lymph node condition at the site of tumours which either be negative or positive (Table 38). Here A is the Tumor size, ID is Identification number, N0 is clear or Negative node, N1 is Cancerous or positive node, M0 is No

spread of Tumor, M1 is Tumor has spread, Weight in Kg and Age in years (Jain, Chaturvedi, 2014).

Experiments performed on proposed data set of cancer: The given below variables are used for applying the Shor Algorithm on data set (Table 39). Here A is the tumor size, X is the random number, N is the number we wish to factorized, M denote the metastasis, M0=no spread of tumor, M1=tumor has spread, NS denote the node status, N0=clear or negative nodes, N1=cancerous or positive node.

Shor based algorithm shows the stages of cancer, Patient age in years and ID shows the identification of patients. First evaluate the data set on the basis of Tumor Size then after applying Shor Algorithm get different Stages according to Tumor Size and Node status and Metastasis condition. According to Node Status and Metastasis Conditions sometime a patient who having a large Tumor may also do not have any Cancer Disease because of having N0 which means negative node or M0 condition which means Tumor is not malignant or not spreading which shows the Tumor is here but not carcinogenic (Jain, Chaturvedi, 2014).



S.no	Name	ID	A	NS	MS	Age	Weight
1	Ankya	1	2	N0	M0	18	45
2	Sujit	5	2	N0 or N1	M0	20	78
3	Dinesh	6	3	N0 or N1	M0	24	68
4	Ruchi	9	7	N1	M1	28	55
5	Kiran	11	1	N0	M0	49	62
6	Dimple	15	3	N0 or N1	M0	51	56
7	Rohan	21	4	N0 or N1	M1	44	72
8	Pradeep	26	4	N0 or N1	M0	52	69
9	Servesesh	31	1	N0	M0	40	73
10	Vinay	35	6	N0	M0	53	75
11	Ashish	39	4	N0 or N1	M1	67	70
12	Shanti	41	1	N0	M0	30	64
13	Upendra	45	3	N0 or N1	M0	26	67
14	Kapil	2	3	N0 or N1	M0	44	70
15	Sanjeev	12	4	N0	M0	73	55
16	Sanchita	14	2	N0 or N1	M1	81	52
17	Shobhit	24	3	N0 or N1	M0	54	48
18	Nirupma	34	2	N0 or N1	M0	31	53

Table 38: Data set of Cancer Disease

S.no	Name	ID	A	X	N	STAGE	NS	MS
1	Ankya	1	2	2	3	1	N0	M0
2	Sujit	5	2	4	7	2	N0 or N1	M0
3	Dinesh	6	3	2	5	3	N0 or N1	M0
4	Ruchi	9	7	6	7	4	N1	M1
5	Kiran	11	1	2	3	0	N0	M0
6	Dimple	15	3	3	5	2	N0 or N1	M0
7	Rohan	21	4	3	7	4	N0 or N1	M1
8	Pradeep	26	4	3	6	3	N0 or N1	M0
9	Servesesh	31	1	1	3	0	N0	M0
10	Vinay	35	6	4	7	1	N0	M0
11	Ashish	39	4	2	6	4	N0 or N1	M1
12	Shanti	41	1	3	4	0	N0	M0
13	Upendra	45	3	2	5	3	N0 or N1	M0
14	Kapil	2	3	3	5	2	N0 or N1	M0
15	Sanjeev	12	4	2	5	1	N0	M0
16	Sanchita	14	2	4	6	4	N0 or N1	M1
17	Shobhit	24	3	2	5	3	N0 or N1	M0
18	Nirupma	34	2	3	7	2	N0 or N1	M0

Table 39: Results obtained from

Shor Algorithm based Cancer Disease

Detection Technique

(Jain, Chaturvedi, 2014).

Results comparison between conventional cancer disease detection technique and shor algorithm based cancer disease detection technique on the basis of dendrogram (Figures 14, 15, 16). By these two profiles search it is easy to understand that there is Shor Algorithm which is a basic Algorithm of Quantum Computing is giving results convenient for an Oncologist and make an exact figure of a patient's condition who suffers from the most devastating disease of this century i.e. Cancer (Jain, Chaturvedi, 2014).

We propose the use of tumour size as a parameter for disease prediction with metastasis condition and node status is used so accuracy is also enhanced in compare to any technique which uses only tumour size as a parameter. So the chances of error in disease detection increases because in many cases this is observed that a patient having a large tumour size have not been suffering from cancer, as the node status is not shows symptoms of Cancer (N0) also the possibility of Metastasis condition where the tumour is not showing any malignancy (M0) but in case of Quantum Computing based Approach i.e. Shor Algorithm based Cancer Disease Detection

Technique these errors has been removed for increasing accuracy (Jain, Chaturvedi, 2014).

The different Hierarchical Clustering helps to understand the results by making clusters of acquired data obtained by both the Detection Techniques. Finally with the help of these three Distance measures we get difference among the conventional approach and Quantum Computing based approach. Where Conventional approach which is used widely for Cancer Disease Detection-based on Classical Computing, on the other hand this Shor Algorithm based Cancer Disease Detection technique is totally based on Quantum Computing so an Oncologist get the results more frequently that is within a few seconds in compare to hours in case of Classical Computing (Jain, Chaturvedi, 2014).

Also the result are more accurate and accessibility to different parameters is more in case of Quantum Computing based approach so it is much easy for an oncologist to create a treatment program with more ease and accuracy within a short period of time which may be act as a boon for the Cancer patients in near future (Jain, Chaturvedi, 2014).

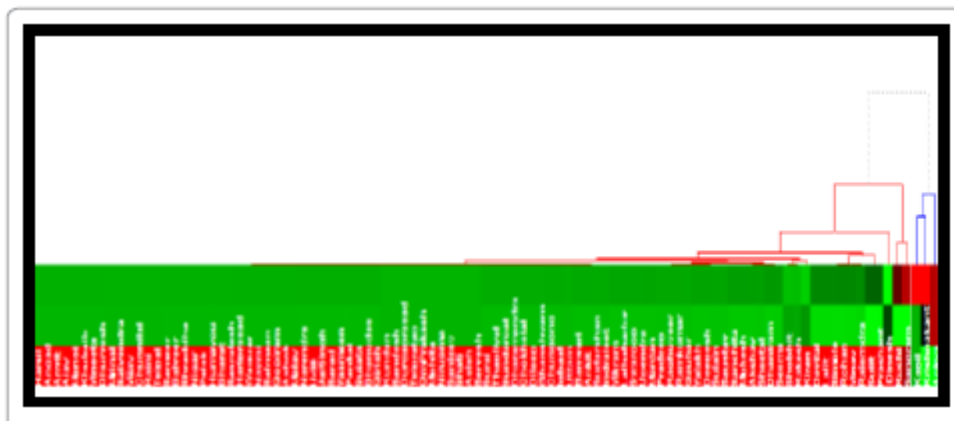


Figure 14: Dendrogram of Conventional Cancer disease detection Technique using Euclidean Distance (Jain, Chaturvedi, 2014).

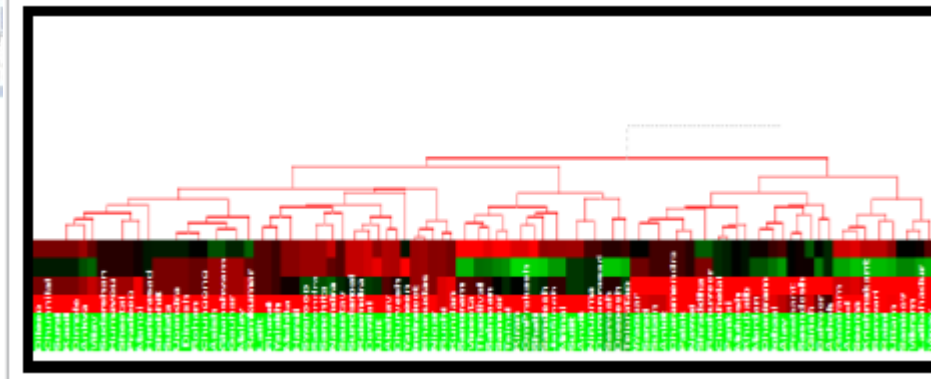


Figure 15: Dendrogram of Shor Algorithm based Cancer Disease Detection Technique using Euclidean Distance (Jain, Chaturvedi, 2014).

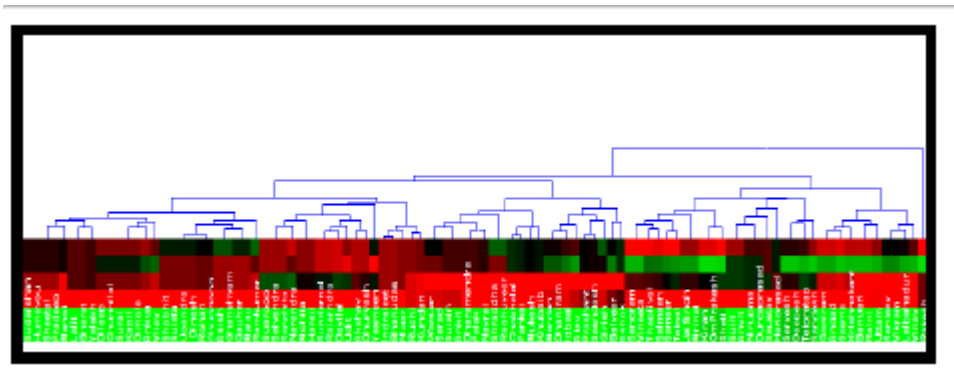


Figure 16: Dendrogram of Shor Algorithm based Cancer Disease Detection Technique using Euclidean Distance (Jain, Chaturvedi, 2014).

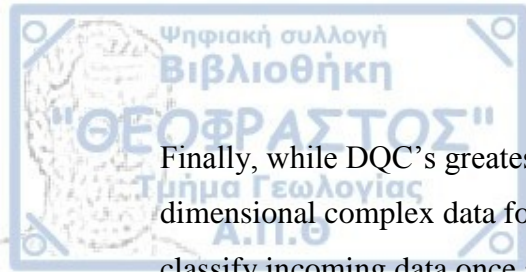
Dynamic Quantum Clustering:

We provide the following question: “How does one search for a needle in a multi-dimensional haystack without knowing what a needle is and without knowing if there is one in the haystack itself?”. The answer in this question is Dynamic Quantum Clustering (DQC). DQC is a powerful visual method that works with big, high-dimensional data. It exploits variations of the density of the data in feature space and unearths subsets of the data that exhibit correlations among all the measured variables. The outcome of a DQC analysis is a movie that shows how and why sets of data-points are eventually classified as members of simple clusters or as members of -

what we call - extended structures. This allows DQC to be successfully used in a non-conventional exploratory mode where one searches data for unexpected information without the need to model the data. The DQC methodology works for big, complex, real-world data sets that come from five distinct fields: i.e., x-ray nano-chemistry, condensed matter, biology, seismology and finance. We know that big, complex data sets often contain interesting structures that will be missed by many conventional clustering techniques. Experience shows that these structures appear frequently enough that it is crucial to know they can exist, and that when they do, they encode important hidden information. However, DQC is able to detect these structures. In short, we not only demonstrate that DQC can be flexibly applied to data sets that present significantly different challenges, we also show how a simple analysis can be used to look for the needle in the haystack, determine what it is, and find what this means (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

We note DQC is not only a density based clustering algorithm, but it is also a visual tool that can reveal subsets of large, complex data that exhibit simultaneous correlations among the many variables being measured. More specifically, a DQC analysis begins with the creation of a movie wherein proxies of the data-points move from their initial position towards the nearest region of higher density. Hereafter this step will be referred to as the DQC evolution of the data. Correlated subsets are distinguished from one another depending on their final shape during or after DQC evolution: extended shapes are referred to as structures, while the term cluster is reserved for subsets that collapse to a point. A DQC analysis results in a movie that visually reveals how and why the algorithm identifies and distinguishes between structures and clusters (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

We present some of the advantages of DQC algorithm. Firstly, DQC doesn't begin by assuming there are structures to be found, and because it has been proven not to find structures in random data and it makes no assumptions about the type or shape (topology) of structures that might be hidden in the data, it can be used to determine if one is collecting the right kind of information. Secondly, DQC exploits variation in the density of the data. Thus, it reveals structures with unusual topologies even in very dense data sets. Furthermore, DQC works well for high-dimensional data since the time spent in a DQC analysis only grows linearly with the dimension of the data.



Finally, while DQC's greatest strength is that it allows one to visually explore high-dimensional complex data for unexpected structure, it can also be used to rapidly classify incoming data once a sufficiently large subset of data has been analyzed (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

The steps of the DQC algorithm are presented in Appendices (Appendix A).

We support that the output DQC evolution is an animation showing how data points move towards the nearest minimum of the potential. More specifically, if the potential has isolated minima due to topologically simple regions of higher density, then the results of the evolution are fixed points describing isolated clusters. If, however, there are higher density regions of the data where the density is constant along complicated and possibly intersecting shapes, then the results of DQC evolution will be filamentary structure (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

This is what one will see if there are subsets of the data that exhibit multivariate correlations that can be parameterized in terms of only a few variables. We can show that these structures encode important information about the data (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

The Galaxies example:

We want to demonstrate that the DQC potential accurately captures the density of data-points, and that DQC evolution can reveal extended, topologically non-trivial structures (or regions of nearly constant density) hidden in the data. So, we apply it to a well understood subset of 139,798 galaxies taken from the Sloan Digital Sky Survey (SDSS). Each data entry consists of the three coordinates of a single galaxy. The first two numbers are θ and ϕ , the angular coordinates defined in our Galaxy; the third coordinate is the redshift, z , a proxy for the distance from us to the other galaxies. It is well known that galaxies are not uniformly distributed, but rather they form a web of filaments and voids, so the question is if DQC evolution will reveal this structure (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

We observe that [Figure 17A](#) is a plot of the quantum potential for a two-dimensional subset of the data obtained by choosing galaxies whose red-shifts differ by a very small amount. The galaxies are plotted as yellow points and the transparent quantum potential constructed from this set of galaxies is plotted upside-down, so that the maxima of the upside-down potential actually correspond to minima. This plot shows that the potential closely conforms to the distribution of galaxies and so it is clearly a very good proxy for the density of the data. Changing by 20%, doesn't change the potential significantly. We note that this two-dimensional slice of the data shows significant structure, but fails to exhibit filamentary features of nearly constant density. Furthermore, in [Figures 17B-17D](#) we see what happens to the full three-dimensional data set as DQC evolution collects the data-points into structures that follow the shape of the minima of the three-dimensional potential. In this case DQC evolution reveals the existence of the network of filaments and voids that is not readily apparent in [Figure 17B](#). The web of filaments revealed in this picture correspond to the topological structure of the minima of the quantum potential (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

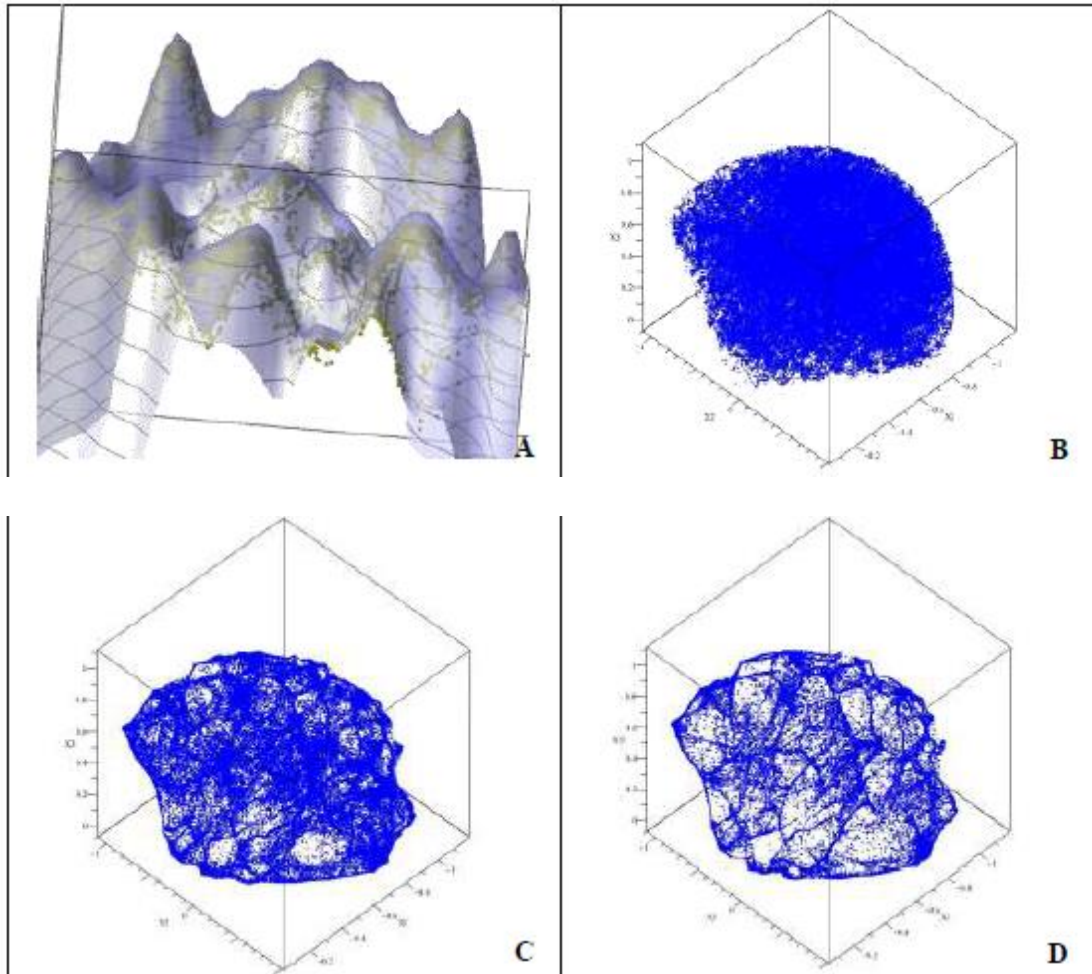


Figure 17: A) Comparison of SDSS data points with the derived DQC potential. The potential is plotted upside down and the yellow data points are slightly shifted in order to increase their visibility. B) The distribution of data in a 3D space defined by θ , ϕ and z . C) Early stage of DQC evolution of the data. D) Further DQC evolution exhibits the clear appearance of string-like structures (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

DQC algorithm can be used in a variety of applications, as it can be applied to all sorts of data. We recommend that possible future extensions demonstrate that DQC can be important to people working in such diverse fields as chemistry, biology, particle physics, astrophysics, genomics, business, finance, analysis of social networks and national security (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

Quantum Meila-Shi Clustering Algorithm vs k-Means Clustering Algorithm:

We know that one of the biggest problems of data analysis is data with no known a priori structure. Therefore, data clustering, which seeks to find internal classes or structures within the data, is one of the most difficult, yet needed implementations. The standard algorithm is K-means, which rests on the following assumptions:

- (1) Assume in advance the number of clusters
- (2) Generate random seeds
- (3) Assume at least one seed “hits” every cluster
- (4) Clusters “grow” in the neighborhood of each seed
- (5) Cluster regions grow until saturation (Scott, Therani, Wang, 2017).

On the other hand, Meila-Shi algorithm supposes we have a real $N \times K$ data matrix Q and then:

$$S = Q \times Q^T \quad (4.3)$$

$$A = a_{i,j} = \frac{s_{i,j}}{\sqrt{\sum_{k=1} s_{i,k} \sum_{k=1} s_{j,k}}} \quad (4.4)$$

$$P = p_{i,j} = \frac{s_{i,j}}{\sum_{k=1} s_{i,k}} \quad (4.5)$$

where S is the similarity matrix, A is the adjacency matrix and P is a row-stochastic matrix, often called a Markov matrix. It is also called a transition matrix. Moreover, φ_i and ψ_i are respectively the normalized eigenvectors of A and P , taken as column vectors, but these matrices share the same eigenvalues λ_i , which have special properties:

$$\lambda_0 = 1 \quad \text{and} \quad \lambda_{i+1} < \lambda_i < \lambda_0, \quad \text{where } i = 1, 2, 3, \dots$$

However, the corresponding eigenvectors of P , i.e. ψ_i provide a much better clustering picture. For $i > 0$, plotting the lead eigenvectors ψ_1 versus ψ_2 (often the

leading two eigenvectors $i = 1, 2$ are sufficient) serves as the principal axes, graphically provides a clustering picture and consequently a considerable dimensional and size reduction of the original problem (Meila, Shi, 2001, Scott, Therani, Wang, 2017).

Artificially-created data set of random points data set:

We compose artificially-created random points within two circular envelopes of different sizes with the circle of a smaller size having a high density of points. We observe Figure 18a shows the result for k-Means, as provided by the MATLAB toolbox, for a choice of two clusters. The result is stable, but illustrates one of the problems experienced with overlapping clusters using k-Means. The smaller cluster, which is shown in red, penetrates the larger circle too much. However, as we see in Figure 18b, the contour plot of the quantum potential allows us to better isolate the smaller cluster. We note the cluster centers are shown in black dots. The continuous transitions between potential minima, which are the cluster centers, provide a continuous description of the “fuzzy-clustering” aspects (Scott, Therani, Wang, 2017).

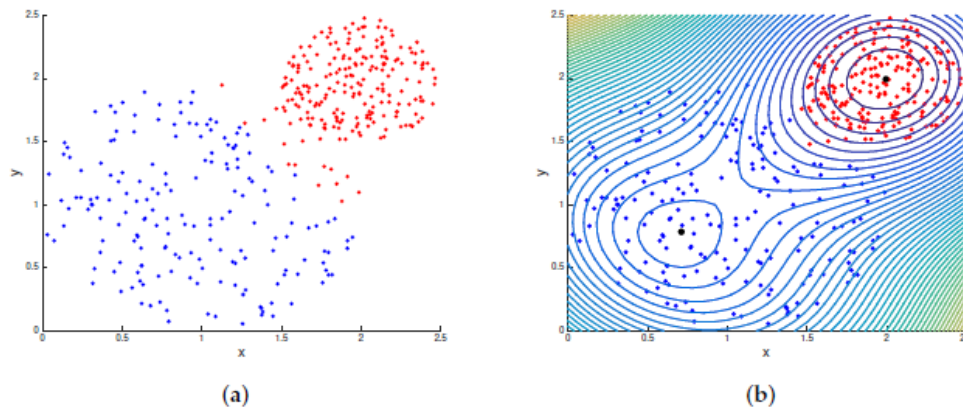


Figure 18: k-Means vs quantum clustering on two overlapping circles. (a) k-Means clustering: the smaller circle “overflows” into larger blue circle. (b) Quantum clustering: the contour plot better isolates the smaller circle, $\sigma = 0.7125$ (Scott, Therani, Wang, 2017).

This example comes from biology and we suppose one has two sexes and two (new) species and consequently four groups. Preserved specimens lost their colour, so it was hoped that morphological differences would enable museum material to be classified. Data was collected on 50 specimens of each sex of each species, collected in Western Australia. Each specimen had measurements to according to: (1) frontal lip, (2) rear width, (3) length along midline, (4) maximum width of carapace and (5) body depth. Thus,, the total data set is a 200 x 5 data matrix.

We observe that [Figure 19](#) shows the outcome of the application of spectral Meila-Shi and quantum clustering on this data. The actual classes are illustrated by the colours red, blue, green and yellow. The lead eigenvectors ψ_1 and ψ_2 are sufficient to provide a complete two-dimensional clustering picture. It is also shown the contour plot from the quantum clustering potential is the minima clearly indicating the cluster centers. All four classes were recovered to within 80% of the data according to the Jaccard index.

The “fuzzy” nature of points that are nearly equally spaced between cluster centers is handled continuously by the quantum potential. Our results are comparable, but the difference in outcome between the two approaches increases for larger data sets in both row and column size ([Ripley, 1996](#), [Horn, Gottlieb, 2001](#), [Scott, Therani, Wang, 2017](#)).

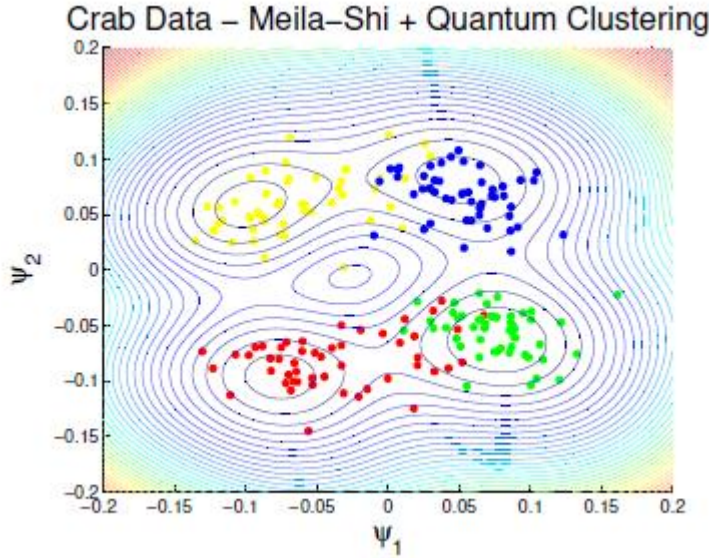


Figure 19: Identification of four classes by quantum clustering (Scott, Therani, Wang, 2017).

Quantum Clustering Algorithms – Results visually:

A given set of data-points in some feature space may be associated with a Schrödinger equation whose potential is determined by the data. This is known to lead to good clustering solutions. We extend this approach into a full-fledged dynamical scheme using a time-dependent Schrödinger equation. Moreover, we approximate this Hamiltonian formalism by a truncated calculation within a set of Gaussian wave functions (coherent states) centered around the original points. This allows for analytic evaluation of the time evolution of all such states, opening up the possibility of exploration of relationships among data-points through observation of varying dynamical-distances among points and convergence of points into clusters. This formalism may be further supplemented by preprocessing, such as dimensional reduction through singular value decomposition or feature filtering (Weinstein, Marvin, Horn, 2009).

We advocate the use of a Schrödinger Hamiltonian H that is intimately connected to the data structure, as defined by the quantum clustering method and summarized below. We extend it into a time-dependent Schrödinger equation:

$$i \frac{\partial \psi(\vec{x}, t)}{\partial t} = H \psi(\vec{x}, t) \quad (4.6)$$

The ensuing Dynamic Quantum Clustering (DQC) formalism allows us, by varying a few parameters, to study in detail the temporal evolution of wave-functions representing the original data-points. Then, this dynamical behavior allows us to explore the structure of the quantum potential function defined by the quantum clustering method. DQC begins by associating each data-point with a state in Hilbert space. The temporal development of the centroids of these states may be viewed in the original data-space as moving images of the original points. Their distances to one-another change with time, thus representing associations they form with each other. Convergence of many points onto a common center at some instant of time is an obvious manifestation of clustering. Many transitional relationships may occur, revealing substructures in clusters or even more complex associations. For this reason we propose this approach as a general method for visually and interactively searching for and exploring structures in sets of data (Weinstein, Marvin, Horn, 2009).

More specifically, we start to describe the Quantum Clustering method. The quantum clustering approach begins by associating to each of n data points \vec{x}_i in an Euclidean space of d dimensions a Gaussian wave-function $\psi_i(\vec{x}) = e^{-\frac{(\vec{x}-\vec{x}_i)^2}{2\sigma^2}}$ and then constructing the sum of all these Gaussians:

$$\psi(\vec{x}) = \sum_i e^{-\frac{(\vec{x}-\vec{x}_i)^2}{2\sigma^2}} \quad (4.7)$$

Conventional scale-space clustering views this function as a probability distribution (up to an overall factor) that could have generated the observed points, and regards therefore its maxima as determining locations of cluster centers. Often these maxima are not very prominent and, in order to uncover more of them, one has to reduce σ down to low values where the number and location of the maxima depend sensitively upon the choice of σ . Quantum clustering took a different approach, requiring ψ to be the ground-state of the Hamiltonian:

$$H\psi \equiv \left(-\frac{\sigma^2}{2} \nabla^2 + V(x) \right) \psi = E_0 \psi \quad (4.8)$$

By positing this requirement, the potential function $V(x)$ has become inextricably bound to the system of datapoints, since $V(\vec{x})$ is determined, up to a constant, by a simple algebraic inversion of Equation (4.8). Moreover, we may expect V to have minima in regions, where ψ has maxima. In fact, it frequently turns out

that a concentration of data points will lead to a local minimum in V , even if ψ does not display a local maximum. Thus, by replacing the problem of finding maxima of the Parzen estimator by the problem of locating the minima of the associated potential $V(\vec{x})$, we simplify the process of identifying clusters. The effectiveness of this approach has been demonstrated in the work by Horn and Gottlieb. It should be noted that the enhancement of features obtained by applying Equation (4.8) comes from the interplay of two effects: attraction of the wave-function to the minima of V and spreading of the wave-function due to the second derivative (kinetic term) (Weinstein, Marvin, Horn, 2009).

Dynamic Quantum Clustering (DQC) drops the probabilistic interpretation of ψ and replaces it by that of a probability-amplitude, as customary in Quantum Mechanics. DQC is set up to associate data-points with cluster centers in a natural fashion. Whereas in Quantum Clustering this association was done by finding their loci on the slopes of V , here we follow the quantum-mechanical temporal evolution of states associated with these points. Specifically, we will view each data-point as the expectation value of the position operator in a Gaussian wave-function: $\psi_i(\vec{x}) = e^{-\frac{(\vec{x} - \vec{x}_i)^2}{2\sigma^2}}$. The temporal development of this state traces the association of the data-point it represents with the minima of $V(\vec{x})$ and thus, with the other data-points (Weinstein, Marvin, Horn, 2009).

We present the detailed description of the Dynamic Quantum Clustering method as provided by Horn, Weinstein and Marvin in Appendices (Appendix B).

We want to test our method, so we apply it to a five-dimensional data set with two-hundred entries, used in Ripley's text book. This data set records five measurements made on male and female crabs that belong to two different species. The data is stored in a matrix M which has 200 rows and 5 columns.

In what follows we study the temporal behavior of the curves $\langle \vec{x}_i(t) \rangle$, for all i . Henceforth we will refer to this as the "motion of points". Figure 20 shows the distribution of the original data points plotted on the unit sphere in three dimensions. This is the configuration before we begin the dynamic quantum evolution. To visually display the quality of the separation we have colored the data according to its known four classes, however this information is not incorporated into our unsupervised

method. To begin with, we see that the two species of crabs ((red,blue) and (orange,green)) are fairly well separated; however, separating the sexes in each species is problematic. The middle plot in Figure 20 shows the distribution of the points after a single stage of quantum evolution, stopped at a time when points first cross one another and some convergence into clusters has occurred (Weinstein, Marvin, Horn, 2009).

It is immediately obvious that the quantum evolution has enhanced the clustering and made it trivial to separate clusters by eye. Once separation is accomplished, extracting the clusters can be performed by eye from the plots or by any conventional technique, e.g. k-means.

An alternative way of displaying convergence is shown in Figure 21, where we plot the Euclidean distance from the first point in the data set to each of the other points. The clusters lie in bands which have approximately the same distance from the first point. It is difficult to get very tight clusters since the points, while moving toward cluster centers, oscillate around them, and arrive at the minima at slightly different times. Given this intuition, it is clear that one way to tighten up the pattern is to stop DQC evolution at a point where the clusters become distinct, and then restart it with the new configuration, but with the points redefined at rest. We refer to this as iterating the DQC evolution. The right-hand plots in Figure 20 and Figure 21 show what happens when we do this. The second stage of evolution clearly tightens up the clusters significantly, as was expected (Weinstein, Marvin, Horn, 2009).

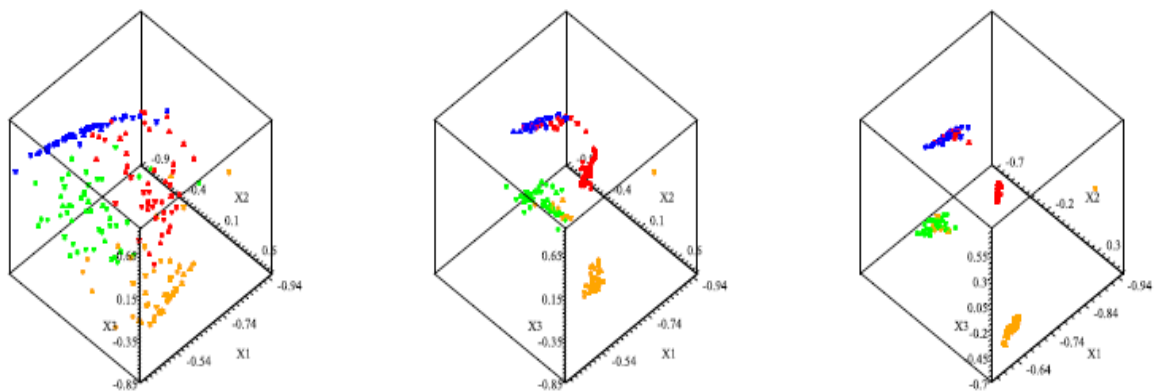


Figure 20: The left hand plot shows three-dimensional distribution of the original data points before quantum evolution. The middle plot shows the same distribution after quantum

evolution. The right hand plot shows the results of an additional iteration of DQC. The values of parameters used to construct the Hamiltonian and evolution operator are: $\sigma = 0.07$ and $m = 0.2$. Colors indicate the expert classification of data into four classes, unknown to the clustering algorithm. Note, small modifications of the parameters lead to the same results (Weinstein, Marvin, Horn, 2009).

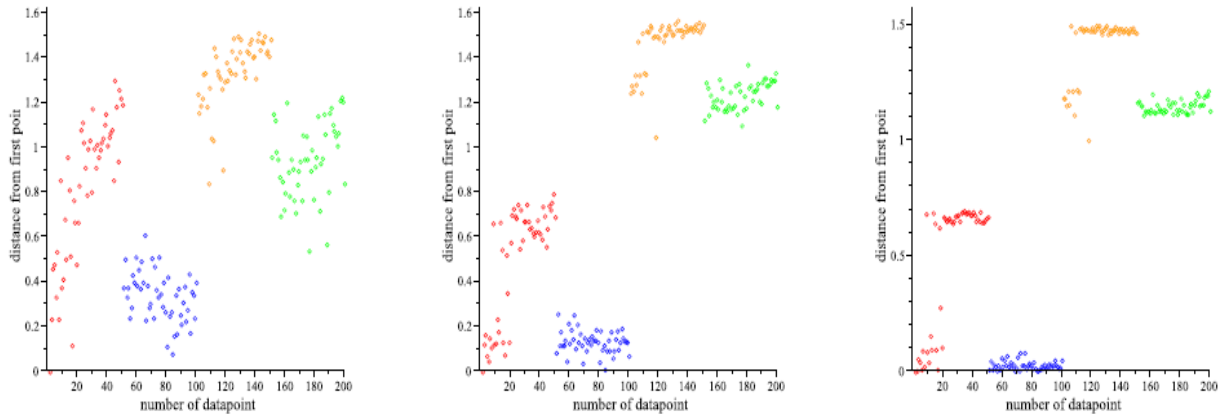


Figure 21: A plot of Euclidean distance of each point i from the first data point. Again, the left hand plot shows the distances for the initial distribution of points. The middle plot shows the same distances after quantum evolution. The right-hand plot shows results after another iteration of DQC. The numbering of the data-points is ordered according to the expert classification of these points into four classes containing 50 instances each (Weinstein, Marvin, Horn, 2009).

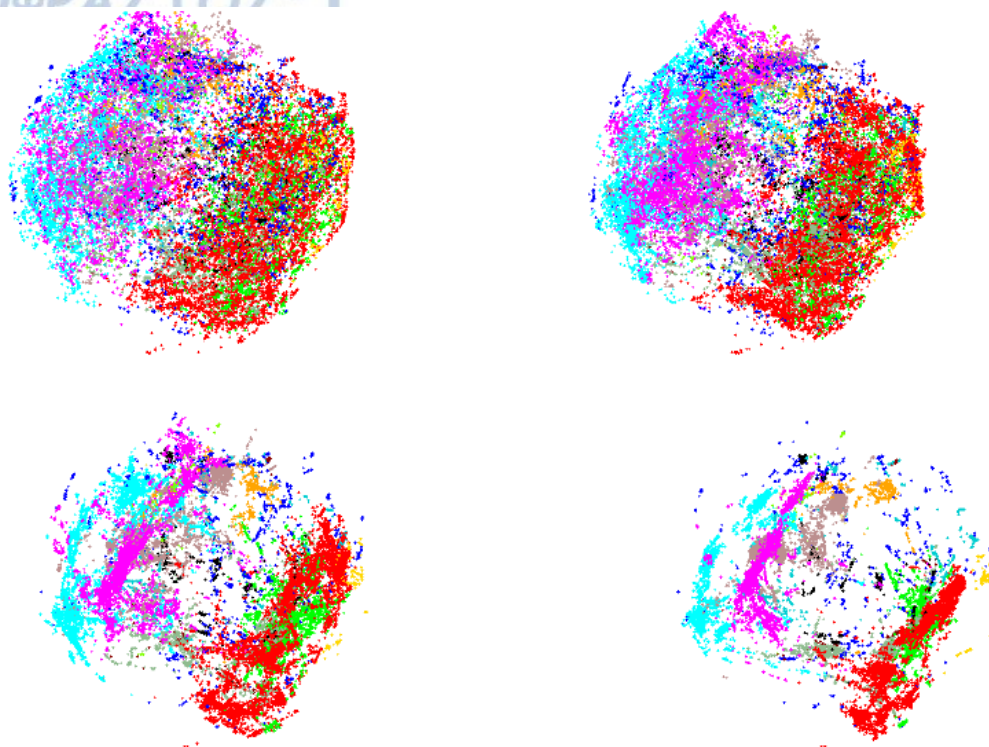


Figure 22: A plot of the first three principal components for a large data-set, comprising 35,213 points in 20 dimensions, before and after DQC evolution. The potential was determined from the full data-set and evolved using a sub-set of 1200 points, whose Gaussians serve as an essentially linearly set of independent states. Three stages of DQC development are shown. The coloring was decided upon by selecting the most obvious clusters from the evolved data and assigning colors to them. The dark blue points correspond to points that we did not bother to assign to clusters. The purpose of coloring is to be able to look at the points in the original data, discern those that belong to common structures, and follow their dynamic distances under DQC evolution (Weinstein, Marvin, Horn, 2009).

In this section we present a novel clustering method to microarray expression data in simple steps:

The first stage involves compression of dimensions that can be achieved by applying Singular Value Decomposition (SVD) to the gene–sample matrix in microarray problems. Thus, the data (samples or genes) can be represented by vectors in a truncated space of low dimensionality (Horn, Axel, 2003).

We find it preferable to project all vectors onto the unit sphere before applying a clustering algorithm.

The clustering algorithm used here is the quantum clustering method that has one free scale parameter. Although the method is not hierarchical, it can be modified to allow hierarchy in terms of this scale parameter (Horn, Axel, 2003).

More specifically, Singular Value Decomposition Algorithm concerns an $m \times n$ gene/sample matrix X . Its columns may be interpreted as sample vectors defined in gene-space, and its rows are gene-vectors in sample space. This matrix of rank $k \leq \min(m, n)$ can be expanded into a sum of k unitary matrices of rank 1:

$$X = \sum_{a=1}^k \sigma_a \mathbf{u}_a \mathbf{v}_a^T \quad (4.18)$$

The two sets $\{\mathbf{u}_a\}$ and $\{\mathbf{v}_\beta^T\}$, where $\alpha, \beta = 1, \dots, k$, of column and row vectors, respectively, are orthonormal sets. This expression can be rewritten in the matrix representation:

$$X = U \Sigma V^T \quad (4.19)$$

where Σ is a (non-square) diagonal matrix, and U, V are orthogonal matrices.

Ordering the non-zero elements of Σ in descending order, we can get an approximation of a lower rank r to the matrix X by taking $\sum_{jj}^r = 0$ for $j > r$, leading to the matrix:

$$Y = U \Sigma^r V^T \quad (4.20)$$

This is the best approximation of rank r to X , i.e. it leads to the minimal sum of square deviations:

$$S = \sum_i^m \sum_j^n (X_{ij} - Y_{ij})^2 \quad (4.21)$$

Once we apply SVD to a given matrix X , we automatically define two spaces dual to each other. The matrix U has orthogonal columns (eigensamples) that serve as axes for representing all genes (rows of U), while the matrix V has orthogonal columns (eigengenes) that serve as axes of a space representing all samples (rows of V or columns of V^T). Truncating these representations to dimension r , the gene-vectors (truncated rows of U) and the sample-vectors (truncated columns of V^T) do not have equal norms (Horn, Axel, 2003).

This leads to a problem for the clustering algorithm that is applied in these spaces since many vectors accumulate around the origin. We employ therefore

rescaling of all vectors to unit length. In other words, we project these vectors onto the unit sphere in r -space (Horn, Axel, 2003).

We present the Quantum Clustering Algorithm suggested by Horn and Gottlieb (2002) in Appendices (Appendix C).

We apply our method to three data sets. The results are very promising. On cancer cell data we obtain a dendrogram that reflects correct groupings of cells. In an AML/ALL data set we obtain very good clustering of samples into four classes of the data. Finally, in clustering of genes in yeast cell cycle data we obtain four groups in a problem that is estimated to contain five families (Horn, Axel, 2003).

We want to describe the quality of the results, so we calculate at each stage of σ , the Jaccard score:

$$J = \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \quad (4.27)$$

where n_{11} is the number of pairs of samples that appear in the same cluster both according to the cell type and according to our clustering algorithm, whereas $n_{10} + n_{01}$ is the number of pairs that appear together in one classification and not in the other. This score should be 1, for perfect clustering and decrease as the clustering quality decreases (Horn, Axel, 2003).

We compare here the QC results with a k-means analysis, which turns out to be worse. The Jaccard scores are 0.72 for the best QC result (varying over σ) and 0.48 for the best k-means (varying over k and averaging over initial conditions). It can be seen in Figure 24 that the $k = 4$ k-means analysis has one quite empty cluster. Indeed, the best k-means results were obtained for $k = 3$ (Horn, Axel, 2003).

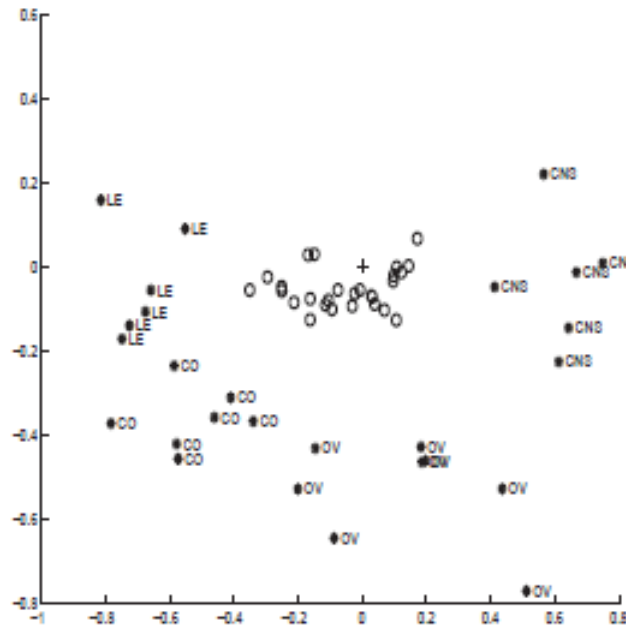


Figure 23: . Representation of data of four classes of cancer cells on two dimensions of the truncated space. These data points (denoted by star and by the relevant letters) are shown after the normalization of each data point in r -space. The circles denote the locations of the data points before this normalization was applied (Horn, Axel, 2003).

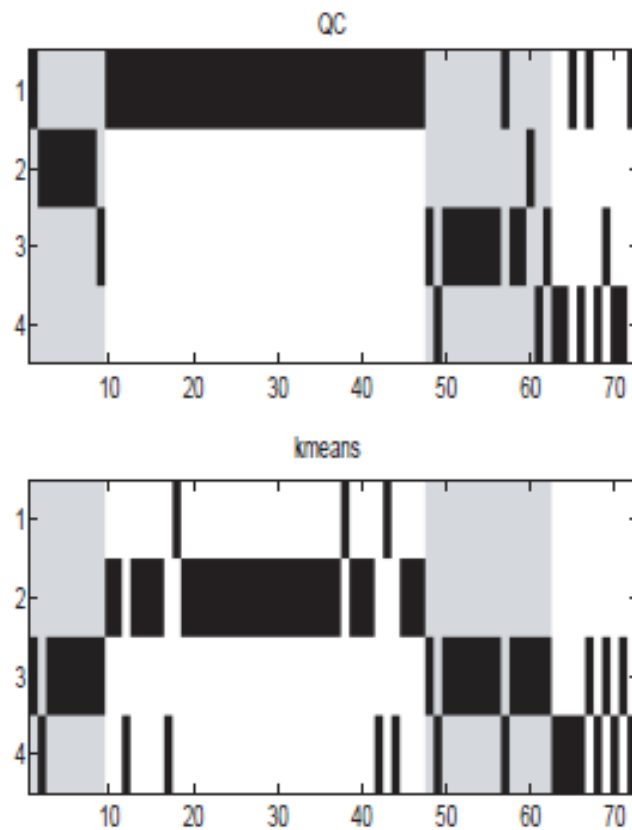


Figure 24: Clustering solutions for the AML/ALL problem using QC with $\sigma = 0.54$ (upper frame) and k-means with $k = 4$ (lower frame). The samples are ordered on the x-axis according to the true classification into four groups, indicated by alternative gray and white areas (Horn, Axel, 2003).

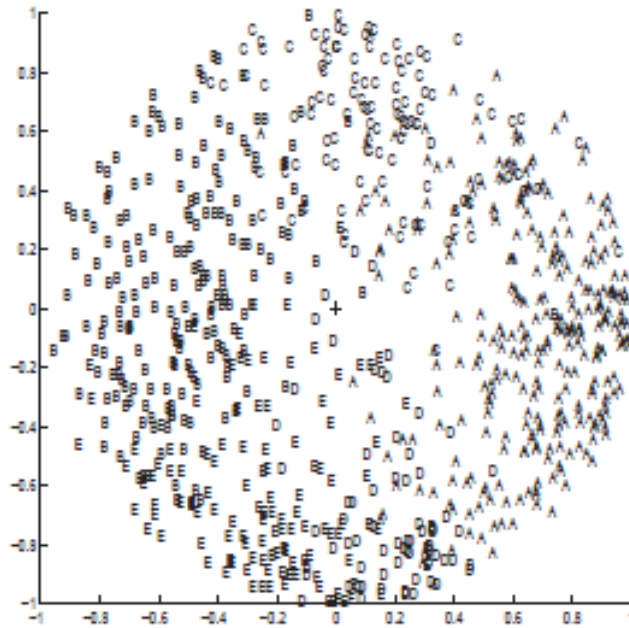


Figure 25: The five gene families as represented in two coordinates of our $r = 4$ dimensional truncated space (Horn, Axel, 2003).

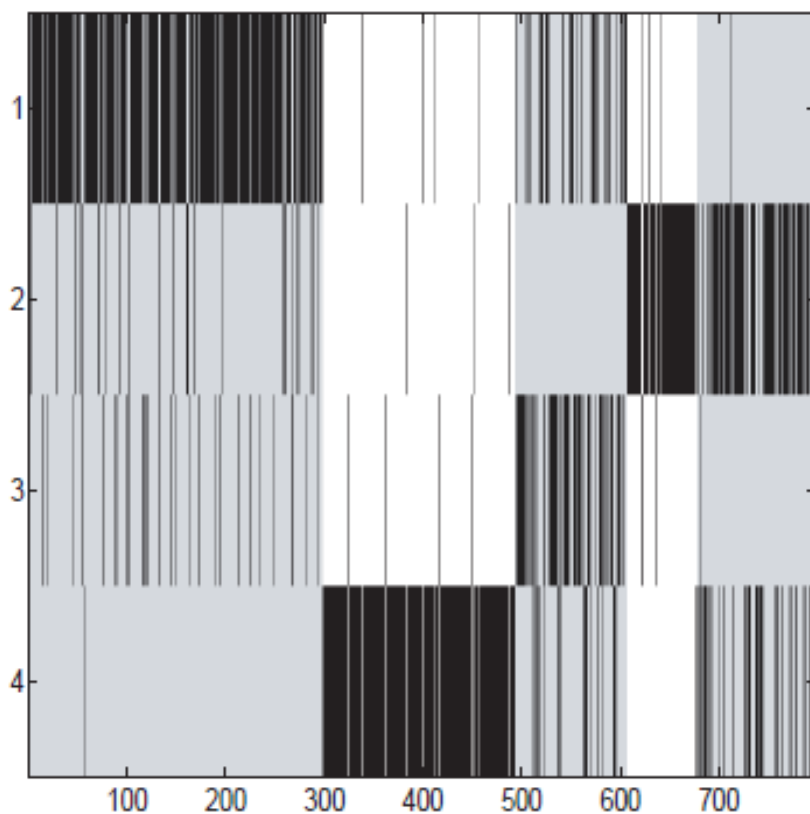


Figure 26: Cluster assignments of genes for QC with $\sigma = 0.46$ as compared to the classification by (Spellman et al., 1998) shown as alternating gray and white areas (Horn, Axel, 2003).



QUANTUM CLUSTERING: NOVEL APPLICATION

We construct the following example:

Data set from UCI Repository: **Absenteeism at work Data Set**

Abstract: The database was created with records of absenteeism at work from July 2007 to July 2010 at a courier company in Brazil.

Number of Instances: 740

Attribute Characteristics: Integer, Real

Number of Attributes: 21

Download date: 12 January 2019

Attribute Information:

1. Individual identification (ID)
2. Reason for absence (ICD).

Absences attested by the International Code of Diseases (ICD) stratified into 21 categories (I to XXI) as follows:

I Certain infectious and parasitic diseases

II Neoplasms

III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism

IV Endocrine, nutritional and metabolic diseases

V Mental and behavioural disorders

VI Diseases of the nervous system

VII Diseases of the eye and adnexa

VIII Diseases of the ear and mastoid process

IX Diseases of the circulatory system

X Diseases of the respiratory system

XI Diseases of the digestive system

XII Diseases of the skin and subcutaneous tissue

XIII Diseases of the musculoskeletal system and connective tissue

XIV Diseases of the genitourinary system

XV Pregnancy, childbirth and the puerperium

XVI Certain conditions originating in the perinatal period

XVII Congenital malformations, deformations and chromosomal abnormalities

XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified

XIX Injury, poisoning and certain other consequences of external causes

XX External causes of morbidity and mortality

XXI Factors influencing health status and contact with health services.

And 7 categories without (CID) patient follow-up (22), medical consultation (23), blood donation (24), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28).

3. Month of absence

4. Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))

5. Seasons (summer (1), autumn (2), winter (3), spring (4))

6. Transportation expense

7. Distance from Residence to Work (kilometers)

8. Service time

9. Age

10. Work load Average/day

11. Hit target

12. Disciplinary failure (yes=1; no=0)

13. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))

14. Son (number of children)

15. Social drinker (yes=1; no=0)

16. Social smoker (yes=1; no=0)

17. Pet (number of pet)

18. Weight

19. Height

20. Body mass index

21. Absenteeism time in hours (target)

Firstly, we implement classical statistical methods for clustering, like k-Means, hierarchical clustering and Model-based clustering for the data set “Absenteeism at work”. Then, for the same data set we implement Quantum clustering method and try to interpret and compare the results of classical and quantum methods.

For this implementation we use David Horn’s quantum clustering algorithm in R. It does not suffer from the curse of dimensionality and takes advantage of eigenfunctions for non-linear clustering. It was adapted from <http://horn.tau.ac.il/software/qc.m> with help from the following paper: Algorithm for Data Clustering Pattern Recognition Problems Based on Quantum Mechanics (2002).

- **Quantum Clustering:**

The goal is to perform clustering analysis according to the target variable “Absenteeism time in hours” of the data set.

Firstly, we implemented the classical methods of cluster analysis and more specifically Partitioning (k-Means) method, Hierarchical (Ward) method and Model-based method (Bayes criteria and maximum likelihood estimation).

These methods are in appendices (see Appendix I).

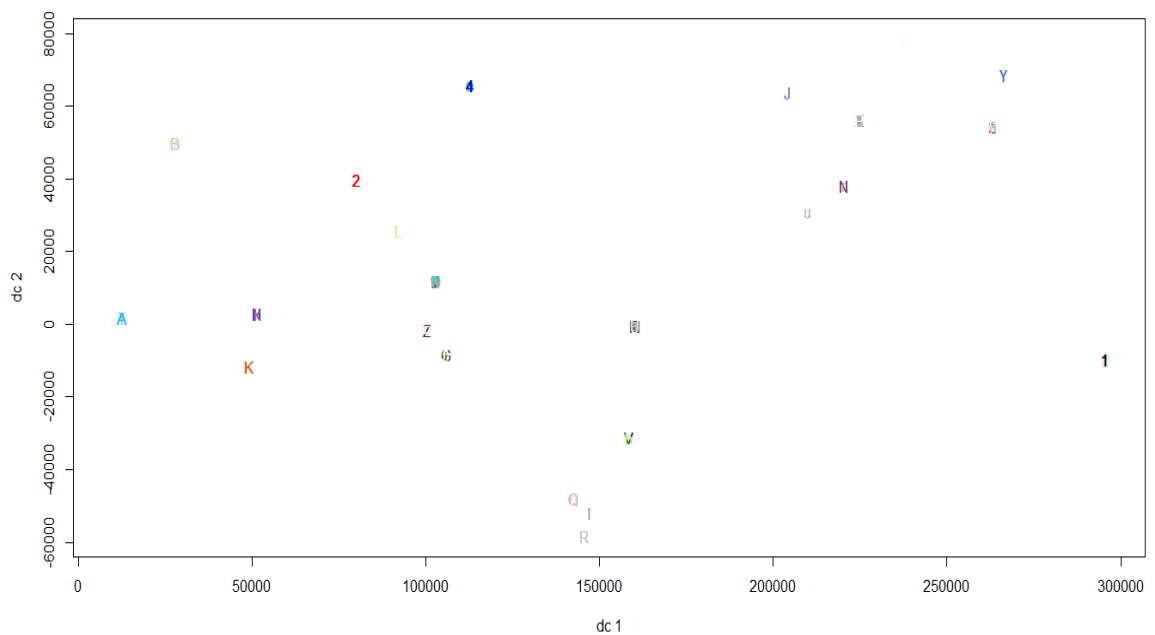
Then, we perform the quantum clustering method.

```
81 #Quantum Clustering|
82
83 clusters14 <- qc(mydatacl, sigma=1, steps=21, min_d_factor=2,
84                 n_clusters_max=14, verbose=FALSE)
85 clusters14
86
87 #Plotting
88 plotcluster(mydatacl, clusters)
```

```
> clusters14 <- qc(mydatacl, sigma=1, steps=21, min_d_factor=2,
+                  n_clusters_max=14, verbose=FALSE)
```

```
> clusters14
```

```
[1] 5 0 1 0 5 1 0 0 0 0 0 0 0 0 1 0 1 10 0 0 0 10 0 0 5 0 5 0
[29] 5 0 10 0 0 0 10 0 5 0 0 0 0 11 10 0 0 11 10 0 0 0 0 0 2 0 0 0
[57] 2 6 0 0 6 9 0 0 0 0 6 2 0 2 0 8 6 2 9 8 2 0 0 9 6 2 6 0
[85] 8 0 0 0 2 0 2 0 2 9 0 2 0 8 0 0 0 2 9 0 0 2 9 2 0 0 0
[113] 0 3 7 7 12 3 0 3 7 3 7 7 7 7 7 0 0 7 0 7 7 3 0 0 0 0 0
[141] 0 0 14 0 0 0 3 3 0 0 3 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 12
[169] 0 0 0 3 0 0 3 0 0 0 0 0 0 3 0 3 0 3 12 0 12 3 0 0 0 0 0
[197] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[225] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 1 0 5 0
[253] 5 0 5 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[281] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 9 0 0 0 8
[309] 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 6 0 0 0 0 9
[337] 0 0 6 2 0 0 0 9 0 6 6 0 0 6 0 0 0 10 5 3 0 12 0 12 0
[365] 0 0 0 1 1 7 1 0 0 14 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 0
[393] 0 14 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 14 0 0 13 0 1
[421] 0 0 14 1 0 0 13 0 4 1 0 14 0 0 0 0 4 0 0 4 0 1 0 0 14 0
[449] 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 10 0 5
[477] 0 0 0 0 1 0 0 11 5 0 0 10 0 0 0 0 11 0 0 0 0 11 0 0 0 10 0
[505] 0 0 0 0 0 0 0 8 0 0 8 2 0 0 2 0 6 0 8 0 2 0 2 0 0 0 2
[533] 0 0 0 6 0 0 0 2 0 8 0 2 0 0 0 2 0 0 0 2 0 0 8 2 0 2
[561] 2 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 12 0 1 0 1 1 1 3 0
[589] 1 3 1 4 0 1 0 1 0 1 1 3 4 0 1 0 4 1 1 1 12 0 1 0 1 13 3 1
[617] 1 0 1 13 0 13 0 1 0 1 0 4 0 0 1 1 1 4 0 13 1 0 1 0 0 1 13 1
[645] 4 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[673] 0 0 0 0 0 0 0 4 0 0 0 0 4 0 0 0 4 0 0 0 0 0 0 0 0 0
[701] 4 0 0 0 0 0 0 4 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
[729] 11 0 0 11 0 0 0 5 0 0 0 0 0 0 0 0 4 0 0 11 0 0 0 0 0 0
```



```
#Quantum Clustering
clusters <- qc(mydatacl, sigma=1, steps=21, min_d_factor=2,
               n_clusters_max=740, verbose=FALSE)
clusters
```


Figure 31: Maximum number of clusters = 740

```

> clusters
[1] 5 303 1 302 5 1 66 22 301 21 65 65 65 64 1 300 1 10 299 298 297
[22] 10 296 22 5 122 5 295 5 294 10 121 293 120 10 122 5 292 121 22 291 11
[43] 10 45 119 11 10 32 22 118 117 116 2 16 115 114 2 6 290 63 6 9 20
[64] 16 114 113 6 2 16 2 63 8 6 2 9 8 2 112 289 9 6 2 6 111
[85] 8 288 110 287 2 286 2 111 2 9 62 2 285 8 62 109 108 44 2 9 109
[106] 31 2 9 2 31 284 283 282 3 7 7 12 3 61 3 3 7 3 7 7 7
[127] 7 7 7 15 19 7 61 7 7 3 19 61 281 64 280 107 14 43 106 106 3
[148] 3 15 279 3 105 105 3 107 278 60 277 276 30 60 104 103 102 275 274 60 12
[169] 19 273 101 3 272 100 3 42 15 42 42 29 271 3 103 3 30 3 12 41 12
[190] 3 101 100 270 30 99 30 269 102 19 268 59 19 267 98 97 266 58 29 42 265
[211] 58 29 97 264 263 262 261 29 96 260 41 259 41 30 99 29 58 41 96 258 257
[232] 256 59 255 254 253 252 251 57 250 249 40 248 56 56 247 246 5 1 245 5 244
[253] 5 243 5 95 94 94 22 5 93 242 40 39 93 21 39 21 241 240 92 91 239
[274] 238 237 236 235 234 92 233 91 232 231 230 229 228 227 226 225 90 89 90 88 224
[295] 223 55 222 28 6 221 54 88 9 54 87 87 53 8 220 53 86 117 85 219 55
[316] 55 53 27 62 218 9 217 216 215 214 213 44 27 44 6 212 211 210 20 209 9
[337] 208 207 6 2 20 20 89 108 9 206 6 6 113 16 44 6 205 204 52 203 10
[358] 5 3 202 12 201 12 200 199 198 197 1 1 7 1 18 43 14 1 1 1 1
[379] 51 1 196 1 26 26 195 1 1 194 1 1 25 26 51 14 1 84 38 193 1
[400] 83 192 191 82 24 24 190 189 81 1 43 188 26 1 14 37 50 13 37 1 24
[421] 187 186 14 1 49 185 13 50 4 1 83 14 38 38 48 36 37 4 25 184 4
[442] 25 1 25 36 14 183 118 1 26 1 80 1 35 39 64 40 51 56 21 40 35
[463] 22 182 79 181 52 11 180 120 17 179 10 39 66 5 45 17 32 178 1 78 17
[484] 11 5 45 32 10 78 79 21 17 11 32 52 17 17 11 119 32 177 176 10 35
[505] 175 28 86 77 76 174 77 8 16 173 8 2 28 28 2 28 6 172 8 75 2
[526] 75 2 110 171 170 116 2 20 16 169 6 27 112 76 2 31 8 27 2 20 16
[547] 27 2 85 115 63 54 2 168 31 167 8 2 166 2 2 31 165 164 163 162 2
[568] 2 2 161 15 104 23 23 15 23 160 23 15 12 15 1 19 1 1 1 3 159
[589] 1 3 1 4 158 1 74 1 18 1 1 3 4 23 1 74 4 1 1 1 12
[610] 73 1 18 1 13 3 1 1 157 1 13 48 13 156 1 36 1 36 4 73 155
[631] 1 1 1 4 154 13 1 18 1 43 48 1 13 1 4 1 1 153 152 1 1
[652] 151 150 50 72 71 70 149 69 34 148 147 146 70 71 145 34 72 144 34 143 69
[673] 142 34 84 47 68 49 80 4 33 47 141 47 4 24 24 25 81 4 140 67 139
[694] 138 33 38 18 37 68 33 4 137 136 67 135 18 134 4 133 49 59 132 131 4
[715] 98 130 82 4 33 129 11 46 57 57 46 45 46 95 11 128 127 11 66 35 126
[736] 5 21 125 124 123

```

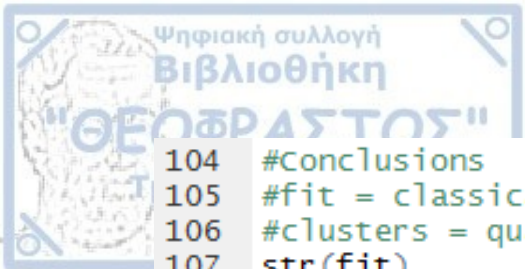
- Changing sigma (controls how closely related data in clusters should be) and n_clusters max (number of clusters) and searching best choice

```

94 ##Change sigma and n_clusters_max = number of clusters (searching best choice)
95 clusters5s20000 <- qc(mydata, sigma=20000, steps=21, min_d_factor=2,
96                        n_clusters_max=5, verbose=FALSE)
97 clusters5s20000
98 clusters50s100
99
100 ## K-Means clustering with 50 clusters
101 fit50 <- kmeans(mydata, 50)
102 fit50$cluster

```

[illegible]



```

104 #Conclusions
105 #fit = classical clustering
106 #clusters = quantum clustering
107 str(fit)
108 str(clusters)
109
110
111 # comparing 2 cluster solutions
112 install.packages("fpc")
113 library(fpc)
114
115 d <- dist(mydatacl, method = "euclidean") # distance matrix

> #Conclusions
> #fit = classical clustering
> #clusters = quantum clustering
> str(fit)
List of 9
 $ cluster      : int [1:740] 5 7 14 2 5 14 10 11 8 3 ...
 $ centers      : num [1:14, 1:20] -0.7519 -0.278 -0.1047 0.1099 0.0965 ...
 .. attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:14] "1" "2" "3" "4" ...
 .. ..$ : chr [1:20] "Reason for absence" "Month of absence" "Day of the week" "Seasons" ..
 .
 $ totss       : num 14780
 $ withinss    : num [1:14] 337 496 196 128 1459 ...
 $ tot.withinss: num 7122
 $ betweenss   : num 7658
 $ size        : int [1:14] 32 39 27 35 134 46 60 80 13 32 ...
 $ iter        : int 5
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
> str(clusters)
num [1:740] 5 303 1 302 5 1 66 22 301 21 ...

```

- **Conclusions from the results above**

```

117 #1.For example the bigger the diameter of the cluster, the worst
118 #the clustering, because the points that belong to the cluster
119 #are more scattered.
120 #2.The higher the average distance of each clustering, the worst
121 #the clustering method. (Let's assume that the average distance
122 #is the average of the distances from each point in the cluster
123 #to the center of the cluster.)
124 cluster.stats(d, fit50$cluster)
125 cluster.stats(d, clusters50s100)
126
127
128
129
130 #First Conclusion: Many clusters ==> quantum clustering
131 # Less clusters ==> classical clustering
132 #Second Conclusion: The quantum algorithm is approximately  $O(n^2)$ 

```

All statistics of quantum clustering can be found in appendices (see Appendix J) and compared with the statistics of classical methods of clustering analysis, such as k-Means partitioning and hierarchical method.

Here, qc {quantum clustering}: Cluster data using a non-linear clustering algorithm which finds organic groups of data from eigenfunctions and does not suffer from the curse of dimensionality. The algorithm is approximately $O(n^2)$. The function qc returns a vector of the numeric clusters assigned to each row in data set.

```
qc(data set, sigma, steps = 21, min_d_factor = 2, n_clusters_max = 1000, verbose = FALSE)
```

Arguments:

data set: The cleaned input data to be clustered in either data frame or matrix format. This should contain only numeric data and must not have any factors, strings, NAs, etc. It also should not contain any irrelevant columns such as observation ID or redundant data.

sigma: A double which controls how closely related data in clusters should be. The smaller the number, more clusters will be created with fewer observations in each. If sigma is too small, observations either will not be clustered or will be in their own individual clusters. If sigma is too large, most – if not all – observations will be in the first cluster

steps: An integer specifying the number of expectation-maximization steps to take. If faster, less accurate results are required, this may be reduced from the default of 21

min_d_factor: A double which controls how close data points must be in order to be considered in the same cluster. Specifically, this value is the number of sigmas of distance to be within said threshold. This value should probably not be changed unless there is a strong reason to do so

n_clusters_max: An integer specifying the maximum number of clusters to return. These clusters will always be the most common clusters with the most observations in them

verbose: A boolean value which toggles the algorithm's verbosity for details as to how far along it is.

We prefer to present our visual results of Quantum Clustering using Matlab language because in Matlab environment the results are more comprehensible than in other programming languages like R or Python.

The necessary functions for our representation are provided in appendices ([Appendix D](#)).

We implement the above code in Matlab and we take the following plot and output:

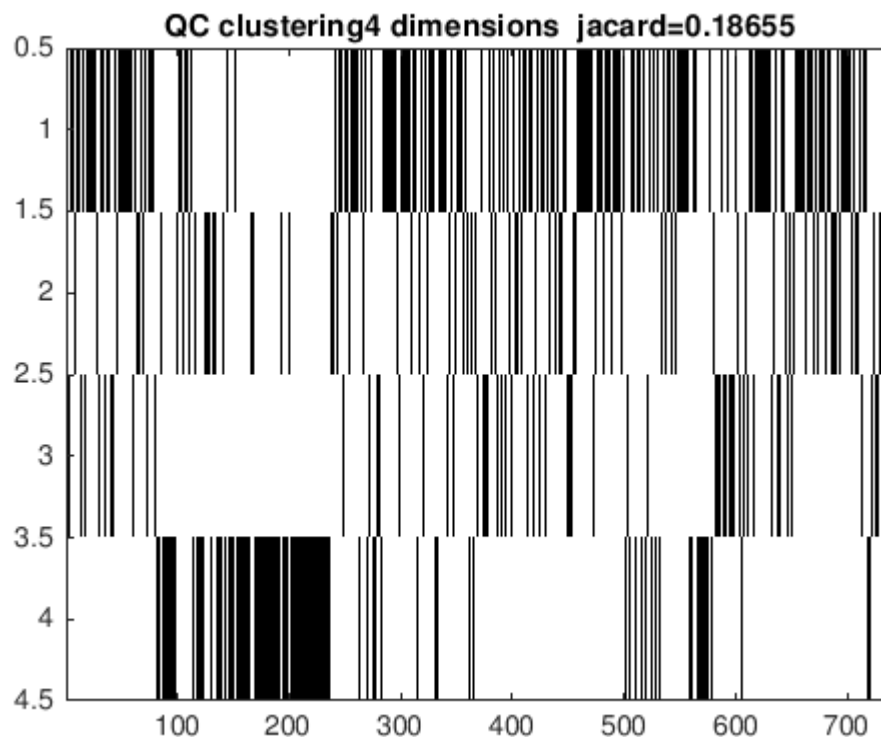
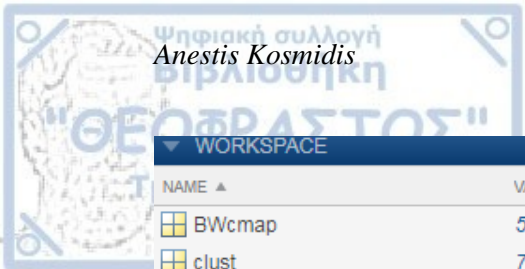


Figure 32: In my data set horizontal axis refers to observations and vertical axis refers to the reduction of dimensions of my data set in 4 (like PCA or SVD method)



WORKSPACE			
NAME ▲	VALUE	SIZE	CLASS
BWcmap	5×3 double	5×3	double
clust	740×1 double	740×1	double
colheaders	1×20 cell	1×20	cell
D	740×4 double	740×4	double
data	740×20 double	740×20	double
dims	4	1×1	double
efficiency	0.3207	1×1	double
genes	740×20 double	740×20	double
index	4×740 double	4×740	double
jm	0.1866	1×1	double
M	740×20 double	740×20	double
mm	1.1826	1×1	double
n	4×740 double	4×740	double
purity	0.3084	1×1	double
q	2.4000	1×1	double
QCjacard_measure	0.1866	1×1	double
QCminkowski_measure	1.1826	1×1	double
realClust	[1,300,495,608,678]	1×5	double
S	20×20 double	20×20	double
samples	20×20 double	20×20	double
textdata	1×20 cell	1×20	cell
x	4×740 logical	4×740	logical
xyData	740×4 double	740×4	double

Figure 33: This table contains all the statistical measures appeared during Quantum Clustering in the Matlab code

Here we present a second visualization of Quantum Clustering for the same data set using Matlab environment. We provide the necessary functions for the main code in the appendices ([Appendix E](#)). Then, we take the following plots:

step #24/25

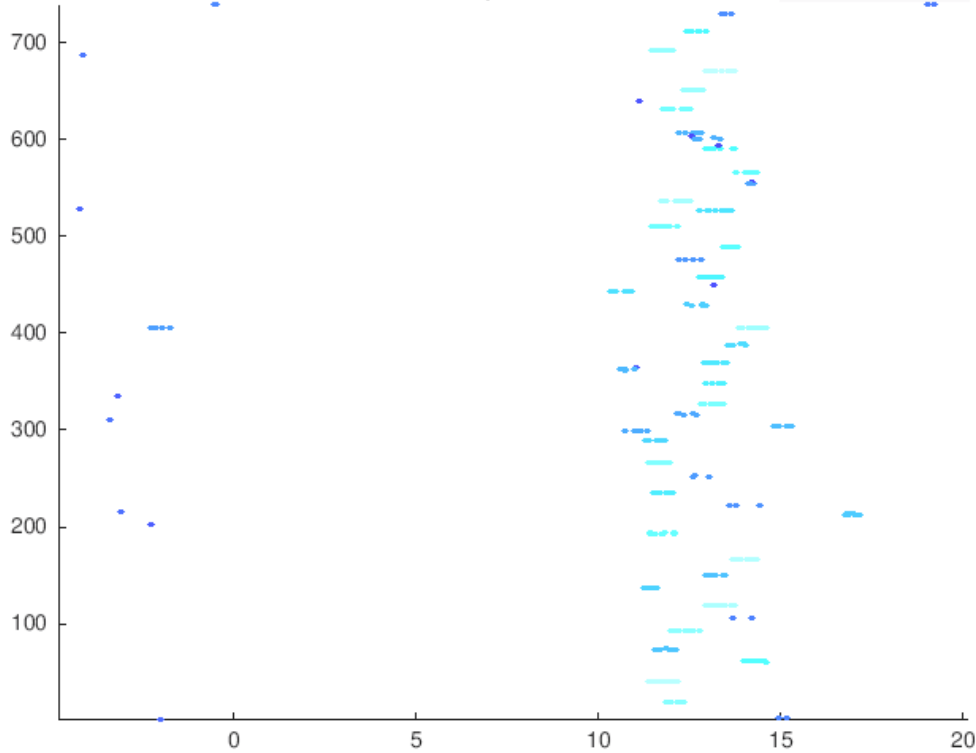


Figure 34: How data is presented before Quantum Clustering, where horizontal axis refers to the fraction V/E and vertical axis refers to the 740 observations of the data set. We view ψ

as an eigenstate of the Schrödinger equation: $H\psi \equiv \left(-\frac{\sigma^2}{2} \nabla^2 + V(\mathbf{x})\right)\psi = E\psi$ and

here we rescale H and V of the conventional quantum mechanical equation to leave only one free parameter σ . The case of a single point at \mathbf{x}_1 corresponds to Schrödinger equation with $V = \frac{1}{2\sigma^2} (\mathbf{x} - \mathbf{x}_1)^2$ and $E = \frac{d}{2}$, where d is the dimension of the Euclidean space. The values V/E are shown as functions of the serial number of the data. Lower cutoffs in V/E , including lower fractions of data, are required to define cluster cores that are well-separated in their relevant spaces.

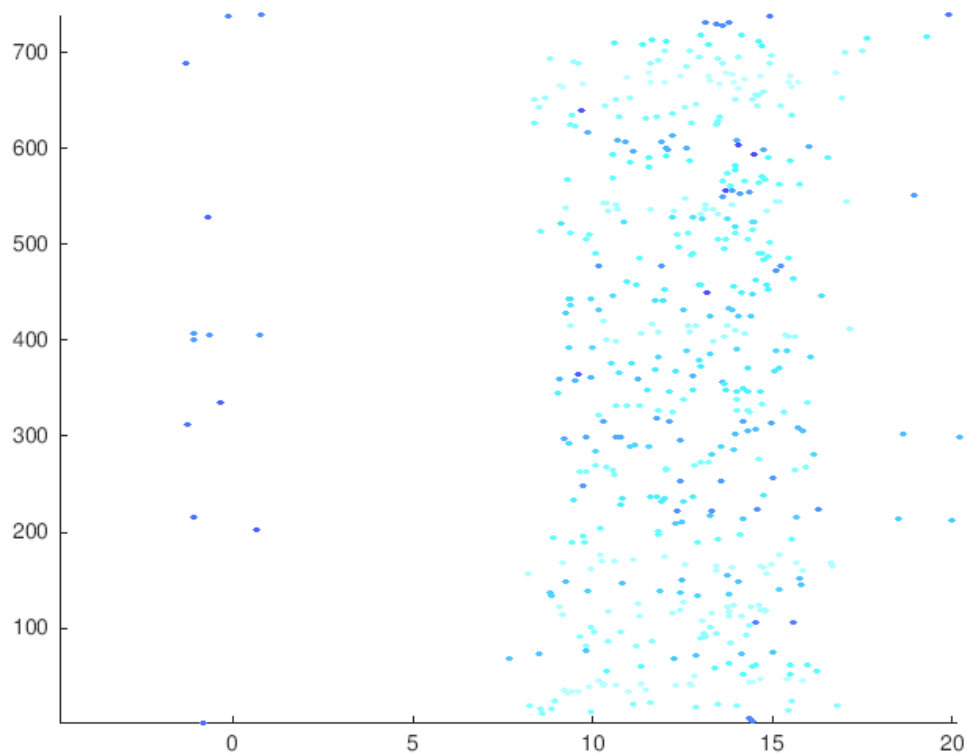
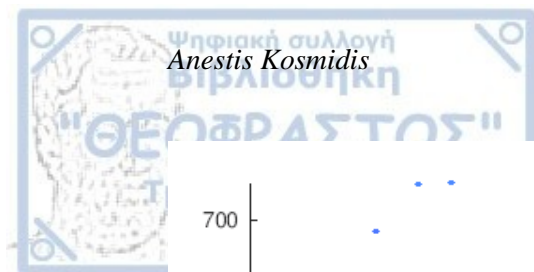
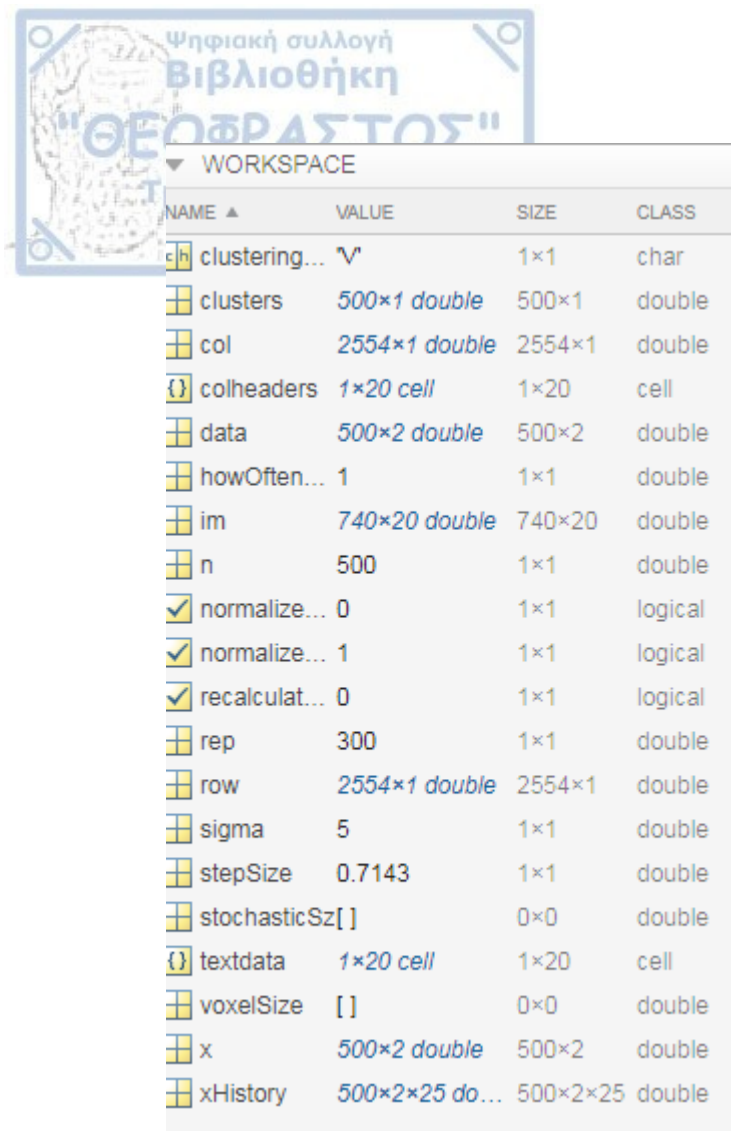


Figure 35: How data is presented after Quantum Clustering (different colours means different clusters from PCA), where horizontal axis refers to the fraction V/E and vertical axis refers to the 740 observations of the data set.

According to the figures of Quantum Clustering we observe that after Quantum Clustering the observations and cores are better separated than before Quantum Clustering.



NAME	VALUE	SIZE	CLASS
clustering...	'V'	1×1	char
clusters	500×1 double	500×1	double
col	2554×1 double	2554×1	double
colheaders	1×20 cell	1×20	cell
data	500×2 double	500×2	double
howOften...	1	1×1	double
im	740×20 double	740×20	double
n	500	1×1	double
normalize...	0	1×1	logical
normalize...	1	1×1	logical
recalculat...	0	1×1	logical
rep	300	1×1	double
row	2554×1 double	2554×1	double
sigma	5	1×1	double
stepSize	0.7143	1×1	double
stochasticSz[]		0×0	double
textdata	1×20 cell	1×20	cell
voxelSize	[]	0×0	double
x	500×2 double	500×2	double
xHistory	500×2×25 do...	500×2×25	double

Figure 36: This table contains all the statistical measures appeared during Quantum Clustering in the Matlab code

Therefore, for Clustering Issues we developed a Quantum Algorithm using the programming language R, which can be implemented in every data set. Then, we plotted the results of the Quantum Clustering Algorithm using the Matlab program. We also implemented the classical statistical methods for Clustering, Partioning (k-Means) method, Hierarchical method (Ward distance) and Model-based method (Bayes criteria and Maximum Likelihood estimation) and plotted their results. We validated cluster solutions through cross-validation and compared the statistics, which comes from the methods k-Means and Quantum Clustering.

We conclude that the comparison between Classical and Quantum Clustering Algorithms leads to results, which prove that in some cases Quantum Clustering algorithm prevails of the corresponding Classical Clustering algorithm. However, these results can not be generalized yet.

Moreover, we observe it is quite difficult to compare the results between Classical and Quantum Clustering methods visually. However, Quantum Clustering is an additional beyond novel method to handle a data set and it is very possible with few modifications in the code to take improved results in the near future.

Comments on the Results

For the clustering algorithm we select a large data set, prepare the data and implement:

- Partitioning (k-Means) method
- Hierarchical method (Ward)
- Model-based method (Bayes criteria and Maximum Likelihood estimation)

We plot clustering solution methods and implement Quantum Clustering. Next we validate the cluster solutions (cross-validation), we observe and compare the statistics which comes from the methods k-Means and Quantum Clustering. However, we are not able to construct analogous plots for all the methods to compare them. For this reason we plot quantum clustering in Matlab. In the first figure horizontal axis refers to observations and vertical axis refers to the reduction of dimensions of my data set in 4 (like PCA or VSD method). In the first figure of second application we see how data is presented before Quantum Clustering and in the second figure we see how data is presented after Quantum Clustering (different colours means different clusters from PCA). Here, horizontal axis refers to the fraction V/E and vertical axis refers to the 740 observations of the data set. For the statistical results of cross-validation, we can check comments of R-studio algorithm. We can implement this algorithm in every data set. In another data set (or in other variables) somebody could extract better or more useful clustering results. In worst case, these new quantum methods could be used as an extra method of machine learning.

CHAPTER 5: QUANTUM REGRESSION

We introduce the Quantum-inspired Machine Learning Algorithms for Regression. These are machine learning algorithms that involve in some quantum theoretical elements, but do not require a quantum machine for implementing it. We present a novel ensemble regression algorithm inspired by quantum mechanics and the theoretical connection between quantum interpretations and machine learning algorithms. The goal of ensemble learning is to combine the predictions of multiple base learners to get more accurate aggregate predictions (Xie, Sato, 2017).

Suppose we are given a data set $X \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^n$ for a regression or classification problem. X is a $n \times m$ data matrix, which contains n data samples, and each feature vector \mathbf{x}^i has m features. The target variable vector \mathbf{y} is a vector with a length of n . We define the Gram matrix $P = XX^T$ that is a symmetric and positive semi-definite $n \times n$ matrix. Then, we have $P = XX^T = U\Sigma^P U^T$ where Σ^P is a $n \times n$ diagonal matrix. Column vectors of U are equal to principal components in Principal Component Analysis. People often use first k column features U as dimension-reduced k -dimension feature vectors (Xie, Sato, 2017).

As the density matrix of quantum mechanics is Hermitian, positive semi-definite and of trace 1, if we normalize the Gram matrix P by multiplying a factor $\frac{1}{\text{Tr}(P)}$, the Gram matrix can be regarded as a density matrix in quantum theory. We denote the normalized Gram matrix by ρ . We redefine ρ with a normalization factor as:

$$\rho = \frac{XX^T}{\text{Tr}(XX^T)} = U\Sigma U^T \quad (5.1)$$

Let \mathbf{u}_i denote the i -th column vector of matrix U , so \mathbf{u}_i is also a pure state vector, which denotes $|u_i\rangle$ in quantum theory. As we have replaced the Gram Matrix by the normalized ρ , the sum of diagonal elements of Σ : $\sum_{i=1}^n s_i^2 = 1$. The density matrix ρ describing the data matrix as a mixed state is also an operator of the form:

$$\rho = \sum_{i=1}^n s_i^2 |u_i\rangle\langle u_i| = \sum_{i=1}^r s_i^2 |u_i\rangle\langle u_i| \quad (5.2)$$

Physically, it means a data matrix X can be regarded as a mixed state or a quantum ensemble consisting of r pure states, where r is the rank. In physics, an ensemble of pure states ρ can reflect statistical expectations of quantum systems $|u_i\rangle$. And the variance s_i^2 is the fraction (weight probability) of the ensemble in each pure state $|u_i\rangle$ (Xie, Sato, 2017).

On the one hand, the quantum interpretation treats PCA naturally as a dimensionality reduction process. In machine learning, researchers usually preserve the first k components with largest variance values as dimensionality reduced features. In quantum mechanics, PCA means that we remove several non-principal eigenstates from the mixed state and preserve those principal eigenstates so that we prepare a new mixed state consisting of less eigenstates. The new state is exactly a low-rank approximated copy of the original mixed state. PCA is also a naive and biased operation that assigns uniform weights to principal eigenstates and weight 0 to non-principal eigenstates (Xie, Sato, 2017).

The second quantum interpretation is we can also regard regression as a state preparation process that we operate several pure states $|x_1\rangle, |x_2\rangle, \dots, |x_n\rangle$ to approximate a target state $|y\rangle$. Translated in quantum theoretical language, it can be written as:

$$\rho_y = |y\rangle\langle y| = \hat{A}\rho_x\hat{A}^\dagger \quad (5.3)$$

where the state operation is noted by some quantum operator \hat{A} . So the quantum mechanism of regression tasks can be understood as we learn a Model Operator to operate eigenstates in a mixed to approximate a target pure state under some metrics.

The importance of an eigenstate $|u_i\rangle$ is also reflected by the Transition Probability from an eigenstate $|u_i\rangle$ jumping into the target state $|y\rangle$. We denote Transition Probability Amplitude as t_i , so $t_i = \langle y|\hat{A}|u_i\rangle$. We note that Transition Probability equals Transition Probability Amplitude squared, namely $|\langle y|\hat{A}|u_i\rangle|^2$ (Xie, Sato, 2017).

Obviously, the Transition Probability is a parameter decided by model operator, the eigenstate, and the target state together. Aggregating fraction probabilities and transition probabilities together, the Fraction Transition Probability

for the i -th principal component is proportional to $s_i^2 |\langle y | \hat{A} | u_i \rangle|^2$. So we take the Fraction Transition Probability for the i th principal component as:

$$p_k = \frac{s_k^2 t_k^2}{\sum_{i=1}^r s_i^2 t_i^2} \quad (5.4)$$

Algorithm 1: Quantum-Inspired Subspace for generating feature subsets

function QISubspace (X, y, \mathcal{F}, T, K)

Input : the data matrix X , the target variable vector y , the ensemble size T , the target space dimensionality $K = \alpha m$

Output: feature subsets $\{\mathcal{F}_i | i = 1, \dots, T\}$

Preprocess data matrix $X_R \leftarrow PCA(X)$ by using full-rank PCA

Compute Fraction Probabilities $p_s \leftarrow$ the diagonal elements of covariance matrix $X^\top X$

Compute Transition Probability Amplitudes $t \leftarrow (X_R^\top X_R)^{-1} X_R^\top y$ which are LR parameters

Compute Transition Probabilities $p_t \leftarrow t \cdot t$

Compute Fraction Transition Probabilities $p \leftarrow \frac{p_s \cdot p_t}{\text{norm}(p_s \cdot p_t)}$

for $i \leftarrow 1$ **to** T **do**

 Select K unique random integers a_1, \dots, a_K from $[1, m]$ in probabilities of p_{a_i}

$\mathcal{F}_i \leftarrow \{a_1, \dots, a_K\}$

end

return $\{\mathcal{F}_i | i = 1, \dots, T\}$

Algorithm 2: Random Forest

function RandomForest (S, \mathcal{F}, T, K)

Input : A training set $S = (x^1, y^1), \dots, (x^n, y^n)$, features \mathcal{F} , and the forest size T , the target space dimensionality K

Output: Random Forest H

$H \leftarrow \emptyset$

for $i \leftarrow 1$ **to** T **do**

$S^i \leftarrow$ a bootstrap sample from S

$\mathcal{F}^i \leftarrow$ a random subset of size K sampled from \mathcal{F}

$h_i \leftarrow \text{TreeLearn}(S^i, \mathcal{F}^i)$

$H \leftarrow H \cup \{h_i\}$

end

return H

Algorithm 3: Quantum-Inspired Forest**function** QIForest (S, \mathcal{F}, T, K)**Input** : A training set $S = (x^1, y^1), \dots, (x^n, y^n)$, features \mathcal{F} , and the forest size T , the target space dimensionality K **Output**: Quantum-Forest H $H \leftarrow \emptyset$ $\{\mathcal{F}_i | i = 1, \dots, T\}$ generated by **function** QISubspace (X, y, \mathcal{F}, T, K)**for** $i \leftarrow 1$ **to** T **do** $S^i \leftarrow$ a bootstrap sample from S $\mathcal{F}^i \leftarrow \mathcal{F}_i$ $h_i \leftarrow \text{TreeLearn}(S^i, \mathcal{F}^i)$ $H \leftarrow H \cup \{h_i\}$ **end****return** H

(Xie, Sato, 2017)

We describe the basic steps of the above algorithms. More specifically, Random Subspace is a fast and efficient ensemble method widely used in many algorithms, including Random Forest. Random Subspace randomly select a subset of features for training a base learner. But Quantum-Inspired Subspace (QIS) can utilize the extra information inspired by quantum mechanics (Xie, Sato, 2017).

We first preprocess the input data matrix X by using full-rank PCA. Different from either preserving principal components with largest eigenvalues or random subspace, QIS selects a component in a probability proportional to the corresponding Fraction Transition Probability. Under Gaussian assumptions of model parameters, we let $p_k = \frac{s_k^2}{\sum_{i=1}^r s_i^2}$ for the component k . When we replace Random Subspace by Quantum-Inspired Subspace for Random Forest, we obtain a novel algorithm, namely Quantum-Inspired Forest. We note that, in principle, full-rank PCA preprocessing generally can neither improve nor damage algorithm performance. The additional computational cost of the proposed algorithm is only brought by Principal Component Analysis and several matrix operations for computing Fraction Transition Probabilities. So it is a very low cost in practice (Xie, Sato, 2017).

We denote by h_1, \dots, h_T the regressors in the ensemble and by \mathcal{F} , the feature set. We need to choose ensemble size T in advance. All base regressors can be trained in parallel, which is also the case with Bagging and Random Forests.

Algorithm 1 explains how to generate the training feature set \mathcal{F}_i for regressor h_i . And we modify Random Forest into Quantum-Inspired Forest by employing

Quantum-Inspired Subspace to generate ensemble feature subsets instead of Random Subspace. We can easily notice the difference between standard Random Forest and Quantum-Inspired Forest respectively described in Algorithm 2 and Algorithm 3 (Xie, Sato, 2017).

It is worthy noting that Quantum-Inspired Subspace is a general method which can be easily applied with other ensemble methods and multiple base learners together. QIS also lend itself naturally to parallel processing, as ensemble feature sets and individual learners can be built in parallel.

The proof following states that the advantages of QIS theoretically increase ensemble ambiguity and decrease the individual error expectation in the first order approximation. And in our empirical analysis, the experimental results support the advantage is still approximately applicable to nonlinear models, such as Decision Tree, as we see below.

Firstly, we show how to obtain Error-Variance-Covariance Decomposition. We use an ensemble of T base regressors h_1, h_2, \dots, h_T to approximate a function $f: \mathbb{R}^m \rightarrow \mathbb{R}$. Then, we use a simple averaging policy for the final ensemble prediction:

$$H(x) = \frac{1}{T} \sum_{i=1}^T h_i(x) \quad (5.5)$$

where $H(x)$ is the ensemble learner. We continue by defining several notations. The generalization error and ambiguity of a base learner are respectively defined as:

$$err(h_i) = (h_i(x) - f(x))^2 \quad (5.6)$$

$$ambi(h_i) = (h_i(x) - H(x))^2 \quad (5.7)$$

Moreover, we denote the expectation prediction of a base learner h_i as:

$$\mathbb{E}[h_i] = \int h_i(x) p(x) dx \quad (5.8)$$

where $p(x)$ is the density function for data x .

The error-ambiguity decomposition of ensemble learning and the generalization error of the ensemble can be written as:

$$err(H) = \overline{err}(H) - \overline{ambi}(H) \quad (5.9)$$

where $\overline{err}(H) = \frac{1}{T} \sum_{i=1}^T err(h_i)$ is the average of individual generalization errors and $\overline{ambi}(H) = \frac{1}{T} \sum_{i=1}^T ambi(h_i)$ is the average of ambiguities which is

also called the ensemble ambiguity. We know that the larger the ensemble ambiguity, the better the ensemble.

Furthermore, the averaged bias, averaged variance and averaged covariance of the individual learners are defined respectively as:

$$\overline{bias}(H) = \frac{1}{T} \sum_{i=1}^T (\mathbb{E}[h_i] - f) \quad (5.10)$$

Finally, we obtain Error-Variance-Covariance Decomposition as:

$$err(H) = \overline{err}(H) - \left(1 - \frac{1}{T}\right) \overline{variance}(H) + \left(1 - \frac{1}{T}\right) \overline{covariance}(H) \quad (5.11) \text{ (Xie, Sato, 2017).}$$

We present now the empirical analysis and compare Random Forest with Quantum-Inspired Forest method. We select 10 UCI data sets that are commonly used in the machine learning literature in order to make the results easier to interpret and compare. We compare Random Ensemble Linear Regression with Quantum-Inspired Ensemble Linear Regression in Table 40, where we replace Decision Tree by Linear Regression as base learners. Ensemble Linear Regressors are not useful in practice, but it can show how our proof holds (Xie, Sato, 2017).

We take the averaged mean square error (MSE) on 10 data sets as the metrics in our empirical analysis. We decide to preprocess data sets and take full-rank PCA preprocessed data matrix and mean normalized target variables y as preprocessed data sets. The first purpose is to ensure any performance differences are purely caused by the proposed Quantum-Inspired Subspace method rather than full-rank PCA preprocessing. We must leave the difference from full-rank PCA out. The second purpose is to remove the scale differences of different data sets so that we can fairly evaluate overall performance on 10 data sets. It's reasonable to start from full-rank PCA preprocessing because full-rank PCA is only an orthogonal transformation and causes no loss or distortion of information. As we mentioned above, in principle, full-rank PCA generally can neither improve nor damage algorithm performance. In practice, full-rank PCA usually brings in uncertain performance improvement or

damage. So the full-rank PCA preprocessing is necessary for removing the uncertain performance differences from the orthogonal transformation (Xie, Sato, 2017).

We present mean square errors with standard deviations as subscripts on each data set or the averaged MSE on all 10 data sets in following tables. In Tables 40-41, we denote better, significantly better, worse and significantly worse respectively as +, ++, - and --. Instances is the data sample size. Dimension is the original data space dimensionality. We typically take 60% data instances as training data. As we notice the performance of Random Forest and Quantum-Inspired Forest adapt to hyperparameters in similar patterns, we decide to study two Forests' performance in multiple settings of forest hyperparameters (Xie, Sato, 2017).

Data	Instances	Dimension	QI-Forest	R-Forest	+/-
Abalone	4177	8	0.3204 _{0.0055}	0.3350 _{0.0073}	++
Communities Crime	1994	122	0.2763 _{0.0025}	0.3016 _{0.0080}	++
Communities Crime Unnormalized 1	2215	140	0.2515 _{0.0053}	0.2766 _{0.0112}	++
Communities Crime Unnormalized 2	2215	140	0.2125 _{0.0052}	0.2697 _{0.0073}	++
Facebook Metrics	500	11	0.1580 _{0.0302}	0.1267 _{0.0480}	-
Forests Fire	517	8	0.8296 _{0.0175}	0.8369 _{0.0231}	+
Housing	505	13	0.2011 _{0.0089}	0.2492 _{0.0171}	++
Slump Test	103	9	0.1704 _{0.0103}	0.2678 _{0.0276}	++
Wine Quality Red	1599	11	0.4379 _{0.0060}	0.4622 _{0.0118}	++
Wine Quality White	4898	11	0.4056 _{0.0025}	0.4087 _{0.0075}	+

Table 40: Quantum-Inspired Forest Regressors vs. Random Forest Regressors: select one half features to train base learners, namely $a = 0.5$, ensemble size $T = 30$, training instances $N = 60\%$. Means square errors with standard deviations as subscripts are presented (Xie, Sato, 2017).

Data	Instances	Dimension	QIE-LR	RE-LR	+/-
Abalone	4177	8	0.3466 _{0.0061}	0.4186 _{0.0207}	++
Communities Crime	1994	122	0.2398 _{0.0021}	0.3220 _{0.0275}	++
Communities Crime Unnormalized 1	2215	140	0.0213 _{0.0001}	0.1935 _{0.0226}	++
Communities Crime Unnormalized 2	2215	140	0.1104 _{0.0022}	0.2389 _{0.0202}	++
Facebook Metrics	500	11	0.0044 _{0.0004}	0.0675 _{0.0196}	++
Forests Fire	517	8	0.7298 _{0.0016}	0.7332 _{0.0029}	++
Housing	505	13	0.2695 _{0.0026}	0.3883 _{0.0247}	++
Slump Test	103	9	0.1075 _{0.0042}	0.2624 _{0.0446}	++
Wine Quality Red	1599	11	0.4764 _{0.0023}	0.4833 _{0.0103}	++
Wine Quality White	4898	11	0.5245 _{0.0010}	0.5334 _{0.0073}	++

Table 41: Quantum-Inspired Ensemble Linear Regressors vs. Random Ensemble Linear Regressors: select one half features to train base learners, namely $a = 0.5$, ensemble size T

$= 30$, training instances $N = 60\%$. Means square errors with standard deviations as subscripts are presented (Xie, Sato, 2017).

α	QI-Forest	R-Forest
0.125	0.4251 _{0.0154}	0.4932 _{0.0208}
0.25	0.3411 _{0.0082}	0.4186 _{0.0182}
0.5	0.3263 _{0.0094}	0.3544 _{0.0168}
0.75	0.3253 _{0.0099}	0.3313 _{0.0118}
1.0	0.3377 _{0.0095}	—

Table 42: Quantum Inspired Forest Regressors vs. Random Forest Regressors: ensemble size $T = 30$, training instances $N = 60\%$, adjust α respectively as 0.125, 0.25, 0.5, 0.75, 1.0. When $\alpha = 1.0$, Quantum Inspired Forest degenerates into Random Forest. Mean Square Errors averaged over 10 data sets are presented (Xie, Sato, 2017).

T	QI-Forest	R-Forest
3	0.4212 _{0.0313}	0.4758 _{0.0613}
10	0.3565 _{0.0219}	0.3888 _{0.0317}
30	0.3263 _{0.0094}	0.3534 _{0.0168}
100	0.3140 _{0.0046}	0.3356 _{0.0076}

Table 43: Quantum Inspired Forest Regressors vs. Random Forest Regressors: $\alpha = 0.5$, training instances $N = 60\%$, adjust ensemble size T respectively as 3, 10, 30, 100 (Xie, Sato, 2017).

Quantum Least Squares Regression:

In this section, we present our quantum approximation algorithm for Least Squares Regression (QLSR), then analyze its error rate and running time (Liu, Zhang, 2017).

Algorithm QLSR

Input: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. A is Hermitian with spectral decomposition $A = \sum_{i=1}^n \lambda_i |v_i\rangle\langle v_i|$, where all the eigenvalues $\lambda_1, \dots, \lambda_n$ satisfy $\frac{1}{\kappa} \leq |\lambda_i| \leq 1$ for $i = 1, \dots, r$ for some known value κ and $\lambda_i = 0$ for $i = r+1, \dots, n$. Suppose that $b = \sum_{i=1}^n \beta_i |v_i\rangle$.

Output: A quantum state proportional to $|\tilde{x}\rangle$ where $\tilde{x} \approx x^* \stackrel{\text{def}}{=} A^+ b$, and a value $\ell \approx \|x^*\|_2^2$.

Algorithm:

1. Prepare the quantum state $|b\rangle = \frac{1}{\|b\|_2} \sum_{i=1}^n \beta_i |v_i\rangle$.
2. Perform phase estimation to create the state $\frac{1}{\|b\|_2} \sum_{i=1}^n \beta_i |v_i\rangle |\tilde{\lambda}_i\rangle$, where $\tilde{\lambda}_i$ is the estimated value of λ_i satisfying $|\tilde{\lambda}_i - \lambda_i| \leq \delta_{\text{PE}} \stackrel{\text{def}}{=} \frac{\epsilon}{2\kappa}$ for $i = 1, \dots, n$.
3. Add a qubit $|0\rangle$ to the state and perform a controlled rotation as follows. If $\tilde{\lambda}_i \geq \frac{1}{2\kappa}$, rotate the qubit to $(\frac{1}{2\kappa\tilde{\lambda}_i}|1\rangle + \sqrt{1 - \frac{1}{4\kappa^2\tilde{\lambda}_i^2}}|0\rangle)$; otherwise do nothing. The resulting state is

$$\frac{1}{\|b\|_2} \sum_{i=1}^r \beta_i |v_i\rangle |\tilde{\lambda}_i\rangle \left(\frac{1}{2\kappa\tilde{\lambda}_i} |1\rangle + \sqrt{1 - \frac{1}{4\kappa^2\tilde{\lambda}_i^2}} |0\rangle \right) + \frac{1}{\|b\|_2} \sum_{i=r+1}^n \beta_i |v_i\rangle |\tilde{\lambda}_i\rangle |0\rangle. \quad (6)$$

4. Use amplitude amplification by repeating the previous steps $O(\kappa^2/\epsilon)$ times.
5. Measure the last qubit.
6. if we observe $|1\rangle$,
 - (a) The remaining state is proportional to $\sum_{i=1}^r \frac{\beta_i}{\tilde{\lambda}_i} |v_i\rangle |\tilde{\lambda}_i\rangle$.
 - (b) Reverse the phase estimation process and get the state proportional to $\sum_{i=1}^r \frac{\beta_i}{\tilde{\lambda}_i} |v_i\rangle = |\tilde{x}\rangle$ as our output.
- else output 0 as an estimate to $|x^*\rangle$.
7. Use amplitude estimation to get an estimate p' to the probability p of observing $|1\rangle$ when measuring the state in Eq.(6), to precision $\delta = \epsilon/(4\kappa^2)$ and with success probability 0.99. Output $\ell = p' \cdot 4\|b\|_2^2 \kappa^2$.

We assume that $A \in \mathbb{R}^{n \times n}$ with rank r is Hermitian and $b \in \mathbb{R}^n$. Our goal is to compute $x^* = A^+ b$. We analyze the precision, error probability and the cost. For convenience, we summarize the parameters: the phase estimation error

$\delta_{\text{PE}} = \frac{\epsilon}{2\kappa}$, the Hamiltonian simulation error $\delta_{\text{HS}} = O(\delta_{\text{PE}}^2) \cdot O\left(\frac{\epsilon}{\kappa^2}\right)$ and last-step measurement precision $\delta = \frac{\epsilon}{4\kappa^2}$.

We analyze the quality of the solution: $|\tilde{x}\rangle$. With probability at least 0.99, the outputted vector \tilde{x} satisfies the inequality:

$$\|\tilde{x} - x^*\|_2 \leq \epsilon \cdot \max\{\|x^*\|_2, \|b\|_2\} \quad (5.12)$$

Next we analyze the estimated norm. With probability at least 0.99, the outputted value ℓ satisfies the inequality:

$$|\ell - \|x^*\|_2^2| \leq \epsilon (\|x^*\|_2^2 + \|b\|_2^2) \quad (5.13)$$

Finally, we analyze the computational cost:

For Step 1 of the QLSR algorithm, we can efficiently prepare $|b\rangle$ in time $O(\log n)$ provided that b_i ($i = 1, \dots, n$) and $\sum_{i_1}^{i_2} |b_i|^2$ ($1 \leq i_1 < i_2 \leq n$) are efficiently computable.

For Step 2 we perform quantum phase estimation by simulating e^{iA} , which takes time $O(s(\log n + \text{poly} \log(s, \kappa)))$.

We want the error of the eigenvalue estimation is at most $\delta_{PE} = \frac{\varepsilon}{2\kappa}$, so the phase estimation algorithm needs $O(\frac{\kappa}{\varepsilon})$ calls of e^{iA} simulation. Thus, the total time for one phase estimation is $O(s(\log n + \text{poly} \log(s, \kappa)) \kappa / \varepsilon)$.

Repeating this for $O(\frac{\kappa^2}{\varepsilon})$ time in Step 4 needs time $O(s(\log n + \text{poly} \log(s, \kappa)) \kappa^3 / \varepsilon^2)$.

Therefore, if we do not need to estimate the norm $\|x^*\|_2$, then the algorithm can just stop before Step 7. The total time cost is $O\left((\log n) \cdot s^2 \cdot \frac{\kappa}{\varepsilon} \cdot \frac{\kappa^2}{\varepsilon}\right) = O((\log n) s^2 \kappa^3 / \varepsilon^2)$.

If we want to estimate the norm $\|x^*\|_2$, the Amplitude Estimation needs to repeat Step 1 to Step 3 at most $O\left(\frac{1}{\delta}\right) = O\left(\frac{\kappa^2}{\varepsilon}\right)$ times. So the total cost is

$$O\left((\log n) \cdot s^2 \cdot \frac{\kappa}{\varepsilon} \cdot \frac{\kappa^2}{\varepsilon}\right) = O((\log n) s^2 \kappa^3 / \varepsilon^2) \quad (\text{Liu, Zhang, 2017}).$$

Quantum Linear Regression Algorithm from classical data set:

The quantum linear regression algorithm is based on the method of encoding classical information such as a 2^n dimensional vector $\mathbf{a} = (a_0, \dots, a_{2^n-1})^T$ into the 2^n amplitudes a_0, \dots, a_{2^n-1} of a n -qubit quantum system: $|\psi_a\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$, where $\{|i\rangle\}$ is a convenient notation for the computational basis: $\{|0 \dots 0\rangle \triangleq |0\rangle, \dots, |1 \dots 1\rangle \triangleq |2^n - 1\rangle\}$ (Schuld, Sinayskiy, Petruccione, 2016).

In other words, the probabilistic description of a set of 2-level quantum systems is used to store and manipulate classical information. We refer to this method here as amplitude encoding and denote every such quantum state by a $|\psi\rangle$ with a subscript referring to the classical vector it encodes. Namely $|\alpha\rangle$ is a quantum state where some mathematical object α is encoded into the basis state in a way that is to specify in detail, while $|\psi_\alpha\rangle$ is a quantum state representing the real vector α via amplitude encoding (Schuld, Sinayskiy, Petruccione, 2016).

The strategy for the quantum algorithm is the following. In Step 1 the general idea of the quantum pattern recognition algorithm is to create a quantum state representing the data matrix $\mathbf{X} = \sum_r \sigma_r \mathbf{u}_r \mathbf{v}_r^T$ via amplitude encoding. In Steps 2 and 3 we use tricks to invert the unknown singular values efficiently. In Step 4, quantum state representations of $\mathbf{y}, \tilde{\mathbf{x}}$ are used to write the desired prediction from Equation:

$$\tilde{\mathbf{y}} = \sum_{r=1}^R \sigma_r^{-1} \tilde{\mathbf{x}}^T \mathbf{v}_r \mathbf{u}_r^T \mathbf{y} \quad (5.14)$$

into the off-diagonal elements of an ancilla qubit, where it can be read out by a simple σ_x, σ_y measurement (Schuld, Sinayskiy, Petruccione, 2016).

Step 1: State Preparation

The quantum algorithm takes copies of the quantum states representing each of the objects \mathbf{X}, \mathbf{y} and $\tilde{\mathbf{x}}$ from above in amplitude encoding:

$$|\psi_X\rangle = \sum_{j=0}^{N-1} \sum_{m=0}^{M-1} x_j^{(m)} |j\rangle |m\rangle \quad (5.15)$$

$$|\psi_y\rangle = \sum_{\mu=0}^{M-1} y^{(\mu)} |\mu\rangle \quad (5.16)$$

$$|\psi_{\tilde{x}}\rangle = \sum_{\gamma=0}^{N-1} \tilde{x}_\gamma |\gamma\rangle \quad (5.17)$$

$$\text{with } \sum_{m,j} |x_j^{(m)}|^2 = \sum_{\mu} |y^{(\mu)}|^2 = \sum_{\gamma} |\tilde{x}_\gamma|^2 = 1.$$

We note that the algorithm thus works with normalized data and the results have to be re-scaled accordingly. Using the Gram-Schmidt decomposition, we can formally write:

$$|\psi_X\rangle = \sum_{r=1}^R \sigma^r \sum_{j=1}^J v_j^r |j\rangle \sum_{m=1}^M u_m^r |m\rangle = \sum_{r=1}^R \sigma^r |\psi_{v^r}\rangle |\psi_{u^r}\rangle \quad (5.18)$$

We state $|\psi_{v^r}\rangle = \sum_{j=1}^J v_j^r |j\rangle$ and $|\psi_{u^r}\rangle = \sum_{m=1}^M u_m^r |m\rangle$ are quantum states representing the orthogonal sets of left and right singular vectors of X via amplitude encoding and σ^r are the corresponding singular values.

We acknowledge that overall, the question of state preparation is an outstanding challenge for quantum machine learning algorithm design. Our goal of the following algorithm is to remain linear in the number of qubits or logarithmic in the problem size. In this direction, we mention the number of qubits needed to construct states (5.15), (5.16) and (5.17) are $\lceil \log N \rceil + \lceil \log M \rceil$, $\lceil \log M \rceil$ and $\lceil \log N \rceil$ respectively. Generally, if given a “classical” data set, techniques for the efficient preparation of arbitrary initial quantum states are a nontrivial and controversially discussed topic. However, some ideas like the linear state preparation in the number of qubits or via Quantum Random Access Memory (QRAM) make their appearance to contribute in this efficient preparation.

Step 2: Extracting the singular values

We want to transform Equation (5.18) into a “quantum representation”, so firstly we invert the singular values of X . For this, we will “extract” the eigenvalues λ_r of $X^\dagger X$ to eigenvectors v_r and use the inversion procedure in the following step. In order to access the eigenvalues, we use copies of the state (5.15) in which on the level of description we ignore the $|m\rangle$ register in order to obtain a mixed state $\rho_{X^\dagger X} = \text{tr}_m \{ |\psi_X\rangle \langle \psi_X| \}$ which represents the positive Hermitian matrix $X^\dagger X$:

$$\rho_{X^\dagger X} = \sum_{j,j'=1}^N \sum_{m=1}^M x_j^{(m)} x_{j'}^{(m)*} |j\rangle \langle j'| \quad (5.19)$$

Now we use the ideas of Quantum Principal Component Analysis to “apply” $\rho_{X^\dagger X}$ to $|\psi_X\rangle$ resulting in:

$$\sum_{k=0}^K |k\Delta t\rangle \langle k\Delta t| \otimes e^{-ik\rho_{X^\dagger X}\Delta t} |\psi_X\rangle \langle \psi_X| e^{ik\rho_{X^\dagger X}\Delta t} \quad (5.20)$$

for some large K . The quantum phase estimation algorithm results in:

$$\sum_{r=1}^R \sigma^r |\psi_{v^r}\rangle |\psi_{u^r}\rangle |\lambda^r\rangle \quad (5.21)$$

in which the eigenvalues $\lambda^r = (\sigma^r)^2$ of $\rho_{X^\dagger X}$ are approximately encoded in the τ qubits of an extra third register that was initially in the ground state.

Step 3: Inverting the singular values

If we add an extra qubit and rotate it conditional on the eigenvalue register, then we have:

$$\sum_{r=1}^R \sigma^r |\psi_{v^r}\rangle |\psi_{u^r}\rangle |\lambda^r\rangle \left(\sqrt{1 - \left(\frac{c}{\lambda^r}\right)^2} |0\rangle + \frac{c}{\lambda^r} |1\rangle \right) \quad (5.22)$$

The constant c is chosen so that the inverse eigenvalues are not larger than 1, which is given if it is smaller than the smallest nonzero eigenvalue λ^{\min} of $X^\dagger X$ or equivalently the smallest nonzero squared singular value $(\sigma^{\min})^2$ of X . We perform a conditional measurement on the ancilla qubit, only continuing the algorithm (“accepting”) if the ancilla is in state $|1\rangle$, else the entire procedure has to be repeated. In next step about runtime analysis we discuss about the amplitude amplification and how it can boost the probability of accepting quadratically.

Uncomputing and discarding the eigenvalue register results in:

$$|\psi_1\rangle = \frac{1}{\sqrt{p(1)}} \sum_{r=1}^R \frac{c}{\sigma^r} |\psi_{v^r}\rangle |\psi_{u^r}\rangle \quad (5.23)$$

where the probability of acceptance is given by: $p(1) = \sum_r \left| \frac{c}{\lambda^r} \right|^2$.

Step 4: Executing the inner products

The last step has the goal to write the desired result:

$\sum_{r=1}^R (\sigma^r)^{-1} \langle \psi_{\tilde{x}} | v^r \rangle \langle \psi_y | u^r \rangle$ into selected entries of an ancilla’s single qubit density matrix, from which it can be accessed by a simple measurement. We consider the result (5.23) of the previous step, as well as $|\psi_2\rangle = |\psi_y\rangle |\psi_{\tilde{x}}\rangle$ from Equations (5.16) and (5.17).

We conditionally prepare the two states so that they are entangled with an ancilla qubit:

$$\frac{1}{\sqrt{2}} (|\psi_1\rangle |0\rangle + |\psi_2\rangle |1\rangle) \quad (5.24)$$

and trace out all registers except from the ancilla, the off-diagonal elements ρ_{12}, ρ_{21} of the ancilla's density matrix read:

$$\frac{c}{2\sqrt{p(1)}} \sum_r (\sigma^r)^{-1} \sum_j v_j^r \tilde{x}_j \sum_m u_m^r y^{(m')} \quad (5.25)$$

and contain the desired result up to a known normalization factor.

Conditionally preparing (5.24) requires us to execute the entire algorithm including state preparation conditioned on the state of the ancilla qubits and might not be easy to implement. In that case one can adapt the algorithm so that the $|0 \dots 0\rangle$ basis state in $|\psi_1\rangle$ and $|\psi_2\rangle$ is “excluded” from all operations and remains with a constant amplitude $\frac{1}{\sqrt{2}}$ throughout the algorithm, while the other $2^n - 1$ amplitudes are renormalized accordingly. This prepares states of the general form $|a\rangle = \frac{1}{\sqrt{2}} (|0 \dots 0\rangle + \sum_{i=1}^N a_i |i\rangle)$, $|b\rangle = \frac{1}{\sqrt{2}} (|0 \dots 0\rangle + \sum_{j=0}^N b_j |j\rangle)$.

A common swap test effectively shifts the inner product by $1/2$ and thus reveals

$$|\langle a|b\rangle|^2 = \left| \frac{1}{2} + \frac{1}{2} \sum_{i=1}^N a_i b_i \right|^2 \text{ from which the sign of } \sum_{i=1}^N a_i b_i \text{ can be extracted}$$

(Schuld, Sinayskiy, Petruccione, 2016).

These are the basic steps of the algorithm. Now we describe briefly the runtime analysis and the computational complexity of the above steps of the algorithm. According to previous researches we need temporal resources $t = k\Delta t$ in $O(\log N)$ and of the order $O(\varepsilon^{-3})$ copies of ρ_{X+X} to “exponentiate” a density matrix in Step 2, where ε is the error and N the dimension of the inputs in our data set. The method requires the density matrix ρ_{X+X} to be close to a low-rank approximation, which is dominated by a few large eigenvalues in order to maintain the exponential speedup. In general, it takes time $t = O\left(\frac{1}{\delta}\right)$ to simulate e^{iHt} for a Hamiltonian H up to error δ and it takes time t^2 to do the same for $e^{i\rho t}$ (Schuld, Sinayskiy, Petruccione, 2016).

This means that if we want to resolve relatively uniform eigenvalues of the order of $1/N$, time grows quadratically with N and the exponential speedup is lost. Hence, the method is only efficient if the density matrix is dominated by a few large eigenvalues.

The singular value inversion procedure in Step 3 determines the runtime's dependency on the condition number of \mathbf{X} , $\kappa = \sigma^{\max} (\sigma^{\min})^{-1}$. The probability to measure the ancilla in the excited state is:

$$p(1) = \sum_r \left| \frac{c}{\lambda^r} \right|^2 \leq R \left| \frac{\lambda^{\min}}{\lambda^{\max}} \right|^2 = R\kappa^{-4} \quad (5.26)$$

which means we need on average less than κ^4 tries to accept the conditional measurement (Schuld, Sinayskiy, Petruccione, 2016).

The amended SWAP routine in Step 4 is also linear in the number of qubits and the final measurement only accounts for a constant factor. The upper bound for the runtime can thus be roughly estimated as $O(\log N \kappa^2 \varepsilon^{-3})$ if we have sufficient copies of $\rho_{X^\dagger X}$ available which is required to be close to a low-rank matrix. We have to remember that this does not include the costs of quantum state preparation, in case the algorithm processes classical information. Our algorithm tackles the problem of pattern recognition or prediction and it can be applied efficiently to non-sparse, but low rank approximations of the matrix $\mathbf{X}^\dagger \mathbf{X}$ (Schuld, Sinayskiy, Petruccione, 2016).

Therefore, in summary we described an algorithm for a universal quantum computer to implement a linear regression model for supervised pattern recognition. This quantum algorithm reproduces the prediction result of a classical linear regression method with least squares optimization. It runs in time logarithmic in the dimension N of the feature vectors as well as independent of the size of the training set if the inputs are given as quantum information. Instead of requiring the matrix containing the training inputs, \mathbf{X} , to be sparse it merely needs $\mathbf{X}^\dagger \mathbf{X}$ to be representable by a low-rank approximation. However, the sensitive dependency on the accuracy as well as the unresolved problem of state preparation illustrate how careful we need to treat “magic” exponential speedups for pattern recognition (Schuld, Sinayskiy, Petruccione, 2016).

In this section we describe the theory and the details of the Quantum

Regression Algorithm, which we use below for our novel application in Python. The algorithm merges two areas, Quantum Computing and Machine Learning and belongs to the category of Quantum Circuit Learning (QCL). The Quantum Circuit Learning is a quantum/classical hybrid algorithm that aims to perform supervised or unsupervised learning tasks (Kopczyk, 2018).

Quick explanation of the algorithm:

The QCL is based on the idea of variational circuits equivalent to transformation $U(x, \theta)$. By repetitive measurement of qubits after such transformation, we can estimate an expectation value, which is expressed as a function $f(x, \theta)$. By proper tuning of θ parameters the quantum circuit learns to output some label y , so that a loss $L(y, f(x, \theta))$ is minimized (Kopczyk, 2018).

Variational Quantum Circuit is just a unitary transformation that is θ -parameterized (θ is a vector) and can be divided into a sequence of smaller unitaries:

$$U(x, \theta) = U_J(\theta_J) \dots U_j(\theta_j) \dots U_1(\theta_1) U_0(x) \quad (5.27)$$

where $U_0(x)$ encodes an input data x into a quantum state. The key insight is that θ parameters can be adjusted so that the variational circuit produces the desired output. In supervised tasks, QCL is supplied with training data $\{x_i\}$ and corresponding labels $\{y_i\}$ for $i = 1, 2, \dots, M$, where M is a number of samples. Then, the algorithm learns to output $f(x_i, \theta)$ in a way it is as close as possible to y_i .

We describe below the steps of the algorithm for N -qubits circuit (Kopczyk, 2018).

Step 1: Encode input data x_i into a quantum state by applying some unitary transformation $U(x_i)$ to initialized qubits $|0\rangle^{\otimes N}$.

Step 2: Apply θ -parameterized unitary on the encoded input state, generate an output state and measure some observable B . As an observable we use a subset of Pauli operators $\{B\} \subset \{I, X, Y, Z\}^{\otimes N}$ (for example Z measured on the first qubit).

Step 3: Repeat Step 1 and Step 2 P -times to get an estimate of the expectation value $f(x_i, \theta)$ of some chosen observable B .

Step 4: Repeat Step 3 for each sample and calculate a loss $L(y, f(x, \theta))$. Minimize it by tuning θ parameters using classical optimization algorithm such as gradient descent.

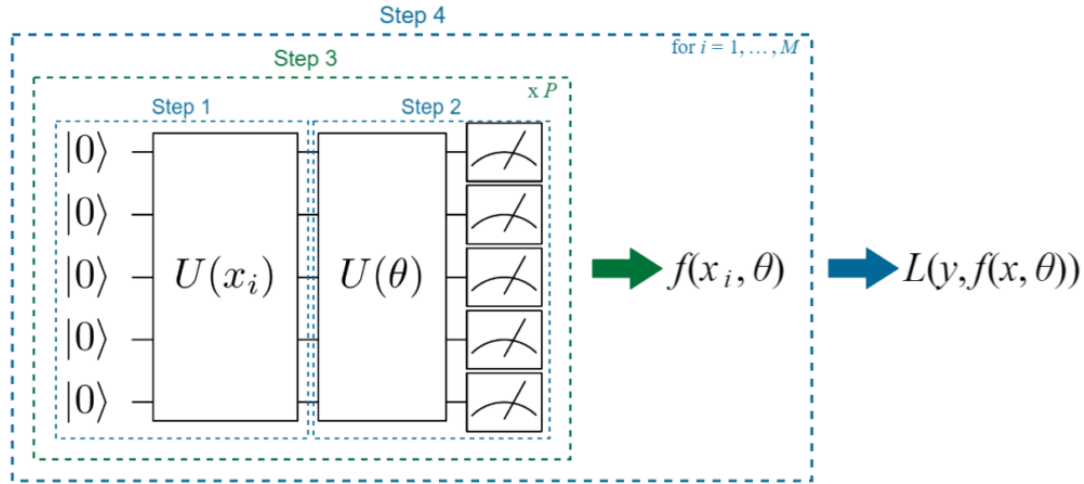


Figure 37: The quantum part of the algorithm amounts to encoding an input data into a quantum state, applying a theta-parameterized unitary and calculating expectation values for each training sample (Kopczyk, 2018).

In this hybrid algorithm, a quantum subroutine calculates the output $f(x_i, \theta)$ for each sample, whereas the calculation of loss and optimization of θ parameters is executed in a classical loop (Kopczyk, 2018).

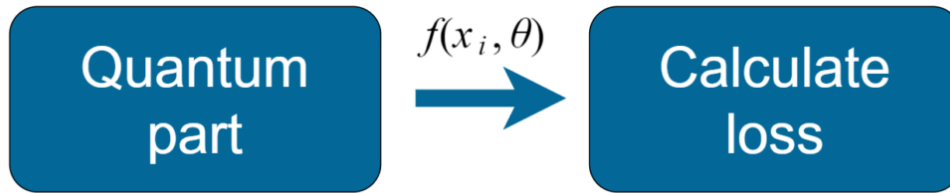
update θ 

Figure 38: Learning in QCL involves iterative execution of quantum and classical parts of the algorithm (Kopczyk, 2018).

In case of larger parameter space, which is most commonly encountered in machine learning, the preferred optimization methods are gradient-based. This, in turn, requires us to calculate a derivative of $f(x, \theta)$. So we want to calculate the derivative of the expectation value with respect to a circuit parameter $\nabla_{\theta} f(x, \theta)$, but in many cases we can express it as a linear combination of the same quantum functions, differing only in a shift ρ in parameter θ (Kopczyk, 2018).

$$\nabla_{\theta} f(x, \theta) \propto f(x, \theta + \rho) - f(x, \theta - \rho)$$

Figure 39: In many cases, the gradient can be expressed or approximated by a linear combination of expectation values with a shift in the parameter (Kopczyk, 2018).

We present here a simple example of single parameterized gate:

$$U(x, \theta_1) = U_1(\theta_1) U_0(x) \quad (5.28)$$

$$\text{with } U_1(\theta_1) = R_X(\theta_1) = e^{-i\frac{\theta_1}{2}X} \quad (5.29)$$

where X is a Pauli operator. The gradient of this unitary and its conjugate transpose is expressed by:

$$\begin{aligned} \nabla_{\theta_1} U_1(\theta_1) &= -i\frac{\theta_1}{2} X U_1(\theta_1) \\ \nabla_{\theta_1} U_1^\dagger(\theta_1) &= i\frac{\theta_1}{2} U_1^\dagger(\theta_1) X \end{aligned} \quad (5.30)$$

as $X^\dagger = X$. The expectation value of measured operator B is defined as:

$$f(x, \theta_1) = \langle 0 | U_0^\dagger(x) U_1^\dagger(\theta_1) B U_1(\theta_1) U_0(x) | 0 \rangle = \langle x | U_1^\dagger(\theta_1) B U_1(\theta_1) | x \rangle \quad (5.31)$$

and the gradient is:

$$\nabla_{\theta_1} f(x, \theta_1) = \langle x | \nabla_{\theta_1} (U_1^\dagger(\theta_1) B U_1(\theta_1)) | x \rangle \quad (5.32)$$

Substituting the derivatives from Equation (5.37) we get:

$$\nabla_{\theta_1} f(x, \theta_1) = \frac{i}{2} \langle x | U_1^\dagger(\theta_1) (XB - BX) U_1(\theta_1) | x \rangle = \frac{i}{2} \langle x | U_1^\dagger(\theta_1) [X, B] U_1(\theta_1) | x \rangle \quad (5.33)$$

where $[X, Y] = XY - YX$ is the commutator. We use the following property of commutator involving Pauli operators, in particular X , and an arbitrary operator B :

$$[X, B] = -i \left(U_1^\dagger\left(\frac{\pi}{2}\right) B U_1\left(\frac{\pi}{2}\right) - U_1^\dagger\left(-\frac{\pi}{2}\right) B U_1\left(-\frac{\pi}{2}\right) \right) \quad (5.34)$$

Thus, we obtain:

$$\begin{aligned} \nabla_{\theta_1} f(x, \theta_1) &= \frac{1}{2} \left\langle x \left| U_1^\dagger\left(\theta_1 + \frac{\pi}{2}\right) B U_1\left(\theta_1 + \frac{\pi}{2}\right) \right| x \right\rangle - \\ &\frac{1}{2} \left\langle x \left| U_1^\dagger\left(\theta_1 - \frac{\pi}{2}\right) B U_1\left(\theta_1 - \frac{\pi}{2}\right) \right| x \right\rangle \end{aligned} \quad (5.35)$$

which is just:

$$\nabla_{\theta_1} f(x, \theta_1) = \frac{1}{2} \left(f\left(x, \theta_1 + \frac{\pi}{2}\right) - f\left(x, \theta_1 - \frac{\pi}{2}\right) \right) \quad (5.36)$$

This is a linear combination of the same quantum functions, but with shift $\rho = \pm \frac{\pi}{2}$ in the parameter! So to calculate the gradient we need to additionally run Step 1 – Step 3 of our algorithm twice with the shifted parameters.

In case of multiple parameterized gates, the same logic applies. To calculate the gradient, one has to modify the variational circuit by inserting $\pm \frac{\pi}{2}$ rotations next to the θ_j -dependent unitary $U_j(\theta_j)$ (Kopczyk, 2018).

Regression Example in Python:

In this example, QCL will try to learn a simple quadratic function x^2 . Firstly, we generate a (very) small data set $x = \{x_1, \dots, x_8\}$ with corresponding labels $y = \{y_1, \dots, y_8\}$:

```
import numpy as np

np.random.seed(0)
m = 8
X = np.linspace(-0.95,0.95,m)
y = X**2
```

Next we implement $U_0(x_i)$ unitary to encode input data x_i into a quantum state. We will do that by applying the following qubit rotations:

$$U_0(x_i) = \prod_{k=1}^N R_X^k(\cos(x_i^2)) R_Y^k(\sin(x_i)) \quad (5.37)$$

to initialized state $|0\rangle^{\otimes N}$ with $N = 3$ expressing a number of qubits in quantum circuit and R_X^k rotating k -th qubit (Kopczyk, 2018).

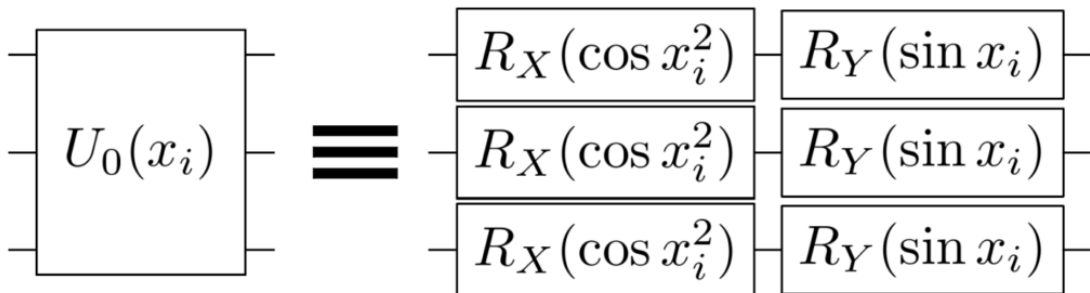


Figure 40: The unitary encoding input data is decomposed to a set of qubit rotations depending on sample value (Kopczyk, 2018).



We code “input_prog” that returns an instance of “Program” object with calculated $U_0(x_i)$ unitary (you can access the unitary with “gate” property). The “inst” method applies a gate on k-th qubit:

```
1 import qsimulator as pq
2 from qsimulator import RX, RY, RZ
3 n_qubits = 3
4 def input_prog(sample):
5     p = pq.Program(n_qubits)
6     for j in range(n_qubits):
7         p.inst(RY(np.arcsin(sample[0])), j)
8         p.inst(RZ(np.arccos(sample[0]**2)), j)
9     return p
```

Then, we implement θ -parameterized gate. Generally, $U(\theta)$ should create a highly entangled output state so that the complex function can be efficiently learned. The output state is generated by evolving a quantum system accordingly to Hamiltonian of fully connected transverse Ising model and then use θ -parameterized qubit rotations:

$$U_j^k(\theta_j^k) = R_X(\theta_{j,2}^k) R_Z(\theta_{j,1}^k) R_X(\theta_{j,0}^k) \quad (5.38)$$

on all $k=1, \dots, N$ qubits. This procedure is repeated D times to increase the learning capacity of the QCL algorithm. The Ising model Hamiltonian is expressed by:

$$H = \sum_{k=1}^N h_k X_k + \sum_{k=1}^N \sum_{m=1}^{k-1} J_{km} Z_k Z_m \quad (5.39)$$

with X, Z being Pauli operators and coefficients h_k, J_{km} can be taken randomly from uniform distribution on $[-1, 1]$. The evolution of the Hamiltonian is e^{-iTH} and describes how interaction behaves in trapped ions or superconducting qubits, thus it is easy to implement on quantum computers. However, it is not straightforward how to emulate it on the classical machine, i.e. how to perform exponentiation of a matrix with non-commuting terms (Kopczyk, 2018).

In our example we use a helper function to generate this unitary with Trotter-Suzuki approximation (we fix $T=10$):

```
1 from qcl import ising_prog_gen
2 ising_prog = ising_prog_gen(trotter_steps=1000, T=10, n_qubits=n_q)
```

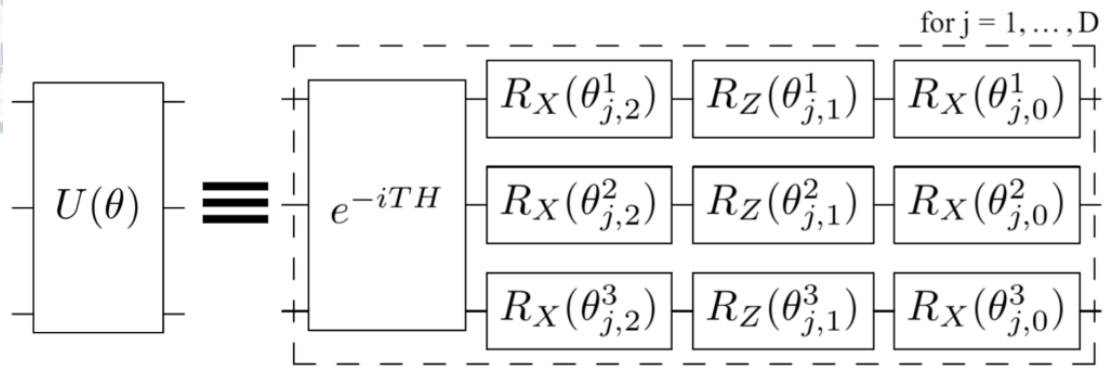


Figure 41: The theta-parameterized unitary is decomposed into a set of qubit rotations depending on adjustable parameters and entangling unitary represented by dynamics of the Ising model (Kopczyk, 2018).

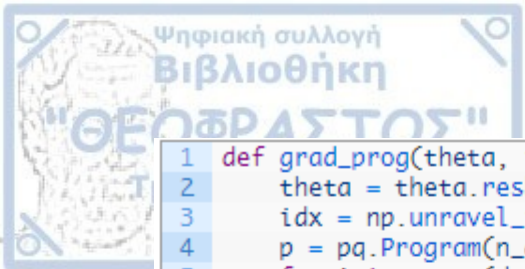
The output state is generated with the following function (D is denoted here by depth variable):

```

1 depth = 3
2 def output_prog(theta):
3     p = pq.Program(n_qubits)
4     theta = theta.reshape(3,n_qubits,depth)
5     for i in range(depth):
6         p += ising_prog
7         for j in range(n_qubits):
8             rj = n_qubits-j-1
9             p.inst(RX(theta[0,rj,i]), j)
10            p.inst(RZ(theta[1,rj,i]), j)
11            p.inst(RX(theta[2,rj,i]), j)
12    return p

```

We also need a function generating a variational circuit responsible for gradient calculations. As explained in the previous section, this is done by inserting $\pm \frac{\pi}{2}$ rotations next to the θ_j -dependent unitary $U_j(\theta_j)$:



```

1 def grad_prog(theta, idx, sign):
2     theta = theta.reshape(3,n_qubits,depth)
3     idx = np.unravel_index(idx, theta.shape)
4     p = pq.Program(n_qubits)
5     for i in range(depth):
6         p += ising_prog
7         for j in range(n_qubits):
8             rj = n_qubits-j-1
9             if idx == (0,rj,i):
10                 p.inst(RX(sign*np.pi/2.0), j)
11                 p.inst(RX(theta[0,rj,i]), j)
12             if idx == (1,rj,i):
13                 p.inst(RZ(sign*np.pi/2.0), j)
14                 p.inst(RZ(theta[1,rj,i]), j)
15             if idx == (2,rj,i):
16                 p.inst(RX(sign*np.pi/2.0), j)
17                 p.inst(RX(theta[2,rj,i]), j)
18     return p

```

Now, it is time to run QCL. We initialize θ parameters with random numbers drawn from uniform distribution on $[0, 2\pi]$. Note that the total number of parameters is equal to $3*N*D$. The $f(x_i, \theta)$ is taken from Z expectation value on the first qubit and we use mean squared error as a loss function minimized. A number of training iterations (epochs) is set to 20 and we use full-batch gradient descend. Additionally, the expectation value is multiplied by a coefficient α which is also tuned, however, this is done inside the code. The aim of this multiplication is to keep the correct scale of the expectation value (Kopczyk, 2018).

The programs that defines $U_0(x_i)$, $U(\theta)$ and $\nabla_{\theta} U(\theta)$ are passed by a dictionary with “input”, “output”, “grad” keys respectively:

```

1 from qsimulator import Z
2 from qcl import QCL
3
4 state_generators = dict()
5 state_generators['input'] = input_prog
6 state_generators['output'] = output_prog
7 state_generators['grad'] = grad_prog
8
9 initial_theta = np.random.uniform(0.0, 2*np.pi, size=3*n_qubits*d)
10
11 operator = pq.Program(n_qubits)
12 operator.inst(Z, 0)
13 operator_programs = [operator]
14 est = QCL(state_generators, initial_theta, loss="mean_squared_err",
15           operator_programs=operator_programs, epochs=20, batch_size=1,
16           verbose=True)

```

We fit a QCL estimator to our training data and labels, extract the results for inspection and predict to produce a plot (Kopczyk, 2018).

```

1 est.fit(X,y)
2 results = est.get_results()
3
4 X_test = np.linspace(-1.0,1.0,50)
5 y_pred = est.predict(X_test)

```

As we see the QCL fits nicely to the data. Assuming, we have a low-noise quantum computer, we can increase significantly a number of qubits, depth and a number of samples to deal with more complex regression tasks (Kopczyk, 2018).

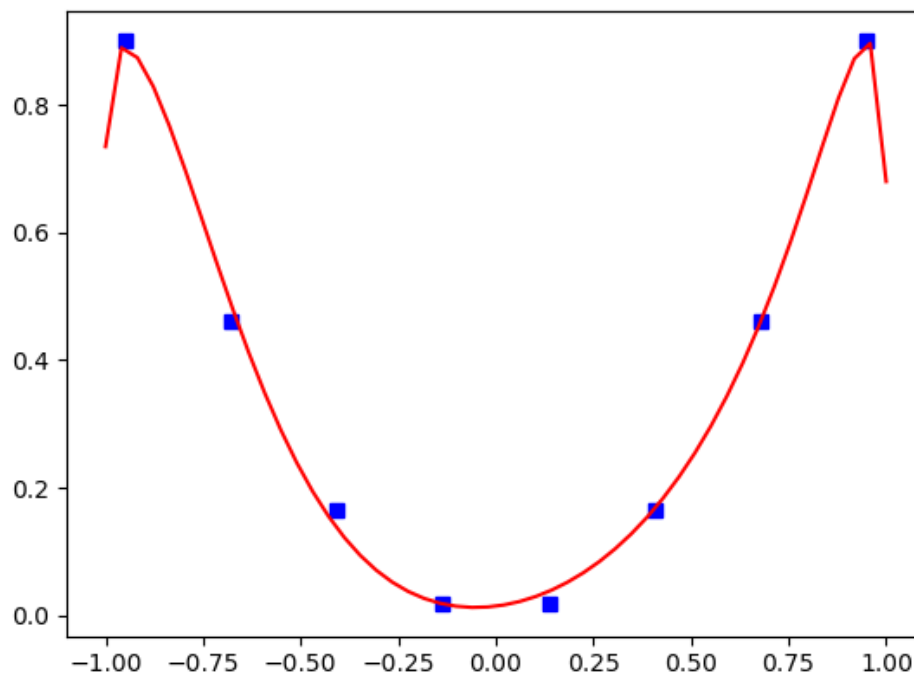


Figure 42: QCL regression results (Kopczyk, 2018).

The history of mean-squared error is presented on the chart below:

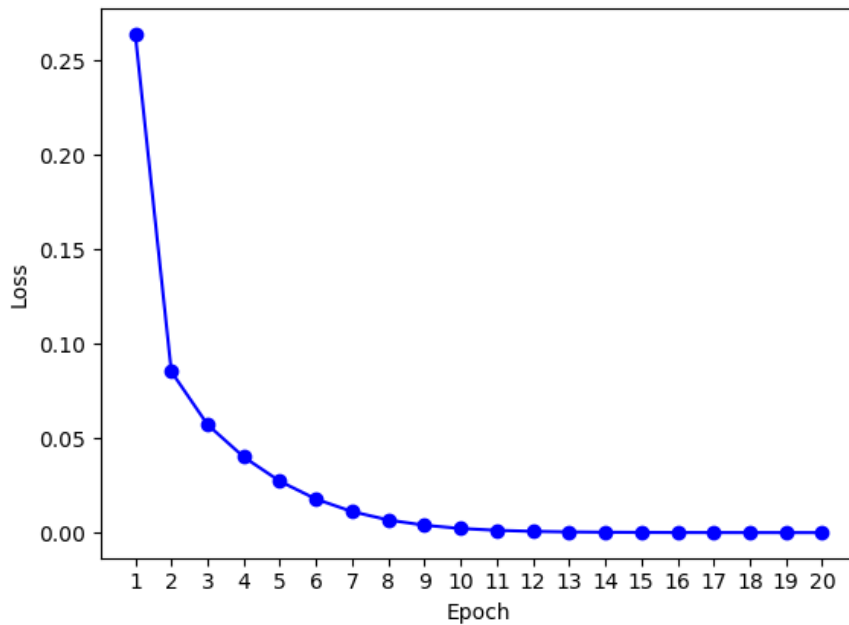


Figure 43: The loss converges after 20 epochs (Kopczyk, 2018).

Benefits and Limitations:

From one side, one can claim that we have just learned θ -parameterized matrices in a complex way to perform the simple regression task and that is the triumph of form over content. However, this algorithm is meant to be executed on the quantum processing devices. Generally, there exist methods to encode classical data of size 2^N into N qubits. Therefore, a huge data that cannot be handled on the classical machine, may be effectively manipulated on the quantum computer due to the exponential advantage (Kopczyk, 2018).

From the other side, the learning costs can kill the quantum advantage. To perform gradient descend in our regression example, for each out of $3 \cdot D \cdot N$ parameters we need to prepare and measure two quantum circuits and then this procedure has to be repeated in each epoch. Furthermore, the estimation of expectation values would require that the circuits are prepared from scratch and measured a sufficient number of times. In today's world of gate noise existence, it makes the QCL hardly possible to implement experimentally for complex tasks. It is

expected that in the future the quantum computers would perform low-noise gate operations so that the power of QCL can be utilized (Kopczyk, 2018).



QUANTUM REGRESSION : NOVEL APPLICATION

We construct the following example:

Data set Title: Student Performance Data Set

Abstract: Predict student performance in secondary education (high school)

Source: UCI Repository

Number of Instances: 649

Number of Attributes: 33

Download date: 10 February 2019

Data set Information: This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features) and it was collected by using school reports and questionnaires. Two data sets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por). The two data sets were modeled under binary/five-level classification and regression tasks. Important note: the target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful (see paper source for more details).

Attribute Information: Attributes for both student-mat.csv (Math course) and student-por.csv (Portuguese language course) data sets:

school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)

sex - student's sex (binary: 'F' - female or 'M' - male)

age - student's age (numeric: from 15 to 22)

address - student's home address type (binary: 'U' - urban or 'R' - rural)

famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)

Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)

Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - " 5th to 9th grade, 3 - " secondary education or 4 - " higher education)

Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - " 5th to 9th

grade, 3 - "secondary education or 4 - "higher education)

Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')

guardian - student's guardian (nominal: 'mother', 'father' or 'other')

traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)

studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

failures - number of past class failures (numeric: n if $1 \leq n < 3$, else 4)

schoolsup - extra educational support (binary: yes or no)

famsup - family educational support (binary: yes or no)

paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)

activities - extra-curricular activities (binary: yes or no)

nursery - attended nursery school (binary: yes or no)

higher - wants to take higher education (binary: yes or no)

internet - Internet access at home (binary: yes or no)

romantic - with a romantic relationship (binary: yes or no)

famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)

freetime - free time after school (numeric: from 1 - very low to 5 - very high)

goout - going out with friends (numeric: from 1 - very low to 5 - very high)

Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)

Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)

health - current health status (numeric: from 1 - very bad to 5 - very good)

absences - number of school absences (numeric: from 0 to 93)

these grades are related with the course subject, Math or Portuguese:

31 G1 - first period grade (numeric: from 0 to 20)

31 G2 - second period grade (numeric: from 0 to 20)

32 G3 - final grade (numeric: from 0 to 20, output target)

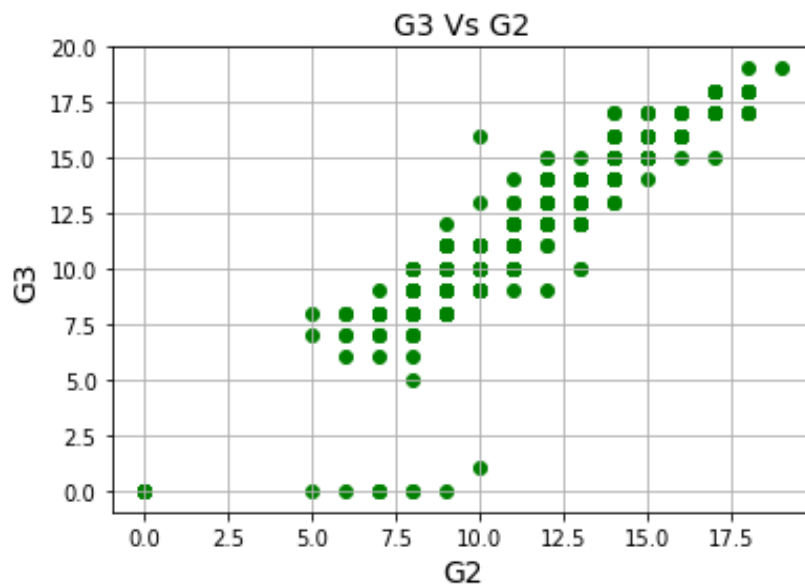
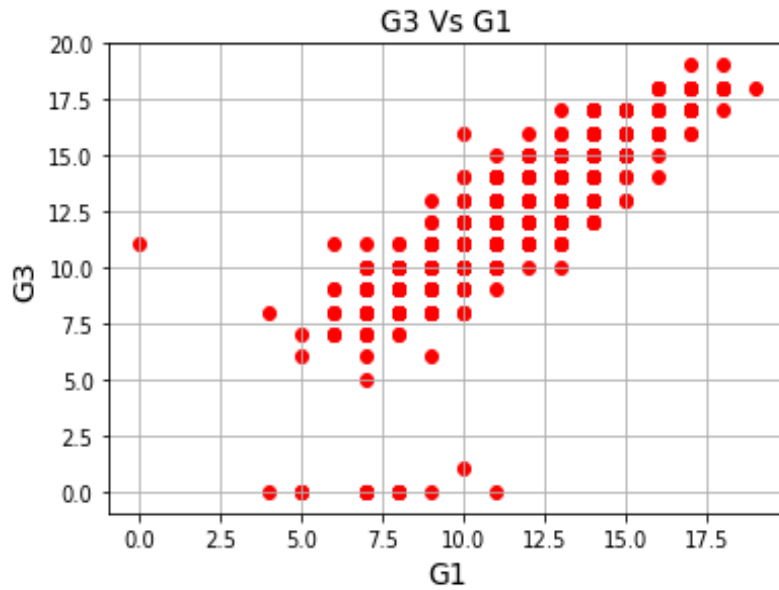


Multiple Linear Regression:

We provide the code of Multiple Linear Regression in appendices ([Appendix F](#)). Our aim is to predict the final grade of students. The output of Multiple Linear Regression is presented below.

Output:

	school	sex	age	add2ess	famsize	...	health	absences	G1	G2	G3
0	1	1	18	1	2	...	3	4	0	11	11
1	1	1	17	1	2	...	3	2	9	11	11
2	1	1	15	1	1	...	3	6	12	13	12
3	1	1	15	1	2	...	5	0	14	14	14
4	1	1	16	1	2	...	5	0	11	13	13
5	1	2	16	1	1	...	5	6	12	12	13
6	1	2	16	1	1	...	3	0	13	12	13
7	1	1	17	1	2	...	1	2	10	13	13
8	1	2	15	1	1	...	1	0	15	16	17
9	1	2	15	1	2	...	5	0	12	12	13
10	1	1	15	1	2	...	2	2	14	14	14
11	1	1	15	1	2	...	4	0	10	12	13
12	1	2	15	1	1	...	5	0	12	13	12
13	1	2	15	1	2	...	3	0	12	12	13
14	1	2	15	1	2	...	3	0	14	14	15
15	1	1	16	1	2	...	2	6	17	17	17
16	1	1	16	1	2	...	2	10	13	13	14
17	1	1	16	1	2	...	4	2	13	14	14
18	1	2	17	1	2	...	5	2	8	8	7
19	1	2	16	1	1	...	5	6	12	12	12
20	1	2	15	1	2	...	1	0	12	13	14
21	1	2	15	1	2	...	5	0	11	12	12
22	1	2	16	1	1	...	5	0	12	13	14
23	1	2	16	1	1	...	5	2	10	10	10
24	1	1	15	2	2	...	5	2	10	11	10
25	1	1	16	1	2	...	5	6	10	11	12
26	1	2	15	1	2	...	5	8	11	12	12
27	1	2	15	1	2	...	1	0	11	11	11
28	1	2	16	1	1	...	5	2	12	12	13
29	1	2	16	1	2	...	5	4	12	11	12
..
619	2	1	18	1	2	...	3	6	13	12	13
620	2	1	17	1	1	...	1	4	15	14	15
621	2	1	17	2	2	...	3	0	13	13	13
622	2	2	18	2	2	...	3	0	8	10	9
623	2	2	18	1	1	...	5	0	15	16	16
624	2	1	17	2	2	...	1	0	8	8	9
625	2	1	18	1	2	...	4	0	10	10	10
626	2	1	18	2	2	...	5	0	7	5	0
627	2	2	18	2	1	...	3	3	9	10	10
628	2	1	17	1	2	...	3	8	10	11	12
629	2	1	17	2	2	...	1	4	7	8	9
630	2	1	18	2	1	...	1	0	15	17	17
631	2	1	18	2	2	...	4	4	10	11	12
632	2	1	19	2	2	...	3	4	7	8	9
633	2	1	18	2	1	...	2	1	13	14	14
634	2	1	18	1	2	...	1	1	16	16	16
635	2	1	17	2	2	...	1	10	8	9	9
636	2	2	18	1	2	...	2	4	17	18	19
637	2	2	18	2	2	...	5	0	7	7	0
638	2	2	17	1	2	...	3	4	14	15	16
639	2	2	19	2	2	...	5	0	5	8	0
640	2	2	18	2	2	...	3	0	7	7	0
641	2	1	18	2	2	...	4	0	14	17	15
642	2	1	17	1	2	...	1	0	6	9	11
643	2	1	18	2	2	...	5	4	7	9	10
644	2	1	19	2	2	...	5	4	10	11	10
645	2	1	18	1	1	...	1	4	15	15	16
646	2	1	18	1	2	...	5	6	11	12	9
647	2	2	17	1	1	...	2	6	10	10	10
648	2	2	18	2	1	...	5	4	10	11	11



Intercept:
-0.1712830879869358
Coefficients:
[0.14889649 0.89714029]
Predicted G3:
[0.65847765]

OLS Regression Results

```

=====
Dep. Variable:          G3      R-squared:          0.848
Model:                  OLS      Adj. R-squared:       0.847
Method:                 Least Squares      F-statistic:       1799.
Date:                   Thu, 14 Mar 2019    Prob (F-statistic): 9.02e-265
Time:                   02:33:44           Log-Likelihood:    -1070.7
No. Observations:      649             AIC:              2147.
Df Residuals:          646             BIC:              2161.
Df Model:               2
Covariance Type:       nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
const         -0.1713      0.215      -0.796      0.426      -0.594      0.251
G1              0.1489      0.036       4.136      0.000       0.078      0.220
G2              0.8971      0.034      26.448      0.000       0.831      0.964
=====
Omnibus:                 478.095    Durbin-Watson:           1.852
Prob(Omnibus):            0.000    Jarque-Bera (JB):        11313.282
Skew:                    -3.000    Prob(JB):                 0.00
Kurtosis:                 22.554    Cond. No.                 72.7
=====

```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Name	Type	Size	Value
New_G1	float	1	0.15
New_G2	float	1	0.9
School	DataFrame	(649,)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
X	DataFrame	(649,)	Column names: const, G1, G2
Y	Series	(649, 33)	Series object of pandas.core.series module
df	DataFrame	(649, 3)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
predictions	Series	(649, 33)	Series object of pandas.core.series module



Logistic Regression:

We provide the code of Logistic Regression in appendices ([Appendix G](#)). Our aim is to predict the final grade of students. The output of Logistic Regression is presented below.

Output:

	school	sex	age	add2ess	famsize	...	health	absences	G1	G2	y
0	1	1	18	1	2	...	3	4	0	11	1
1	1	1	17	1	2	...	3	2	9	11	1
2	1	1	15	1	1	...	3	6	12	13	1
3	1	1	15	1	2	...	5	0	14	14	1
4	1	1	16	1	2	...	5	0	11	13	1
5	1	2	16	1	1	...	5	6	12	12	1
6	1	2	16	1	1	...	3	0	13	12	1
7	1	1	17	1	2	...	1	2	10	13	1
8	1	2	15	1	1	...	1	0	15	16	1
9	1	2	15	1	2	...	5	0	12	12	1
10	1	1	15	1	2	...	2	2	14	14	1
11	1	1	15	1	2	...	4	0	10	12	1
12	1	2	15	1	1	...	5	0	12	13	1
13	1	2	15	1	2	...	3	0	12	12	1
14	1	2	15	1	2	...	3	0	14	14	1
15	1	1	16	1	2	...	2	6	17	17	1
16	1	1	16	1	2	...	2	10	13	13	1
17	1	1	16	1	2	...	4	2	13	14	1
18	1	2	17	1	2	...	5	2	8	8	0
19	1	2	16	1	1	...	5	6	12	12	1
20	1	2	15	1	2	...	1	0	12	13	1
21	1	2	15	1	2	...	5	0	11	12	1
22	1	2	16	1	1	...	5	0	12	13	1
23	1	2	16	1	1	...	5	2	10	10	1
24	1	1	15	2	2	...	5	2	10	11	1
25	1	1	16	1	2	...	5	6	10	11	1

26	1	2	15	1	2	...	5	8	11	12	1
27	1	2	15	1	2	...	1	0	11	11	1
28	1	2	16	1	1	...	5	2	12	12	1
29	1	2	16	1	2	...	5	4	12	11	1
...
619	2	1	18	1	2	...	3	6	13	12	1
620	2	1	17	1	1	...	1	4	15	14	1
621	2	1	17	2	2	...	3	0	13	13	1
622	2	2	18	2	2	...	3	0	8	10	0
623	2	2	18	1	1	...	5	0	15	16	1
624	2	1	17	2	2	...	1	0	8	8	0
625	2	1	18	1	2	...	4	0	10	10	1
626	2	1	18	2	2	...	5	0	7	5	0
627	2	2	18	2	1	...	3	3	9	10	1
628	2	1	17	1	2	...	3	8	10	11	1
629	2	1	17	2	2	...	1	4	7	8	0
630	2	1	18	2	1	...	1	0	15	17	1
631	2	1	18	2	2	...	4	4	10	11	1
632	2	1	19	2	2	...	3	4	7	8	0
633	2	1	18	2	1	...	2	1	13	14	1
634	2	1	18	1	2	...	1	1	16	16	1
635	2	1	17	2	2	...	1	10	8	9	0
636	2	2	18	1	2	...	2	4	17	18	1
637	2	2	18	2	2	...	5	0	7	7	0
638	2	2	17	1	2	...	3	4	14	15	1
639	2	2	19	2	2	...	5	0	5	8	0
640	2	2	18	2	2	...	3	0	7	7	0
641	2	1	18	2	2	...	4	0	14	17	1
642	2	1	17	1	2	...	1	0	6	9	1
643	2	1	18	2	2	...	5	4	7	9	1
644	2	1	19	2	2	...	5	4	10	11	1
645	2	1	18	1	1	...	1	4	15	15	1
646	2	1	18	1	2	...	5	6	11	12	0
647	2	2	17	1	1	...	2	6	10	10	1
648	2	2	18	2	1	...	5	4	10	11	1

[649 rows x 33 columns]

(649, 33)

['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2', 'y']

(161,)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

y = column_or_1d(y, warn=True)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

y = column_or_1d(y, warn=True)

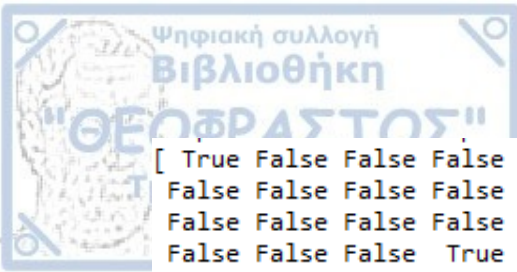
length of oversampled data is 768

Number of no subscription in oversampled data 384

Number of subscription 384

Proportion of no subscription data in oversampled data is 0.5

Proportion of subscription data in oversampled data is 0.5



```
[ True False False False False False False False False False False False
False False False False False False False False False False False False
False False False False False False False False False False False False
False False False True False False False False False False False False
False False False False False True False False False False False False
False False False False False False False False False False False False
False False False False False False False False False False False False
False True False False False False False False False False False False
False False False False False False False False False False False False
True False False False False False False False False False False False
False False False False False False False False False False False False
False False True True True True True True True False False False
False False False False False False True True True True True True
True True False False]
```

```
[ 1 32 82 42 68 104 38 41 106 47 90 141 67 69 57 99 70 113
101 123 117 102 11 58 7 12 96 66 29 53 139 110 14 48 81 64
78 116 93 1 136 91 61 3 129 77 21 76 86 125 4 44 45 1
100 115 19 138 24 98 97 18 8 118 103 87 54 28 34 130 43 83
105 74 17 126 6 109 30 22 31 27 111 112 25 1 122 37 84 36
39 88 92 5 35 33 13 89 55 51 62 63 108 71 59 135 46 2
1 26 72 128 65 49 60 56 132 80 114 120 124 95 131 52 79 121
134 137 127 20 140 23 15 107 1 1 1 1 1 1 1 50 94 119
73 16 85 10 9 40 1 1 1 1 1 1 1 1 75 133]
```

Warning: Maximum number of iterations has been exceeded.
Current function value: 0.311345
Iterations: 35

Function evaluations: 42
Gradient evaluations: 42

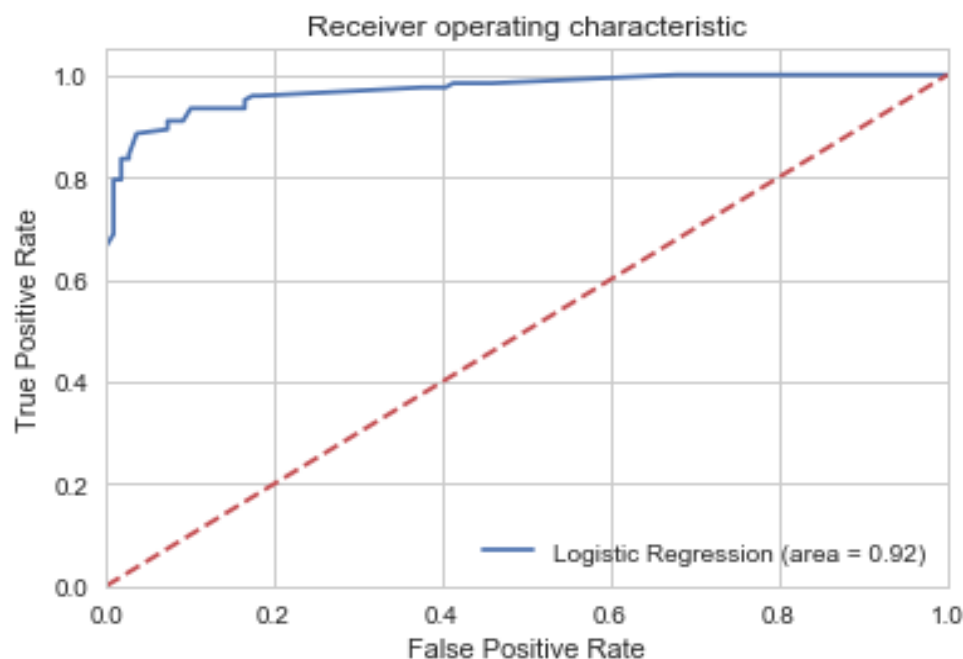
Logit Regression Results			
=====			
Dep. Variable:	y	No. Observations:	768
Model:	Logit	Df Residuals:	749
Method:	MLE	Df Model:	18
Date:	Thu, 14 Mar 2019	Pseudo R-squ.:	0.5508
Time:	02:45:23	Log-Likelihood:	-239.11
converged:	False	LL-Null:	-532.34
		LLR p-value:	6.288e-113

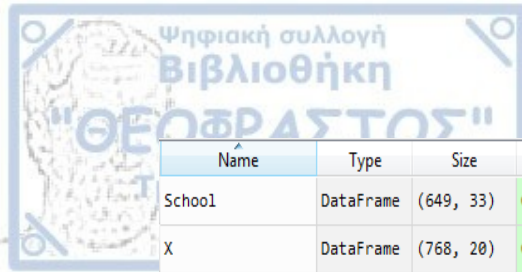
	coef	std err	z	P> z	[0.025	0.975]
reason_2	0.1277	nan	nan	nan	nan	nan
failures_0	0.0671	nan	nan	nan	nan	nan
goout_3	0.3023	nan	nan	nan	nan	nan
absences_5	0.7997	nan	nan	nan	nan	nan
G1_10	0.1888	nan	nan	nan	nan	nan
G1_11	3.6075	nan	nan	nan	nan	nan
G1_12	2.8330	nan	nan	nan	nan	nan
G1_13	2.5927	nan	nan	nan	nan	nan
G1_14	2.9104	nan	nan	nan	nan	nan
G1_15	1.6754	nan	nan	nan	nan	nan
G1_16	1.3937	nan	nan	nan	nan	nan
G1_10	0.1888	nan	nan	nan	nan	nan
G2_10	0.7451	nan	nan	nan	nan	nan
G2_11	2.5992	nan	nan	nan	nan	nan
G2_12	2.8651	nan	nan	nan	nan	nan
G2_13	2.5947	nan	nan	nan	nan	nan
G2_14	2.2373	nan	nan	nan	nan	nan
G2_15	1.9031	nan	nan	nan	nan	nan
G2_16	1.2684	nan	nan	nan	nan	nan
G2_17	1.4494	nan	nan	nan	nan	nan

Possibly complete quasi-separation: A fraction 0.15 of observations can be perfectly predicted. This might indicate that there is complete quasi-separation. In this case some parameters will not be identified.
Accuracy of logistic regression classifier on test set: 0.92

```
[[105  4]
 [ 14 108]]
```

	precision	recall	f1-score	support
0	0.88	0.96	0.92	109
1	0.96	0.89	0.92	122
avg / total	0.93	0.92	0.92	231





Name	Type	Size	Value
School	DataFrame	(649, 33)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
X	DataFrame	(768, 20)	Column names: reason_2, failures_0, goout_3, absences_5, G1_10, G1_11, ...
X_test	DataFrame	(231, 20)	Column names: reason_2, failures_0, goout_3, absences_5, G1_10, G1_11, ...
X_train	DataFrame	(537, 20)	Column names: reason_2, failures_0, goout_3, absences_5, G1_10, G1_11, ...
cat_list	DataFrame	(649, 16)	Column names: G2_0, G2_5, G2_6, G2_7, G2_8, G2_9, G2_10, G2_11, G2_12, ...
cat_vars	list	32	['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fed ...
cols	list	20	['reason_2', 'failures_0', 'goout_3', 'absences_5', 'G1_10', 'G1_11', ...
confusion_matrix	int64	(2, 2)	[[105 4] [14 108]]
datal	DataFrame	(649, 193)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
df	DataFrame	(649, 193)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
df_final	DataFrame	(649, 161)	Column names: y, school_1, school_2, sex_1, sex_2, age_15, age_16, age ...
df_final_vars	list	161	['y', 'school_1', 'school_2', 'sex_1', 'sex_2', 'age_15', 'age_16', 'a ...
df_vars	list	193	['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fed ...
Name	Type	Size	Value
fpr	float64	(61,)	[0. 0. 0. ... 0.4587156 0.67889908 1. ...
logit_roc_auc	float64	1	0.9242743269664612
os_data_X	DataFrame	(768, 160)	Column names: school_1, school_2, sex_1, sex_2, age_15, age_16, age_17 ...
os_data_y	DataFrame	(768, 1)	Column names: y
prosorin	object	(161,)	ndarray object of numpy module
thresholds	float64	(61,)	[0.99434491 0.99002953 0.9886928 ... 0.08073715 0.07452801 0. ...
to_keep	list	161	['y', 'school_1', 'school_2', 'sex_1', 'sex_2', 'age_15', 'age_16', 'a ...
tpr	float64	(61,)	[0.00819672 0.03278689 0.04918033 ... 0.98360656 1. 1. ...
var	str	1	G2
y	Series	(768,)	Series object of pandas.core.series module
y_pred	int64	(231,)	[0 1 1 ... 0 0 1]
y_test	Series	(231,)	Series object of pandas.core.series module
y_train	Series	(537,)	Series object of pandas.core.series module

Quantum Regression:

We provide the code of Quantum Regression in appendices ([Appendix H](#)).

Our aim is to predict the final grade of students. The output of Quantum Regression is presented below.

Output:

```
state_generators = dict()
state_generators['input'] = input_prog
state_generators['output'] = output_prog
state_generators['grad'] = grad_prog

initial_theta = np.random.uniform(0.0, 2*np.pi, size=3*n_qubits*depth)

operator = pq.Program(n_qubits)
operator.inst(Z, 0)
operator_programs = [operator]
est = QCL(state_generators, initial_theta, loss="mean_squared_error",
          operator_programs=operator_programs, epochs=20, batch_size=m,
          verbose=True)

est.fit(X,y)
results = est.get_results()
print(results)
```


Name	Type	Size	Value
Parad	float64	(649,)	[-1. -0.1 0.2 ... 0.1 0. 0.]
Parad2	float64	(649,)	[0.1 0.1 0.2 ... -0.1 0. 0.1]
School	DataFrame	(649, 33)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
X	float64	(649,)	[-1. -0.1 0.2 ... 0.1 0. 0.]
X_test	float64	(649,)	[-1. -0.1 0.2 ... 0.1 0. 0.]
Z	float64	(2, 2)	[[1. 0.] [0. -1.]]
depth	int	1	3
df	DataFrame	(649, 33)	Column names: school, sex, age, add2ess, famsize, Ps121us, Medu, Fedu, ...
initial_theta	float64	(27,)	[3.09168751 2.2650118 2.49432685 ... 3.81415956 3.67470872 0.50319296 ...
m	int	1	649
n_qubits	int	1	3
operator_programs	list	1	[Program]
results	OptResults	6	OptResults object of optimizer module
state_generators	dict	3	{'input':function, 'output':function, 'grad':function}
y	float64	(649,)	[0.1 0.1 0.2 ... -0.1 0. 0.1]
y_pred	float64	(649, 1)	[[0.20664799] [-0.02425995]

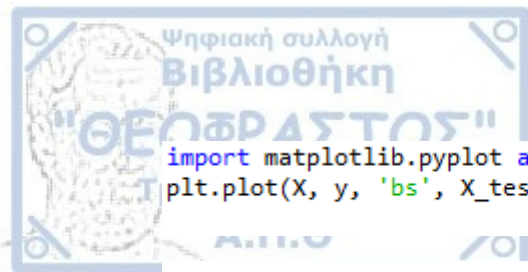
```
In [12]: print(results)
{'theta': array([3.09168751, 2.2650118 , 2.49432685, 5.10400605, 4.54727012,
4.39181545, 5.31613726, 3.75461054, 2.48358065, 4.36619517,
4.84403917, 1.3589583 , 4.04567775, 4.96235862, 2.88011611,
5.4941249 , 1.27715984, 6.10524986, 4.16430622, 1.95439806,
0.65573147, 1.8574468 , 1.1308391 , 3.27321019, 3.81415956,
3.67470872, 0.50319296]), 'coeff': 0.8726539270163359, 'loss': 0.03539840369083222, 'history_loss': [0.33563107552303045, 0.16678360357479552,
0.10995682309369449, 0.07926703658330951, 0.06680242839355208, 0.062395108953629236, 0.05853602248594747, 0.055020720757291995, 0.051832704440071986,
0.04895952500192934, 0.04640569610740909, 0.04416812220897138, 0.042240544882982414, 0.04060602427317463, 0.039241945325953105, 0.03811839001462474,
0.037203921038133886, 0.036465237046362974, 0.035872510122532696, 0.03539840369083222], 'history_theta': [array([3.09168751, 2.2650118 , 2.49432685, 5.10400605, 4.54727012,
4.39181545, 5.31613726, 3.75461054, 2.48358065, 4.36619517,
4.84403917, 1.3589583 , 4.04567775, 4.96235862, 2.88011611,
5.4941249 , 1.27715984, 6.10524986, 4.16430622, 1.95439806,
0.65573147, 1.8574468 , 1.1308391 , 3.27321019, 3.81415956,
3.67470872, 0.50319296]), array([3.09168751, 2.2650118 , 2.49432685, 5.10400605, 4.54727012,
4.39181545, 5.31613726, 3.75461054, 2.48358065, 4.36619517,
4.84403917, 1.3589583 , 4.04567775, 4.96235862, 2.88011611,
5.4941249 , 1.27715984, 6.10524986, 4.16430622, 1.95439806,
0.65573147, 1.8574468 , 1.1308391 , 3.27321019, 3.81415956,
3.67470872, 0.50319296]), array([3.09168751, 2.2650118 , 2.49432685, 5.10400605, 4.54727012,
4.39181545, 5.31613726, 3.75461054, 2.48358065, 4.36619517,
4.84403917, 1.3589583 , 4.04567775, 4.96235862, 2.88011611,
5.4941249 , 1.27715984, 6.10524986, 4.16430622, 1.95439806,
0.65573147, 1.8574468 , 1.1308391 , 3.27321019, 3.81415956,
3.67470872, 0.50319296]), array([3.09168751, 2.2650118 , 2.49432685, 5.10400605, 4.54727012,
4.39181545, 5.31613726, 3.75461054, 2.48358065, 4.36619517,
4.84403917, 1.3589583 , 4.04567775, 4.96235862, 2.88011611,
```



```
X_test = X
y_pred = est.predict(X_test)
print(y_pred)
```

[0.50071635]
[0.2727206]
[0.3603614]
[0.0798974]
[-0.02425995]
[0.2727206]
[0.17891786]
[0.17891786]
[0.3603614]
[0.17891786]
[-0.02425995]
[0.2727206]
[0.17891786]
[-0.02425995]
[0.3603614]
[0.17891786]
[0.3603614]
[-0.2441329]
[-0.02425995]
[0.17891786]
[0.2727206]
[0.0798974]
[-0.02425995]
[-0.13302855]
[-0.2441329]
[-0.13302855]
[-0.02425995]

202



```
import matplotlib.pyplot as plt
plt.plot(X, y, 'bs', X_test, y_pred, 'r-')
```

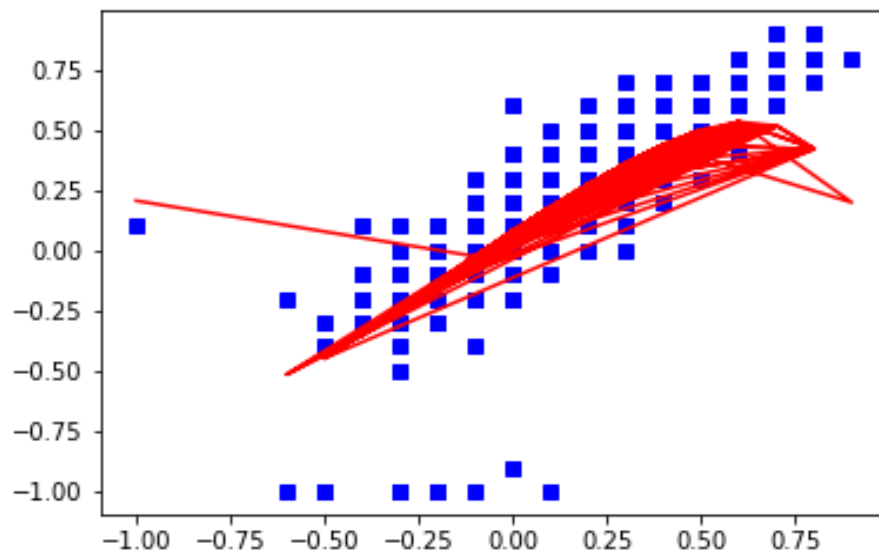


Figure 44: Quantum Regression Curve

In the above Figure 44 we can observe the data fitting after the quantum regression implementation. The visual results of quantum regression are not very distinct and comprehensible. The results of Quantum Regression for the prediction of the students performance are presented in Table 44. However, if we use a different data set, the results and the curve-line might be better.

Comments on the Results

We select a data set (Student Performance Data Set) and implement:

- Multiple Linear Regression
- Logistic Regression
- Quantum Regression

The results of Quantum Regression may not be directly comparable with the rest two methods (Multiple Linear Regression, Logistic Regression), but in case that our model is not adapted well to the data by using Linear or Logistic Regression, we have an extra selection (Quantum Regression). This algorithm can work for every data set. It belongs to the category of QAML (Quantum Assisted Machine Learning) algorithms, that is it is a quantum algorithm which can be implemented and give results in a classical computer. However, if it was potential to implement this algorithm (maybe with few modifications) in a quantum computer, the extracted results may be more useful and well-presented.



The goal of this work is to explore the possibility to apply quantum probability models and related statistical algorithms to data sets. This goal involves two parts:

Firstly, we analyze the available criteria for data sets. We selected the Wigner-d'Espagnat inequality as the simplest criterion to apply. The main result of this research is to construct a data set, which violates the Wigner-d'Espagnat inequality.

Secondly, we implement Quantum Machine Learning from data sets. We use data sets related with the school grades system and absenteeism at work. In these data sets firstly we implement classical algorithms, such as k-Means or hierarchical clustering and linear or logistic regression. Then, we implement quantum algorithms, such as quantum clustering and quantum regression. The code of quantum regression in Python was developed by MIT university and the code in Matlab by David Horn. However, when we implemented them in our data sets, these codes were not working. Therefore, we modified these codes and finally got results. We also compared to some extent the results between the quantum and classical algorithms statistically and visually.

In addition to extensive bibliographical processing the following new results are presented:

1) The example on Wigner-d'Espagnat inequality (pages 88-94, chapter 2) is original. No such example was found in the bibliography. This example is very simple and straightforward, showing in a clear and simple way the distinction between Kolmogorov and quantum probability.

2) The example on clustering (pages 149-162, chapter 4) is original, although there are similar examples in the bibliography. This example is very simple and helps to compare classical clustering algorithms and quantum clustering algorithm.

3) The example on regression (pages 189-204, chapter 5) is original, although there are similar examples in the bibliography. This example is very simple and helps to compare classical regression algorithms and quantum regression algorithm.

It is expected that quantum computers implementing quantum statistics will find useful applications demonstrating the advantage of quantum resources (High-Level Steering Committee, 2017, Lavín, Anguita, 2018).



Appendix A: Dynamic Quantum Clustering Algorithm

Here we present the steps of the DQC algorithm. A DQC analysis begins with data that is presented as an $m \times n$ data matrix. Each data point is one of the m rows of the matrix and is defined by the n -numbers that appear in that row. These n numbers are referred to as features and the set of all possible sets of n -values that might appear in a row is referred to as the feature space. The process of creating a clustering algorithm using ideas borrowed from quantum mechanics starts with the creation of a potential function that serves as a proxy for the density of data points. We do this as follows:

1) Define a function – Parzen estimator on the n -dimensional feature space. This function is constructed as a sum of Gaussian functions centered at each data point, i.e. for m data points \vec{x}_l :

$$\varphi(\vec{x}) = \sum_{l=1}^m e^{-\frac{1}{2\sigma^2}(\vec{x} - \vec{x}_l) \cdot (\vec{x} - \vec{x}_l)} \quad (4.1)$$

2) We derive a potential function $V(\vec{x})$ defined over the same n -dimensional space. Since $\varphi(\vec{x})$ is a positive definite function, we can define $V(\vec{x})$ as that function for which $\varphi(\vec{x})$ satisfies the time-independent Schrödinger equation:

$$-\frac{1}{2\sigma^2} \nabla^2 \varphi + V(\vec{x}) \varphi = E \varphi = 0 \quad (4.2)$$

We note that the value zero is chosen to simplify the mathematics and plays no important role. Clearly, the energy E can always be set to zero by adding a constant to the potential. It is straightforward to solve the above equation (4.2) for $V(\vec{x})$.

3) The quantum potential is of interest for two main reasons: first, physical intuition tells us that the local minima of $V(\vec{x})$ will correspond to the local maxima of $\varphi(\vec{x})$ if the latter are well separated from one another. Second, $V(\vec{x})$ may have minima at points, where $\varphi(\vec{x})$ exhibits no corresponding maxima. If the Parzen estimator is meant to be a proxy for the density of the data, then DQC's quantum potential can be thought of as an unbiased way of contrast enhancing the Parzen function to better reveal structure in the data. An additional benefit of working with this contrast enhanced version of the Parzen estimator is that its features depend much less sensitively upon the choice of parameter σ that appears in Equation (4.2).

4) Using the Hamiltonian defined by this potential, evolve each Gaussian that is associated with a specific data point by multiplying it by the quantum time-evolution operator $e^{-i\delta t H}$, where δt is chosen to be small. We note this operator is constructed in the subspace spanned by all of the Gaussians corresponding to the original data points.

5) We compute the new location of the center of each evolved Gaussian. Hereafter we refer to it as the evolution of the data-point.

6) Iterate this procedure. Ehrenfest's theorem guarantees that for small time steps, the center of each Gaussian will follow Newton's laws of motion, where the force is given by the expectation value of the gradient-descent in classical mechanics. The fact that we use quantum evolution rather than more familiar classical methods, allows us to convert the computationally intensive problem of gradient descent in a multi-dimensional potential into an exercise in matrix multiplication. This greatly reduces the workload and allows parallel execution of the code in order to quickly deal with enormous sets of data (Weinstein, Meirer, Hume, Sciau, Shaked, Hofstetter, Horn, 2013).

Appendix B: Detailed description of Dynamic Quantum Clustering method by Horn, Weinstein and Marvin

We present here the detailed description of the Dynamic Quantum Clustering method. As we already noted, the conversion of the static Quantum Clustering method to a full dynamical one, begins by focusing attention on the Gaussian wave-function: $\psi_i(\vec{x}) = C e^{-\frac{(\vec{x}-\vec{x}_i)^2}{2\sigma^2}}$ associated with the i^{th} data point, where C is the appropriate normalization factor. Thus, by construction, the expectation value of the operator \vec{x} in this state is simply the coordinates of the original data point:

$$\vec{x}_i = \langle \psi | \vec{x} | \psi \rangle = \int d\vec{x} \psi_i^*(\vec{x}) \vec{x} \psi_i(\vec{x}) \quad (4.9)$$

The dynamical part of the DQC algorithm is that, having constructed the potential function $V(\vec{x})$, we study the time evolution of each state $\psi_i(\vec{x})$ as determined by the time dependent Schrödinger equation:

$$i \frac{\partial \psi_i(\vec{x}, t)}{\partial t} = H \psi_i(\vec{x}, t) = \left(-\frac{\nabla^2}{2m} + V(\vec{x}) \right) \psi_i(\vec{x}, t) \quad (4.10)$$

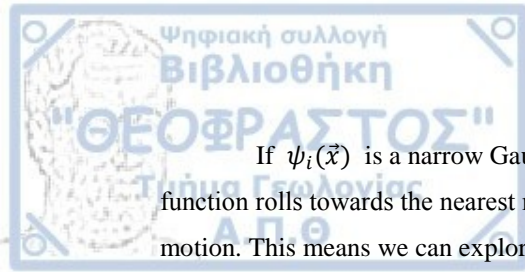
where m is an arbitrarily chosen mass for a particle moving in d -dimensions. If we set $m = \frac{1}{\sigma^2}$ then, by construction, $\psi(\vec{x})$ of Equation (4.7) is the lowest energy eigenstate of the Hamiltonian. If m is chosen to have a different value, then not only does each individual state $\psi_i(\vec{x})$ evolve in time, but so does the sum of the states $\psi(\vec{x})$ (Weinstein, Marvin, Horn, 2009).

The important feature of quantum dynamics, which makes the evolution so useful in the clustering problem, is that according to Ehrenfest's theorem, the time-dependent expectation value:

$$\langle \psi(t) | \vec{x} | \psi(t) \rangle = \int d\vec{x} \psi_i^*(\vec{x}, t) \vec{x} \psi_i(\vec{x}, t) \quad (4.11)$$

satisfies the equation:

$$\frac{d^2 \langle \vec{x}(t) \rangle}{dt^2} = -\frac{1}{m} \int d\vec{x} \psi_i^*(\vec{x}, t) \vec{\nabla} V(\vec{x}) \psi_i(\vec{x}, t) = \langle \psi(t) | \vec{\nabla} V(\vec{x}) | \psi(t) \rangle \quad (4.12)$$



If $\psi_i(\vec{x})$ is a narrow Gaussian, this is equivalent to saying that the center of each wave-function rolls towards the nearest minimum of the potential according to the classical Newton's law of motion. This means we can explore the relation of this data point to the minima of $V(\vec{x})$ by following the time-dependent trajectory: $\langle \vec{x}_i(t) \rangle = \langle \psi_i(t) | \vec{x} | \psi_i(t) \rangle$.

Clearly, given Ehrenfest's theorem, we expect to see any points located in, or near, the same local minimum of $V(\vec{x})$ to oscillate about that minimum, coming together and moving apart. In our numerical solutions we generate animations which display this dynamics for a finite time. This allows us to visually trace the clustering of points associated with each one of the potential minima (Weinstein, Marvin, Horn, 2009).

In their quantum clustering paper Horn and Gottlieb successfully used classical gradient descent to cluster data by moving points (on classical trajectories) to the nearest local minimum of $V(\vec{x})$. The idea being that points which end up at the same minimum are in the same cluster. At first glance it would seem that DQC replaces the conceptually simple problem of implementing gradient descent with the more difficult one of solving complicated partial differential equations. We will show the difficulty is only apparent. In fact, the solution of the Schrödinger equation can be simplified considerably and will also allow further insights than the gradient descent method. The DQC algorithm translates the problem of solving the Schrödinger equation into a matrix form which captures most of the details of the analytic problem, but which involves $N \times N$ -matrices whose dimension, N , is less than or equal to the number of data points. This reduction is independent of the data-dimension of the original problem. From a computational point of view there are many advantages to this approach. First, the formulas for constructing the mapping of the original problem to a matrix problem are all analytic and easy to evaluate, thus computing the relevant reduction is fast. Second, the evolution process only involves matrix multiplications, so many data points can be evolved simultaneously and, on a multi-core processor, in parallel. Third the time involved in producing the animations showing how the points move in data space scales linearly with the number of dimensions to be displayed. Finally, by introducing an m that is different from $1/\sigma^2$ we allow ourselves the freedom of employing low σ , which introduces large numbers of minima into V , yet also having a low value for m which guarantees efficient tunneling, thus connecting points that may be located in nearby, nearly degenerate potential minima. By using this more general Hamiltonian, we reduce the sensitivity of the calculation to the specific choice of σ (Weinstein, Marvin, Horn, 2009).

Here we describe the Calculation Method. We begin by assuming that there are n -data points that we wish to cluster. To these data points we associate n -states: $|\psi_i\rangle$. These states are n Gaussian wave-functions such that the i^{th} Gaussian is centered on the coordinates of the i^{th} data point. These states form a basis for the vector space within which we calculate the evolution of our model (Weinstein, Marvin, Horn, 2009).

Let us denote by N , the $n \times n$ matrix formed from the scalar products:

$$N_{i,j} = \langle \psi_i | \psi_j \rangle \quad (4.13)$$

and by H , the $n \times n$ matrix:

$$H_{i,j} = \langle \psi_i | H | \psi_j \rangle \quad (4.14)$$

and by $\vec{X}_{i,j}$ the matrix of expectation values:

$$\vec{X}_{i,j} = \langle \psi_i | \vec{x} | \psi_j \rangle \quad (4.15)$$

The calculation process can be described in five steps:

First, begin by finding the eigenvectors of the symmetric matrix N which correspond to states having eigenvalues larger than some pre-assigned value (e.g. 10^{-5}). These vectors are linear combinations of the original Gaussians which form an orthonormal set.

Second, compute H in this orthonormal basis H^{tr} .

Third, do the same for $\vec{X}_{i,j}$.

Fourth, find the eigenvectors and eigenvalues of H^{tr} , construct $|\psi_i(t)\rangle = e^{-it H^{tr}} |\psi_i\rangle$ that is the solution to the reduced time dependent Schrödinger problem:

$$i \frac{\partial}{\partial t} |\psi_i(t)\rangle = H^{tr} |\psi_i(t)\rangle \quad (4.16)$$

such that $|\psi_i(t=0)\rangle = |\psi_i\rangle$.

Fifth, construct the desired trajectories:

$$\langle \vec{x}_i(t) \rangle = \langle \psi_i | e^{it H^{tr}} \vec{X} e^{-it H^{tr}} | \psi_i \rangle \quad (4.17)$$

by evaluating this expression for a range of t and use them to create an animation.

Stop

the animation when clustering of points is obvious (Weinstein, Marvin, Horn, 2009).

Appendix C: Quantum Clustering Algorithm by Horn and Gottlieb

It starts out with a Parzen window approach, assigning to each data-point a Gaussian of width σ thus constructing:

$$\psi(x) = \sum_i e^{-\frac{(x-x_i)^2}{2\sigma^2}} \quad (4.22)$$

that can serve (but for an overall normalization) as a probability density generating the data. One then proceeds to construct a potential function:

$$V(x) = E + \frac{\sigma^2 \nabla^2 \psi}{\psi} \quad (4.23)$$

where

$$E = - \min \frac{\frac{\sigma^2}{2} \nabla^2 \psi}{\psi} \quad (4.24)$$

thus rendering V positive definite. In fact V has a global minimum at zero, and grows as a polynomial of second order outside the domain over which the data points are defined. Within this domain, V develops minima that are identified with cluster centers (Horn, Axel, 2003).

The intuition behind this approach is that this choice of V is the correct one for the Schrodinger equation:

$$H\psi \equiv \left(-\frac{\sigma^2}{2} \nabla^2 + V(\mathbf{x}) \right) \psi = E\psi \quad (4.25)$$

whose solution (lowest eigenstate) is the probability density $\psi(\mathbf{x})$. In this equation, the potential function $V(\mathbf{x})$ can be regarded as the source of attraction, whereas the first Lagrangian term is the source of diffusion of the distribution, governed by the parameter σ (Horn, Axel, 2003).

Once the minima of $V(\mathbf{x})$ are defined as cluster centers, the assignment of data points to clusters can proceed through a gradient descent algorithm, allowing auxiliary point variables $\mathbf{y}_i(0) = \mathbf{x}_i$ to follow dynamics of:

$$\mathbf{y}_i(t + \Delta t) = \mathbf{y}_i(t) - \eta(t) \nabla V(\mathbf{y}_i(t)) \quad (4.26)$$

that lead to asymptotic fixed points: $\mathbf{y}_i(t) \rightarrow \mathbf{z}_i$ coinciding with the cluster centers.

We emphasize that although a search is carried out here for the minima of a continuous function $V(\mathbf{x})$, which may be a complex problem in high dimensions, it can in fact be simplified by evaluating this function only at the data points and their gradient descendants $V(\mathbf{y}_i)$ which is sufficient to carry out the algorithm of clustering (Horn, Axel, 2003).

In this section we describe the Hierarchical Quantum Clustering (QC) Algorithm. The QC algorithm has a free parameter σ that characterizes the length scale over which we search for cluster structures. Varying it from low to high values, we can get anywhere from N clusters (where N is the number of data points) to one cluster. The algorithm has to be applied judiciously, e.g. by limiting oneself to a small number of clusters that stays stable over a range of σ . It is however important to realize that this algorithm does not guarantee hierarchy, i.e. the assignments of data points to clusters does not follow a tree, or dendrogram representation, as σ is being varied (Horn, Axel, 2003).

We find it useful to define a modified version that produces a hierarchical formulation in an agglomerative manner. We start out with very low σ , such that each data point is a cluster of its own and we have the first trivial clustering $\mathbf{z}_i^1 = \mathbf{x}_i$. Then we increase σ by some amount obtaining, after the QC gradient descent algorithm, new clustering centers \mathbf{z}_i^2 . Although there are N values specified here, there should now be several coinciding with one another, thus describing small clusters with a few points in each (Horn, Axel, 2003).

We use z_i^2 as the data points in our next stage of QC, after once again increasing σ . This leads to a new set of cluster values z_i^3 . This procedure is continued until large σ values are reached with only one cluster. On the way it defines a dendrogram whose clustering quality we may compare to biological sample data. We call this method hierarchical quantum clustering (HQC) (Horn, Axel, 2003).

Appendix D: Quantum Clustering first visualization (Matlab)

- **norm_fcn**

```
norm_fcn.m x clustMeasure.m x fineCluster.m x graddesc.m
1 function nm = normc_fcn(m)
2 nm = sqrt(m.^2 ./ sum(m.^2)) .* sign(m);
3 end
```

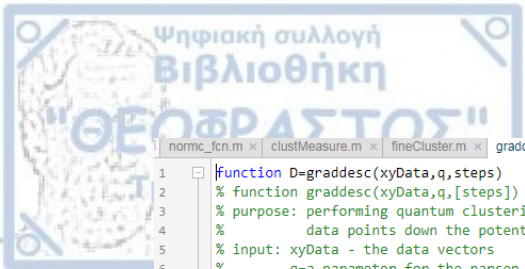
- **clustMeasure**

```
norm_fcn.m x clustMeasure.m x fineCluster.m x graddesc.m x plotClust.m x qc.m x QCscript.m x +
1 function [minkowski_measure,jacard_measure,purity,efficiency]=clustMeasure(clust,realClust)
2 % function [minkowski_measure,jacard_measure,purity,efficiency]=clustMeasure(clust,realClust)
3 % input: clust=vector with all the cluster # of each data point
4 % realClust=vector of the starting place of each cluster
5 % (assuming the data points are sorted accordingly)
6
7 pNum=length(clust);
8 % S=the clustering result in pairs - S(i,j)=1 iff data point i and j are assigned to the same cluster
9 S=(repmat(clust,1,pNum)==repmat(clust',pNum,1));
10 for i=1:(length(realClust)-1)
11 t(realClust(i):(realClust(i+1)-1))=i;
12 end
13 l=length(clust);
14 t(realClust(i+1):l)=i+1;
15 T=(repmat(t',1,1)==repmat(t,1,1));
16 % T=the true clustering (same definition as for S)
17
18 sum(sum(T==1));
19 sum(sum(T~=S));
20 minkowski_measure=sqrt(sum(sum(T~=S))/sum(sum(T==1)));
21 S1=S*2-1; % replace 0 for -1
22 TP=(T==S1); % all places where both T and S equal 1
23 jacard_measure = sum(sum(TP))/(sum(sum(T~=S))+sum(sum(TP)));
24
25 efficiency = sum(sum(TP))/sum(sum(T==1)); % n11/(n10 + n11)
26 purity= sum(sum(TP))/sum(sum(S==1)); % n11/(n01 + n11)
```

- **fineCluster**

```
norm_fcn.m x clustMeasure.m x fineCluster.m x graddesc.m x plotClust.m x qc.m x QCscript.m x +
1 function clust=fineCluster(xyData,minD)
2 % clust=fineCluster(xyData,minD) cluster xyData points when closer than minD
3 % output: clust=vector the cluster index that is assigned to each data point
4 % (it's cluster serial #)
5
6 n=length(xyData);
7 clust=zeros(1,length(xyData));
8 i=1;
9 clustInd=1;
10 while min(clust)==0
11 x=xyData(i,:);
12 D=sum(((repmat(x,n,1)-xyData).^2)).^.5;
13 clust(D<minD)=clustInd;
14 i=find(~clust);
15 if length(i)>0
16 i=i(1); % index of the first non-clustered point
17 end
18 clustInd=clustInd+1;
19 end
20 clust=clust';
```

- **graddesc**



```

1 function D=graddesc(xyData,q,steps)
2 % function graddesc(xyData,q,[steps])
3 % purpose: performing quantum clustering in and moving the
4 % data points down the potential gradient
5 % input: xyData - the data vectors
6 % q=a parameter for the parzen window variance (q=1/(2*sigma^2))
7 % steps=number of gradient descent steps (default=50)
8 % output: D=location of data o=point after GD
9 if nargin<3
10 steps=50;
11 end
12
13 eta=0.1;
14 D=xyData;
15 [V,P,E,dV] = qc (xyData,q,D);
16 for j=1:4
17 for i=1:(steps/4)
18 dV=normc_fcn(dV)';
19 D=D-eta*dV;
20 [V,P,E,dV] = qc (xyData,q,D);
21 end;
22 eta=eta*0.5;
23 end

```

- **plotClust**

```

1 %plot clusters
2
3
4 x=zeros(length(xyData),max(clust));
5 x=repmat(1:max(clust),length(xyData),1)';
6 index=repmat(clust,1,max(clust))';
7 x(x==index);
8 x(:,realClust)=x(:,realClust)+2;
9 BWcmap=[1 1 1;0 0 0;1 0 0;0.7 0.7 0.7;0.2 0.2 0.2];
10
11 colormap(BWcmap);
12 image(x*2)

```

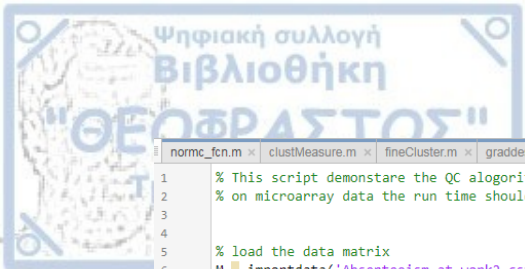
- **qc**

```

normcfcn.m x clustMeasure.m x fineCluster.m x graddesc.m x plotClust.m x qc.m x QCscript.m x +
1 function [V,P,E,dV] = qc (r1,q,r)
2 % function qc
3 % purpose: performing quantum clustering in n dimensions
4 % input:
5 %   r1 - a vector of points in n dimensions
6 %   q - the factor q which determines the clustering width
7 %   r - the vector of points to calculate the potential for. equals r1 if not specified
8 % output:
9 %   V - the potential
10 %   P - the wave function
11 %   E - the energy
12 %   dV - the gradient of V
13 % example: [V,P,E,dV] = qc ([1,1;1,3;3,3],5,[0.5,1,1.5]);
14 % see also: qc2d
15
16 %close all;
17 if nargin<3
18     r=r1;
19 end
20
21 %default q
22 if nargin<2
23     q=0.5;
24 end
25 %default xi
26 [pointsNum,dims] = size(r1);
27 calculatedNum=size(r,1);
28 % prepare the potential
29 V=zeros(calculatedNum,1);
30 dP2=zeros(calculatedNum,1);
31
32 % prepare P
33 P=zeros(calculatedNum,1);
34 singlePoint = ones(pointsNum,1);
35 singleLaplace = zeros(pointsNum,1);
36 singledV1=zeros(pointsNum,dims);
37 singledV2=zeros(pointsNum,dims);
38 % prevent division by zero
39 % calculate V
40 %run over all the points and calculate for each the P and dP2
41 for point = 1:calculatedNum
42     singlePoint = ones(pointsNum,1);
43     singleLaplace = singleLaplace.*0;
44     D2=sum((( repmat(r(point,:),calculatedNum,1)-r1).^2)');
45     singlePoint=exp(-q*D2)';
46     %EXPij=(repmat(singlePoint',calculatedNum,1).*( repmat(singlePoint,1,calculatedNum)));
47     % singleLaplace = sum((D2').*singlePoint);
48     for dim=1:dims
49         singleLaplace = singleLaplace + (r(point,dim)-r1(:,dim)).^2.*singlePoint;
50     end
51     for dim=1:dims
52         singledV1(:,dim) = (r(point,dim)-r1(:,dim)).*singleLaplace;
53     end
54     for dim=1:dims
55         singledV2(:,dim) = (r(point,dim)-r1(:,dim)).*singlePoint;
56     end
57     P(point) = sum(singlePoint);
58     dP2(point) = sum(singleLaplace);
59     dV1(point,:)=sum(singledV1,1);
60     dV2(point,:)=sum(singledV2,1);
61 end
62 % dill with zero
63 %v1(find(v1==0)) = min(v1(find(v1)));
64 %v2x(find(v2x==0)) = min(v2x(find(v2x>0)));
65 %v2y(find(v2y==0)) = min(v2y(find(v2y>0)));
66 P(find(P==0)) = min(P(find(P)));
67
68 V=-dims/2+q*dP2./P;
69 E=-min(V);
70 V=V+E;
71 for dim=1:dims
72     dV(:,dim)=-q*dV1(:,dim)+(V-E+(dims+2)/2).*dV2(:,dim);
73 end
74 dV(find(P==0),:)=0;

```

- **QCscript**



```

normc_fcn.m × clustMeasure.m × fineCluster.m × graddesc.m × plotClust.m × qc.m × QCscript.m × +
1 % This script demonstare the QC algorithm in a truncated SVD space |
2 % on microarray data the run time should take several minutes
3
4
5 % load the data matrix
6 M = importdata('Absenteeism_at_work2.csv', ';', 1)
7 M = M.data
8 %%load
9 % perform SVD - the result are 3 matrixes s.t genes x S x samples = M
10 [genes,S,samples] = svd(M,0);
11 dims=4;
12 q=2.4; % q=1/(2*sigma^2) => sigma=0.46 (smaller q -> less clusters)
13
14
15 %xyData=samples(:,1:dims); % load dims most significant vectors to xyData
16 xyData=genes(:,1:dims); % load dims most significant vectors to xyData
17
18
19
20 % data normalization (gives all vector unit length)
21 n = normc_fcn(xyData');
22 xyData=n';
23
24
25 %show_qc; % run qc and then plot the result (if more than 2 dimentions are used -
26 % then the result is the projection of on the first to dimentions)
27 % this procedure does not perform the gradient descent and uses only
28 % for presentation purpose
29
30 % QC
31 D=graddesc(xyData,q,80); %performs gradient descent on xyData with 20 steps
32 clust=fineCluster(D,0.1); % "collapse" the points to their final places and
33 % return the division of data into clusters
34
35 plotClust;
36 [mm,jm,purity,efficiency]=clustMeasure(clust,realClust); %minkowski measure and pairwise measure
37 QCjacard_measure=jm;
38 QCminkowski_measure=mm;
39 title(strcat('QC clustering ',int2str(dims),' dimensions jacard=' ,num2str(QCjacard_measure) ))

```


Appendix E: Quantum Clustering second visualization (Matlab)

- **normr**

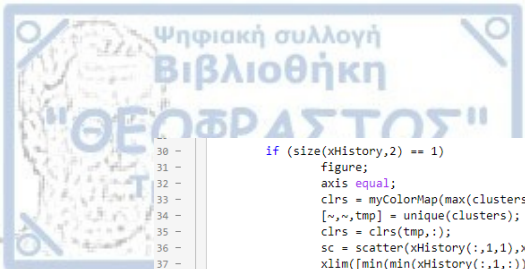
```

1  function n = normr(m)
2  %NORMR Normalize rows of matrix.
3  %
4  %   NORMR(M)
5  %   M - a matrix.
6  %   Returns a matrix the same size with each
7  %   row normalized to a vector length of 1.
8  %
9  %   See also NORMC, PNORMC.
10
11  % Mark Beale, 1-31-92
12  % Copyright (c) 1992-97 by The MathWorks, Inc.
13  % $Revision: 1.3 $ $Date: 1997/05/14 22:10:50 $
14
15  if nargin < 1,error('Not enough input arguments.');
```

- **DisplayQC**

```

1  function DisplayQC(xHistory,clusters,principalComponents,dataInPcbasis)
2  % displays the quantum clustering process dynamically
3  % xHistory - the entire evolution of QC as returned from the function PerformGQQC
4  % clusters - vector with cluster indexes of data points. numel(clusters) ~ size(xHistory,1). can be [] for no clustering.
5  % optional:principalComponents - the principal components of the original data, as returned from function pca. The display will project the data back to the origi
6  % optional:dataInPcbasis - the original data in the principal component basis. The display will also use the coordinated of the unused principal components in t
7
8  captureVideo = false;
9
10
11  showLines = 0;
12
13  if (nargin>=3)
14      n = size(xHistory,2);
15      newXHistory = zeros(size(xHistory,1),size(principalComponents,1),size(xHistory,3));
16      for ii=1:size(xHistory,3)
17          if (nargin==4)
18              newXHistory(:,ii) = [xHistory(:,ii),dataInPcbasis(:,(n+1):end)]*principalComponents';
19          else
20              newXHistory(:,ii) = xHistory(:,ii)*principalComponents(:,1:n)';
21          end
22      end
23      xHistory = newXHistory;
24  end
25
26  if isempty(clusters)
27      clusters = ones(size(xHistory,1),1);
28  end
29
30
```



```

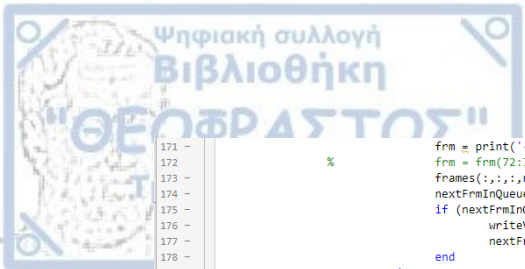
30 - if (size(xHistory,2) == 1)
31 -     figure;
32 -     axis equal;
33 -     clrs = myColorMap(max(clusters));
34 -     [~,~,tmp] = unique(clusters);
35 -     clrs = clrs(tmp,:);
36 -     sc = scatter(xHistory(:,1,1),xHistory(:,1,1)*0,10,clrs,'filled');
37 -     xlim([min(min(xHistory(:,1,:))), max(max(xHistory(:,1,:)))]);
38 -     ylim([-1, 1]);
39 -     if captureVideo
40 -         frm = print('-r0','-opengl','-noui');
41 -         frm = frm(250:570,:);
42 -         %
43 -         frames = zeros(size(frm,1),size(frm,2),3,20);
44 -         nextFrmInQueue = 1;
45 -         frames(:, :, nextFrmInQueue) = frm;
46 -         nextFrmInQueue = nextFrmInQueue+1;
47 -         vid = VideoWriter('filename.mp4','MPEG-4');
48 -         vid.FrameRate = 18;
49 -         open(vid);
50 -     end
51 -     for ii=1:size(xHistory,3)
52 -         set(sc,'XData',xHistory(:,1,ii));
53 -         title(['step #' num2str(ii-1) '/' num2str(size(xHistory,3))]);
54 -         if captureVideo
55 -             frm = print('-r0','-opengl','-noui');
56 -             frm = frm(72:700,180:1050,:);
57 -             %
58 -             frames(:, :, nextFrmInQueue) = frm;
59 -             nextFrmInQueue = nextFrmInQueue+1;
60 -             if (nextFrmInQueue==21)
61 -                 writeVideo(vid,frames/255);
62 -                 nextFrmInQueue = 1;
63 -             end
64 -         end
65 -         pause(0.05);
66 -     end
67 -     if captureVideo
68 -         frames(:, :, nextFrmInQueue) = frm;
69 -         writeVideo(vid,frames(:, :, 1:nextFrmInQueue)/255);
70 -         close(vid);
71 -     end
72 -     figure;
73 -     axis equal;
74 -     scatter(xHistory(:,1,1),xHistory(:,1,1)*0,10,clrs,'filled');
75 -     hold on;
76 -     xlim([min(min(xHistory(:,1,:))), max(max(xHistory(:,1,:)))]);
77 -     ylim([-1, 1]);
78 -     for ii=1:max(clusters)
79 -         %%
80 -         text(mean(xHistory(clusters==ii,1,1)),0.5,num2str(ii),'BackgroundColor','w','EdgeColor','k','HorizontalAlignment','center','VerticalAl:
81 -         text(mean(xHistory(clusters==ii,1,1)),0.5,num2str(sum(clusters==ii)), 'BackgroundColor','w','EdgeColor','k','HorizontalAlignment','cent
82 -     end
83 - elseif (size(xHistory,2) == 2)
84 -     figure;
85 -     axis equal;
86 -     clrs = myColorMap(max(clusters));
87 -     [~,~,tmp] = unique(clusters);
88 -     clrs = clrs(tmp,:);

```

```

86 - sc = scatter(xHistory(:,1,1),xHistory(:,2,1),10,clrs,'filled');
87 - xlim([min(min(xHistory(:,1,:))) , max(max(xHistory(:,1,:)))]);
88 - ylim([min(min(xHistory(:,2,:))) , max(max(xHistory(:,2,:)))]);
89 - if captureVideo
90 -     frm = print('-RGBImage','-r0','-opengl','-noui');
91 -     % frm = frm(250:570,:,:);
92 -     frames = zeros(size(frm,1),size(frm,2),3,20);
93 -     nextFrmInQueue = 1;
94 -     frames(:,:,nextFrmInQueue) = frm;
95 -     nextFrmInQueue = nextFrmInQueue+1;
96 -     vid = VideoWriter('filename.mp4','MPEG-4');
97 -     vid.FrameRate = 18;
98 -     open(vid);
99 -
100 - end
101 - for ii=1:size(xHistory,3)
102 -     set(sc,'XData',xHistory(:,1,ii),'YData',xHistory(:,2,ii));
103 -     title(['step #' num2str(ii-1) '/' num2str(size(xHistory,3))]);
104 -     if ((ii>1) & showLines)
105 -         line([xHistory(:,1,ii-1);xHistory(:,1,ii)], [xHistory(:,2,ii-1);xHistory(:,2,ii)], 'Color',[0.95,0.95,0.95]);
106 -     end
107 -     uistack(sc,'top');
108 -     if captureVideo
109 -         % frm = print('-RGBImage','-r0','-opengl','-noui');
110 -         % frm = frm(72:700,180:1050,:);
111 -         frames(:,:,nextFrmInQueue) = frm;
112 -         nextFrmInQueue = nextFrmInQueue+1;
113 -         if (nextFrmInQueue==21)
114 -             writeVideo(vid,frames/255);
115 -             nextFrmInQueue = 1;
116 -         end
117 -         end
118 -         pause(0.05);
119 -     end
120 -     if captureVideo
121 -         frames(:,:,nextFrmInQueue) = frm;
122 -         writeVideo(vid,frames(:,:,1:nextFrmInQueue)/255);
123 -         close(vid);
124 -     end
125 -     figure;
126 -     axis equal;
127 -     scatter(xHistory(:,1,1),xHistory(:,2,1),10,clrs,'filled');
128 -     xlim([min(min(xHistory(:,1,:))) , max(max(xHistory(:,1,:)))]);
129 -     ylim([min(min(xHistory(:,2,:))) , max(max(xHistory(:,2,:)))]);
130 -     % for ii=1:clusters
131 -     %     text(mean(xHistory(clusters==ii,1)),mean(xHistory(clusters==ii,2)),num2str(ii), 'BackgroundColor','w','EdgeColor','k','HorizontalAlignment','center','FontSize',12);
132 -     % end
133 -     %
134 - else
135 -     if ((size(xHistory,2) > 3) && (nargin<3))
136 -         [principalComponents,dataInPCbasis,~] = pca(xHistory(:,1:3),'Centered',false);
137 -         n = size(xHistory,2);
138 -         newXHistory = zeros(size(xHistory,1),size(dataInPCbasis,2),size(xHistory,3));
139 -         for ii=1:size(xHistory,3)
140 -             newXHistory(:,ii) = xHistory(:,ii)*principalComponents;
141 -         end
142 -         xHistory = newXHistory;
143 -     end
144 -     figure;
145 -     axis equal;
146 -     clrs = jet(max(clusters));
147 -     [~,~,tmp] = unique(clusters);
148 -     clrs = clrs(tmp,:);
149 -     sc = scatter3(xHistory(:,1,1),xHistory(:,2,1),xHistory(:,3,1),10,clrs,'filled');
150 -     xlim([min(min(xHistory(:,1,:))) , max(max(xHistory(:,1,:)))]);
151 -     ylim([min(min(xHistory(:,2,:))) , max(max(xHistory(:,2,:)))]);
152 -     zlim([min(min(xHistory(:,3,:))) , max(max(xHistory(:,3,:)))]);
153 -     if captureVideo
154 -         % frm = print('-RGBImage','-r0','-opengl','-noui');
155 -         % frm = frm(250:570,:,:);
156 -         frames = zeros(size(frm,1),size(frm,2),3,20);
157 -         nextFrmInQueue = 1;
158 -         frames(:,:,nextFrmInQueue) = frm;
159 -         nextFrmInQueue = nextFrmInQueue+1;
160 -         vid = VideoWriter('filename.mp4','MPEG-4');
161 -         vid.FrameRate = 18;
162 -         open(vid);
163 -     end
164 -     for ii=1:(size(xHistory,3)-1)
165 -         set(sc,'XData',xHistory(:,1,ii),'YData',xHistory(:,2,ii),'ZData',xHistory(:,3,ii));
166 -         title(['step #' num2str(ii-1) '/' num2str(size(xHistory,3)-1)]);
167 -         if ((ii>1) & showLines)
168 -             line([xHistory(:,1,ii-1);xHistory(:,1,ii)], [xHistory(:,2,ii-1);xHistory(:,2,ii)], [xHistory(:,3,ii-1);xHistory(:,3,ii)], 'Col
169 -         end
170 -         uistack(sc,'top');
171 -         if captureVideo

```



```

171 - % frm = print('-RGBImage','-r0','-opengl','-noui');
172 - frm = frm(72:700,180:1050,:);
173 - frames(:,:,nextFrmInQueue) = frm;
174 - nextFrmInQueue = nextFrmInQueue+1;
175 - if (nextFrmInQueue==21)
176 -     writeVideo(vid,frames/255);
177 -     nextFrmInQueue = 1;
178 - end
179 - end
180 - pause(0.05);
181 - end
182 - if captureVideo
183 -     frames(:,:,nextFrmInQueue) = frm;
184 -     writeVideo(vid,frames(:,:,1:nextFrmInQueue)/255);
185 -     close(vid);
186 - end
187 - figure;
188 - axis equal;
189 - scatter3(xHistory(:,1,1),xHistory(:,2,1),xHistory(:,3,1),10,clr,'filled');
190 - axis equal;
191 - xlim([min(min(xHistory(:,1,:))), max(max(xHistory(:,1,:)))]);
192 - ylim([min(min(xHistory(:,2,:))), max(max(xHistory(:,2,:)))]);
193 - zlim([min(min(xHistory(:,3,:))), max(max(xHistory(:,3,:)))]);
194 - clr = jet(max(clusters));
195 - for ii=1:max(clusters)
196 -     text(mean(xHistory(clusters==ii,1,1)),mean(xHistory(clusters==ii,2,1)),mean(xHistory(clusters==ii,3,1)),num2str(ii),'BackgroundColor','w',
197 -     text(mean(xHistory(clusters==ii,1,1)),mean(xHistory(clusters==ii,2,1)),mean(xHistory(clusters==ii,3,1)),num2str(sum(clusters==ii)), 'Backg
198 - end
199 - end
200 -
201 - % if (size(xHistory,2) > 1)
202 - %     figure;
203 - %     clr = jet(max(clusters));
204 - %     hold on;
205 - %     for ii=1:max(clusters)
206 - %         plot(xHistory(clusters==ii,:),1,'Color',clr(ii,:));
207 - %     end
208 - % end
209 -
210 - end

```

- FindApproximateEntropy

```

1 < norm.m x DisplayQC.m x FindApproximateEntropy.m x FindApproximatePotential.m x FindApproximateStochasticEntropy.m x FindApproximateStochasticPotential.m x FindApproximateStochasticWaveFunction.m > +
2 function [S,dS] = FindApproximateEntropy(data,coeff,sigma,x)
3 % finds the approximate entropy and its gradient for quantum clustering
4 % data - matrix with data. each row corresponds to one data point.
5 % coeff - a weight for each row in data
6 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
7 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
8 % S - the entropy. It is a column vector with size(x,1) elements.
9 % dS - the gradient of the entropy at the points x. It has the same size as x.
10 -
11 - if isempty(x)
12 -     x = data;
13 - end
14 -
15 - S = zeros(size(x,1),1);
16 - dS = zeros(size(x));
17 - for ii=1:size(x,1)
18 -     difference = (repmat(x(ii,:),size(data,1),1) - data);
19 -     squaredDifference = sum(difference.^2,2);
20 -     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
21 -     laplacian = sum(coeff.*gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
22 -     parzen = sum(coeff.*gaussian);
23 -     V = (1/(2*sigma^2))*laplacian/parzen;
24 -     S(ii) = V + log(abs(parzen));
25 -
26 -     dS(ii,:) = (1/parzen)*sum(difference.*repmat(coeff.*gaussian,1,size(data,2)).*(2*sigma^2*V-repmat(squaredDifference,1,size(data,2))));
27 -
28 - end

```

- FindApproximatePotential

```

1 < norm.m x DisplayQC.m x FindApproximateEntropy.m x FindApproximatePotential.m x FindApproximateStochasticEntropy.m x FindApproximateStochasticPotential.m x FindApproximateStochasticWaveFunction.m > +
2 function [V,dV] = FindApproximatePotential(data,coeff,sigma,x)
3 % finds the approximate potential and its gradient for quantum clustering
4 % data - matrix with data. each row corresponds to one data point.
5 % coeff - a weight for each row in data
6 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
7 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
8 % V - the potential that gives the Parzen wavefunction as an eigenfunction. It is a column vector with size(x,1) elements.
9 % dV - the gradient of the potential at the points x. It has the same size as x.
10 -
11 - if isempty(x)
12 -     x = data;
13 - end
14 -
15 - V = zeros(size(x,1),1);
16 - dV = zeros(size(x));
17 - for ii=1:size(x,1)
18 -     difference = (repmat(x(ii,:),size(data,1),1) - data);
19 -     squaredDifference = sum(difference.^2,2);
20 -     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
21 -     laplacian = sum(coeff.*gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
22 -     parzen = sum(coeff.*gaussian);
23 -     V(ii) = 1+(1/(2*sigma^2))*laplacian/parzen;
24 -
25 -     dV(ii,:) = (1/parzen)*sum(difference.*repmat(coeff.*gaussian,1,size(data,2)).*(2*sigma^2*V(ii)-repmat(squaredDifference,1,size(data,2))));
26 -
27 - end
28 - V = V-1;
29 - end

```

- FindApproximateStochasticEntropy

```

1 function [S,dS] = FindApproximateStochasticEntropy(data,coeff,sigma,x,sz)
2 % finds the estimated approximate entropy and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % coeff - a weight for each row in data
5 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
6 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
7 % sz - a number between 0 and 1. The entropy and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
8 % S - the entropy. It is a column vector with size(x,1) elements.
9 % dS - the gradient of the entropy at the points x. It has the same size as x.
10
11 if isempty(x)
12     x = data;
13 end
14 sz = cell(sz*size(data,1));
15
16 S = zeros(size(x,1),1);
17 dS = zeros(size(x));
18 for ii=1:size(x,1)
19     inds = randi(size(data,1),sz,1);
20     currentData = data(inds,:);
21
22     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
23     squaredDifference = sum(difference.^2,2);
24     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
25     laplacian = sum(coeff.*gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
26     parzen = sum(coeff.*gaussian);
27     V = (1/(2*sigma^2))*laplacian/parzen;
28     S(ii) = V + log(abs(parzen));
29
30     dS(ii,:) = (1/parzen)*sum(difference.*repmat(coeff.*gaussian,1,size(currentData,2)).*(2*sigma^2*V-repmat(squaredDifference,1,size(currentData,2))))
31
32 end
33 end

```

- FindApproximateStochasticPotential

```

1 function [V,dV] = FindApproximateStochasticPotential(data,coeff,sigma,x,sz)
2 % finds the estimation to the approximate potential and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % coeff - a weight for each row in data
5 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
6 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
7 % sz - a number between 0 and 1. The potential and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
8 % V - the potential that gives the Parzen wavefunction as an eigenfunction. It is a column vector with size(x,1) elements.
9 % dV - the gradient of the potential at the points x. It has the same size as x.
10
11 if isempty(x)
12     x = data;
13 end
14 sz = cell(sz*size(data,1));
15
16 V = zeros(size(x,1),1);
17 dV = zeros(size(x));
18 for ii=1:size(x,1)
19     inds = randi(size(data,1),sz,1);
20     currentData = data(inds,:);
21
22     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
23     squaredDifference = sum(difference.^2,2);
24     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
25     laplacian = sum(coeff.*gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
26     parzen = sum(coeff.*gaussian);
27     V(ii) = 1*(1/(2*sigma^2))*laplacian/parzen;
28
29     dV(ii,:) = (1/parzen)*sum(difference.*repmat(coeff.*gaussian,1,size(currentData,2)).*(2*sigma^2*V(ii)-repmat(squaredDifference,1,size(currentData,2))))
30
31 end
32 V = V-1;
33 end

```

- FindApproximateStochasticWaveFunction

```

1 function [Psi,dPsi] = FindApproximateStochasticWaveFunction(data,coeff,sigma,x,sz)
2 % finds the estimated approximate wave function/Parzen function and its gradient.
3 % data - matrix with data. each row corresponds to one data point.
4 % coeff - a weight for each row in data
5 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
6 % x - matrix with points where the wave function will be evaluated. Each row is a point. If x is empty, then x = data.
7 % sz - a number between 0 and 1. The wave function and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
8 % V - the wave function. It is a column vector with size(x,1) elements.
9 % dV - the gradient of the wave function at the points x. It has the same size as x.
10
11 if isempty(x)
12     x = data;
13 end
14 sz = cell(sz*size(data,1));
15
16 Psi = zeros(size(x,1),1);
17 dPsi = zeros(size(x));
18 for ii=1:size(x,1)
19     inds = randi(size(data,1),sz,1);
20     currentData = data(inds,:);
21
22     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
23     squaredDifference = sum(difference.^2,2);
24     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
25     Psi(ii) = sum(coeff.*gaussian);
26
27     dPsi(ii,:) = -1*sum(difference.*repmat(coeff.*gaussian,1,size(currentData,2))*2*sigma^2);
28
29 end
30
31 end

```

- FindApproximateWaveFunction

```

1 function [Psi,dPsi] = FindApproximateWaveFunction(data,coeff,sigma,x)
2 % finds the approximate wave function\Parzen function and its gradient.
3 % data - matrix with data. each row corresponds to one data point.
4 % coeff - a weight for each row in data
5 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
6 % x - matrix with points where the wave function will be evaluated. Each row is a point. If x is empty, then x = data.
7 % V - the wave function. It is a column vector with size(x,1) elements.
8 % dV - the gradient of the wave function at the points x. It has the same size as x.
9
10
11 if isempty(x)
12     x = data;
13 end
14
15 Psi = zeros(size(x,1),1);
16 dPsi = zeros(size(x));
17 for ii=1:size(x,1)
18     difference = (repmat(x(ii,:),size(data,1),1) - data);
19     squaredDifference = sum(difference.^2,2);
20     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
21     Psi(ii) = sum(coeff.*gaussian);
22
23     dPsi(ii,:) = -1*sum(difference.*repmat(coeff.*gaussian,1,size(data,2))*2*sigma^2);
24
25 end
26 end

```

- FindEntropy

```

1 function [S,dS] = FindEntropy(data,sigma,x)
2 % finds the entropy and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
6 % S - the entropy. It is a column vector with size(x,1) elements.
7 % dS - the gradient of the entropy at the points x. It has the same size as x.
8
9 if isempty(x)
10     x = data;
11 end
12
13 S = zeros(size(x,1),1);
14 dS = zeros(size(x));
15 for ii=1:size(x,1)
16     difference = (repmat(x(ii,:),size(data,1),1) - data);
17     squaredDifference = sum(difference.^2,2);
18     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
19     laplacian = sum(gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
20     parzen = sum(gaussian);
21     V = (1/(2*sigma^2))*laplacian/parzen;
22     S(ii) = V + log(abs(parzen));
23
24     dS(ii,:) = (1/parzen)*sum(difference.*repmat(gaussian,1,size(data,2)).*(2*sigma^2*V-repmat(squaredDifference,1,size(data,2))));
25
26 end
27 end

```

- FindEntropyStochastic

```

1 function [S,dS] = FindEntropyStochastic(data,sigma,x,sz)
2 % finds the estimation for the entropy and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
6 % sz - a number between 0 and 1. The entropy and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
7 % S - the entropy. It is a column vector with size(x,1) elements.
8 % dS - the gradient of the entropy at the points x. It has the same size as x.
9
10
11 if isempty(x)
12     x = data;
13 end
14 sz = ceil(sz*size(data,1));
15
16 S = zeros(size(x,1),1);
17 dS = zeros(size(x));
18 for ii=1:size(x,1)
19     inds = randi(size(data,1),sz,1);
20     currentData = data(inds,:);
21
22     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
23     squaredDifference = sum(difference.^2,2);
24     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
25     laplacian = sum(gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
26     parzen = sum(gaussian);
27     V = (1/(2*sigma^2))*laplacian/parzen;
28     S(ii) = V + log(abs(parzen));
29
30     dS(ii,:) = (1/parzen)*sum(difference.*repmat(gaussian,1,size(currentData,2)).*(2*sigma^2*V-repmat(squaredDifference,1,size(currentData,2))));
31
32 end
33 end

```

- FindPotential

```

1 function [V,dV] = FindPotential(data,sigma,x)
2 % finds the potential and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
6 % V - the potential that gives the Parzen wavefunction as an eigenfunction. It is a column vector with size(x,1) elements.
7 % dV - the gradient of the potential at the points x. It has the same size as x.
8
9
10
11 if isempty(x)
12     x = data;
13 end
14
15 V = zeros(size(x,1),1);
16 dV = zeros(size(x));
17 for ii=1:size(x,1)
18     difference = (repmat(x(ii,:),size(data,1),1) - data);
19     squaredDifference = sum(difference.^2,2);
20     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
21     laplacian = sum(gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
22     parzen = sum(gaussian);
23     V(ii) = 1/(1/(2*sigma^2))*laplacian/parzen;
24
25     dV(ii,:) = (1/parzen)*sum(difference.*repmat(gaussian,1,size(data,2)).*(2*sigma^2*V(ii)-repmat(squaredDifference,1,size(data,2))));
26
27 end
28
29 V = V-1;
30 end

```

• FindPotentialStochastic

```

1 function [V,dV] = FindPotentialStochastic(data,sigma,x,sz)
2 % finds the estimated potential and its gradient for quantum clustering
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the potential will be evaluated. Each row is a point. If x is empty, then x = data.
6 % sz - a number between 0 and 1. The potential and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
7 % V - the potential that gives the Parzen wavefunction as an eigenfunction. It is a column vector with size(x,1) elements.
8 % dV - the gradient of the potential at the points x. It has the same size as x.
9
10
11 if isempty(x)
12     x = data;
13 end
14
15 sz = ceil(sz*size(data,1));
16
17 V = zeros(size(x,1),1);
18 dV = zeros(size(x));
19 for ii=1:size(x,1)
20     inds = randi(size(data,1),sz,1);
21     currentData = data(inds,:);
22
23     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
24     squaredDifference = sum(difference.^2,2);
25     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
26     laplacian = sum(gaussian.*squaredDifference); % this is not the true Laplacian, since I omit a constant additive term from the potential
27     parzen = sum(gaussian);
28     V(ii) = 1/(1/(2*sigma^2))*laplacian/parzen;
29
30     dV(ii,:) = (1/parzen)*sum(difference.*repmat(gaussian,1,size(currentData,2)).*(2*sigma^2*V(ii)-repmat(squaredDifference,1,size(currentData,2))));
31
32 end
33
34 V = V-1;
35 end

```

• FindWaveFunction

```

1 function [Psi,dPsi] = FindWaveFunction(data,sigma,x)
2 % finds the wave function\Parzen function and its gradient.
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the wave function will be evaluated. Each row is a point. If x is empty, then x = data.
6 % V - the wave function. It is a column vector with size(x,1) elements.
7 % dV - the gradient of the wave function at the points x. It has the same size as x.
8
9
10 if isempty(x)
11     x = data;
12 end
13
14 Psi = zeros(size(x,1),1);
15 dPsi = zeros(size(x));
16 for ii=1:size(x,1)
17     difference = (repmat(x(ii,:),size(data,1),1) - data);
18     squaredDifference = sum(difference.^2,2);
19     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
20     Psi(ii) = sum(gaussian);
21
22     dPsi(ii,:) = -1*sum(difference.*repmat(gaussian,1,size(data,2))*2*sigma^2);
23
24 end
25
26 end

```


- FindWaveFunctionStochastic

```

1 function [Psi,dPsi] = FindWaveFunctionStochastic(data,sigma,x,sz)
2 % finds the estimated wave function/Parzen function and its gradient, using a sample of data points.
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % x - matrix with points where the wave function will be evaluated. Each row is a point. If x is empty, then x = data.
6 % sz - a number between 0 and 1. The wave function and gradient will be calculated based on a random sample of size 'sz*size(data,1)' of data points
7 % V - the wave function. It is a column vector with size(x,1) elements.
8 % dv - the gradient of the wave function at the points x. It has the same size as x.
9
10
11 if isempty(x)
12     x = data;
13 end
14 sz = cell(sz*size(data,1));
15
16 Psi = zeros(size(x,1),1);
17 dPsi = zeros(size(x,1));
18 for ii=1:size(x,1)
19     inds = randi(size(data,1),sz,1);
20     currentData = data(inds,:);
21
22     difference = (repmat(x(ii,:),size(currentData,1),1) - currentData);
23     squaredDifference = sum(difference.^2,2);
24     gaussian = exp(-(1/(2*sigma^2))*squaredDifference);
25     Psi(ii) = sum(gaussian);
26
27     dPsi(ii,:) = -1*sum(difference.*repmat(gaussian,1,size(currentData,2))*2*sigma^2);
28 end
29
30 end

```

- getApproximateWaveFunction

```

1 function [newData,coeff] = getApproximateWaveFunction(data,sigma,voxelSize)
2 % finds one representative data point for each voxel, and assigns a weight to each such data point.
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % voxelSize - scalar, the size of the size of one voxel.
6 % out:
7 % newData - matrix. A new set of data points, with only one data point per voxel.
8 % coeff - vector, same size as size(newData,1). The weight\coefficient of the corresponding new data point.
9
10 newData = unique(floor(data/voxelSize)*voxelSize+voxelSize/2,'rows');
11
12 N = zeros(numel(newData));
13 N = squareform(pdist(newData)).^2;
14 N = -N;
15 N = exp(N/(4*sigma^2));
16
17 M = zeros(numel(newData),numel(data));
18 M = pdist2(newData,data).^2;
19 M = -M;
20 M = exp(M/(4*sigma^2));
21
22 C = N\M;
23
24 coeff = sum(C,2);
25
26 coeff = size(data,1)*coeff/sum(coeff);
27
28 end

```

- myColorMap

```

1 function [m]=myColorMap(seuil,varargin)
2 % mycolormap - custom the jet colormap to display X% of the data
3 % [m]=mycolormap(X)
4 % [m]=mycolormap(X,CM) CM is a colormap
5
6 if nargin==1
7
8     m=ones(100-seuil,3)*[.2 .7 1;0 0 0;0 0 0];
9     m = [m ; ones(2*seuil,3)*.5];
10    m = [m ; ones(100-seuil,3)*[1 .4 .2;0 0 0;0 0 0]];
11
12    l=.3;
13    m=hot(110);
14    m = [m(1:100,:)]*(1-l)+ones(100,3)*l;
15    m = [fliplr(flipud(m)) ; ones(2*seuil,3)*l ; m];
16
17 else
18     n=varargin{1};
19     m=[ n(1:ceil(length(n)/2),:); ones(2*seuil/100*length(n),3)*.5 ; n(ceil(length(n)/2):end,:);];
20
21 end

```

- PerformFinalClustering

```

1 function clusters = PerformFinalClustering(data,stepSize,th)
2 % final clustering of data after performing quantum clustering.
3 % data - matrix, the result of performing QC
4 % stepSize - the step size used by QC. this step size is the resolution of movement of the replicas in gradient descent, so we cluster together points that are with
5 % th - we cluster together points that are within a distance of the 'stepSize*th' from each other.
6 % out:
7 % clusters - a vector with the same number of rows as 'data', such that 'clusters(i1)' is the cluster of 'data(i1,:)'. The clusters' numbers are ordered from largest
8
9 if ~exist('th','var') || isempty(th) || (th==0)
10     th = 3;
11 end
12 clusters = zeros(size(data,1),1);
13 ii = 1;
14 c = 1;
15 distances = squareform(pdist(data));
16 while ~isempty(ii)
17     inds = find(clusters==0);
18     clusters(inds(distances(ii,inds) <= th*stepSize)) = c;
19     c = c+1;
20     ii = find(clusters==0,1,'first');
21 end
22
23 [~,inds] = sort(accumarray(clusters,1),'descend');
24 [~,inds] = sort(inds);
25 clusters = inds(clusters);
26 end

```

- PerformGDQC**

```

1 function [x,xHistory] = PerformGDQC(data,sigma,rep,stepSize,clusteringType,recalculatePotential,normalizeData,dataInitialPosition,normalizeGradient,stochasticSz
2 % performs gradient descent on data using potential from quantum clustering.
3 % data - matrix with data. each row corresponds to one data point.
4 % sigma - scalar, the parameter that appears in the Parzen wavefunction.
5 % rep - scalar, maximal number of steps of gradient descent. default: 200
6 % stepSize - scalar. gradient will be multiplied by this number to perform each gradient descent step. default: 'sigma/7'
7 % clusteringType - char, either 'V','S' or 'P' for minimizing the potential, maximizing the entropy or maximizing the wavefunction. default: 'S'
8 % recalculatePotential - boolean. The gradient on each step will be derived from the potential using the current replica points (not initial data points). default
9 % normalizeData - boolean. Each data point will be normalized to norm 1 on each gradient descent step. default: false
10 % dataInitialPosition - matrix with same column number as 'data'. Each row is a point to be moved by the gradients. If empty, 'data' will be used. default: 'data'
11 % normalizeGradient - boolean. The gradient is normalized to unit norm before multiplied by 'stepSize'. default: 'data'
12 % stochasticSz - scalar. If this is non-empty, a stochastic version of the algorithm will be used, in which the gradient is calculated based on a random sample
13 % howOftenToTestIfDone - scalar. once in how many steps to check whether a point is done moving. This is done by comparing the current value of the potential/ent
14 % out:
15 % x - the replica points after gradient descent evolution.
16 % optional output: xHistory - a 3d matrix with size [size(x,1),size(x,2),rep]. xHistory(:, :, i1) contains the points after the (i1-1)'th step of gradient descent
17
18
19 if ~exist('stepSize','var') || isempty(stepSize)
20     stepSize = sigma/7;
21 end
22
23 if ~exist('rep','var') || isempty(rep)
24     rep = 200;
25 end
26
27 if ~exist('clusteringType','var') || isempty(clusteringType)
28     clusteringType = 'V';
29 end
30
31
32 if ~exist('recalculatePotential','var') || isempty(recalculatePotential)
33     recalculatePotential = false;
34 end
35
36 if ~exist('normalizeData','var') || isempty(normalizeData)
37     normalizeData = false;
38 end
39
40 if ~exist('dataInitialPosition','var') || isempty(dataInitialPosition)
41     dataInitialPosition = data;
42 end
43
44 if ~exist('normalizeGradient','var') || isempty(normalizeGradient)
45     normalizeGradient = true;
46 end
47
48 if ~exist('stochasticSz','var')
49     stochasticSz = [];
50 end
51
52 if ~exist('voxelSize','var')
53     voxelSize = [];
54 end
55
56 if ~exist('howOftenToTestIfDone','var') || isempty(howOftenToTestIfDone)
57     howOftenToTestIfDone = 1;
58 end
59
60
61 switch clusteringType
62     case 'V'
63         maximizeEntropy = false;
64         maximizeWaveFunction = false;
65     case 'S'
66         maximizeEntropy = true;
67         maximizeWaveFunction = false;
68     case 'P'
69         maximizeEntropy = false;
70         maximizeWaveFunction = true;
71 end
72
73 if ~isempty(voxelSize)
74     [data,coeff] = getApproximateWaveFunction(data,sigma,voxelSize);
75 end
76
77 x = dataInitialPosition;
78
79 if normalizeData
80     data = normr(data);
81     x = normr(x);
82 end

```

```

83 - if (nargout>2)
84 -     xHistory = zeros(size(x,1),size(x,2),rep+1);
85 -     xHistory(:,:,1) = x;
86 - end
87 -
88 - % initialize prevV to have the previous values of V\psi, so that can be compared to current values on each step of algorithm to check if arrived at extr
89 - prevV = inf(size(x,1),1);
90 -
91 - % these are the indices of replicas that are currently still moving
92 - inds = 1:size(x,1);
93 -
94 - % master loop
95 - for ii=1:rep
96 -     if isempty(voxelSize)
97 -         if recalculatePotential
98 -             if maximizeEntropy
99 -                 if isempty(stochasticSz)
100 -                     [S,dx] = FindEntropy(x,sigma,x);
101 -                 else
102 -                     [S,dx] = FindEntropyStochastic(x,sigma,x,stochasticSz);
103 -                 end
104 -                 if normalizeGradient
105 -                     dx = normr(dx);
106 -                 end
107 -                 x = x + stepSize*dx;
108 -                 V = -1*S;
109 -             elseif maximizeWaveFunction
110 -                 if isempty(stochasticSz)
111 -                     [P,dx] = FindWaveFunction(x,sigma,x);
112 -                 else
113 -                     [P,dx] = FindWaveFunctionStochastic(x,sigma,x,stochasticSz);
114 -                 end
115 -                 if normalizeGradient
116 -                     dx = normr(dx);
117 -                 end
118 -                 x = x + stepSize*dx;
119 -                 V = -1*P;
120 -             else
121 -                 if isempty(stochasticSz)
122 -                     [V,dx] = FindPotential(x,sigma,x);
123 -                 else
124 -                     [V,dx] = FindPotentialStochastic(x,sigma,x,stochasticSz);
125 -                 end
126 -                 if normalizeGradient
127 -                     dx = normr(dx);
128 -                 end
129 -                 x = x - stepSize*dx;
130 -                 V = V;
131 -             end
132 -         else
133 -             if maximizeEntropy
134 -                 if isempty(stochasticSz)
135 -                     [S,dx] = FindEntropy(data,sigma,x(inds,:));
136 -                 else
137 -                     [S,dx] = FindEntropyStochastic(data,sigma,x(inds,:),stochasticSz);
138 -                 end
139 -                 if normalizeGradient
140 -                     dx = normr(dx);
141 -                 end
142 -                 x(inds,:) = x(inds,:) + stepSize*dx;
143 -                 V = -1*S;
144 -             elseif maximizeWaveFunction
145 -                 if isempty(stochasticSz)
146 -                     [P,dx] = FindWaveFunction(data,sigma,x(inds,:));
147 -                 else
148 -                     [P,dx] = FindWaveFunctionStochastic(data,sigma,x(inds,:),stochasticSz);
149 -                 end
150 -                 if normalizeGradient
151 -                     dx = normr(dx);
152 -                 end
153 -                 x(inds,:) = x(inds,:) + stepSize*dx;
154 -                 V = -1*P;
155 -             else
156 -                 if isempty(stochasticSz)
157 -                     [V,dx] = FindPotential(data,sigma,x(inds,:));
158 -                 else
159 -                     [V,dx] = FindPotentialStochastic(data,sigma,x(inds,:),stochasticSz);
160 -                 end
161 -                 if normalizeGradient
162 -                     dx = normr(dx);
163 -                 end
164 -                 x(inds,:) = x(inds,:) - stepSize*dx;
165 -                 V = V;
166 -             end
167 -         end
168 -     else
169 -         if recalculatePotential
170 -             if maximizeEntropy
171 -                 if isempty(stochasticSz)
172 -                     [S,dx] = FindApproximateEntropy(x,coeff,sigma,[x;data]);
173 -                 else
174 -                     [S,dx] = FindApproximateStochasticEntropy(x,coeff,sigma,[x;data],stochasticSz);
175 -                 end
176 -                 if normalizeGradient
177 -                     dx = normr(dx);
178 -                 end
179 -                 x = x + stepSize*dx;
180 -                 V = -1*S;
181 -             elseif maximizeWaveFunction
182 -                 if isempty(stochasticSz)
183 -                     [P,dx] = FindApproximateWaveFunction(x,coeff,sigma,[x;data]);
184 -                 else
185 -                     [P,dx] = FindApproximateStochasticWaveFunction(x,coeff,sigma,[x;data],stochasticSz);
186 -                 end
187 -                 if normalizeGradient
188 -                     dx = normr(dx);
189 -                 end
190 -                 x = x + stepSize*dx;
191 -                 V = -1*P;
192 -             else
193 -

```

```

194 -         if isempty(stochasticSz)
195 -             [V,dx] = FindApproximatePotential(x,coeff,sigma,[x;data]);
196 -         else
197 -             [V,dx] = FindApproximateStochasticPotential(x,coeff,sigma,[x;data],stochasticSz);
198 -         end
199 -         if normalizeGradient
200 -             dx = normr(dx);
201 -         end
202 -         x = x - stepSize*dx;
203 -         V = V;
204 -     end
205 - else
206 -     if maximizeEntropy
207 -         if isempty(stochasticSz)
208 -             [S,dx] = FindApproximateEntropy(data,coeff,sigma,x(inds,:));
209 -         else
210 -             [S,dx] = FindApproximateStochasticEntropy(data,coeff,sigma,x(inds,:),stochasticSz);
211 -         end
212 -         if normalizeGradient
213 -             dx = normr(dx);
214 -         end
215 -         x(inds,:) = x(inds,:) + stepSize*dx;
216 -         V = -1*S;
217 -     elseif maximizeWaveFunction
218 -         if isempty(stochasticSz)
219 -             [P,dx] = FindApproximateWaveFunction(data,coeff,sigma,x(inds,:));
220 -         else
221 -             [P,dx] = FindApproximateStochasticWaveFunction(data,coeff,sigma,x(inds,:),stochasticSz);
222 -         end
223 -         if normalizeGradient
224 -             dx = normr(dx);
225 -         end
226 -         x(inds,:) = x(inds,:) + stepSize*dx;
227 -         V = -1*P;
228 -     else
229 -         if isempty(stochasticSz)
230 -             [V,dx] = FindApproximatePotential(data,coeff,sigma,x(inds,:));
231 -         else
232 -             [V,dx] = FindApproximateStochasticPotential(data,coeff,sigma,x(inds,:),stochasticSz);
233 -         end
234 -         if normalizeGradient
235 -             dx = normr(dx);
236 -         end
237 -         x(inds,:) = x(inds,:) - stepSize*dx;
238 -         V = V;
239 -     end
240 - end
241 - end
242 - if normalizeData
243 -     x(inds,:) = normr(x(inds,:));
244 - end
245 - if (nargout>2)
246 -     xHistory(:,ii+1) = x;
247 - end
248 -
249 - % check if points can stop. This condition makes sense only when V\S\Psi are fixed, not changing with time as in the case of recalculatePotential
250 - if (~recalculatePotential) && (mod(ii,howOftenToTestIfDone)==0)
251 -     prevInds = inds;
252 -     inds = inds(prevV(inds)*V);
253 -     if isempty(inds)
254 -         break;
255 -     end
256 -     inds = 1:size(x,1);
257 -     prevV(prevInds) = V;
258 - end
259 - if (nargout>2)
260 -     xHistory = xHistory(:,1:(ii+1));
261 - end
262 - end
263 -
264 - end

```

• QC

```

1 % load example data set. This data set is generated by the black and white image "data.bmp", and the image can be edited using Microsoft paint.
2 n = 500;
3 sigma = 1;
4 %im = imread('data.bmp');
5 %im = flipud(im);
6 im = importdata('Absenteeism_at_work2.csv', ',', 1)
7 im = im_data
8 [row,col] = find(im == 0);
9 data = [col(:),row(:)];
10 rng(111);
11 data = data(randi(size(data,1),n,1),:);
12 data = data+sigma*randn(size(data));
13 data = unique(data,'rows')-1;
14
15 % parameters
16 sigma = 5; % parameter of QC - width of gaussians
17 rep = 300; % number of steps for gradient descent
18 stepSize = sigma/7; % step size for gradient descent
19 clusteringType = 'V'; % either 'V','S','P' for minimizing potential/maximizing entropy/maximizing wave function
20 recalculatePotential = false; % whether to use new replicas location for calculating a new wavefunction each step?
21 normalizeData = false; % whether to normalize data to the unit sphere on each step?
22 normalizeGradient = true; % whether to normalize gradient unit length before multiplying by step size?
23 voxelSize = []; % voxel size for approx. QC. can be [] for the non-approx algorithm
24 stochasticSz = []; % gradient will be estimated based on this relative part of the data points, chosen at random each step for each replica. can be [] for the non
25 howOftenToTestIfDone = 1; % how often to perform a check of whether stop condition of replica is fulfilled? This could be 1, unless using stochasticSz and then t
26
27 [x,xHistory] = PerformQOC(data,sigma,rep,stepSize,clusteringType,recalculatePotential,normalizeData,[],normalizeGradient,stochasticSz,voxelSize,howOftenToTestIfD
28 clusters = PerformFinalClustering(x,stepSize);
29
30
31
32 DisplayQC(xHistory,clusters);

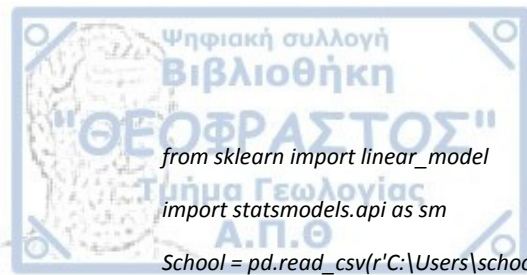
```

Appendix F: Multiple Linear Regression (Python)

import pandas as pd

from pandas import DataFrame

import matplotlib.pyplot as plt



```
from sklearn import linear_model
import statsmodels.api as sm
```

```
School = pd.read_csv(r'C:\Users\schoo12.csv', sep=",")
```

```
df = DataFrame(School,
columns=['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2', 'G3'])
```

```
print (df)
```

```
#Checkng Linearity
```

```
plt.scatter(df['G1'], df['G3'], color='red')
```

```
plt.title('G3 Vs G1', fontsize=14)
```

```
plt.xlabel('G1', fontsize=14)
```

```
plt.ylabel('G3', fontsize=14)
```

```
plt.grid(True)
```

```
plt.show()
```

```
plt.scatter(df['G2'], df['G3'], color='green')
```

```
plt.title('G3 Vs G2', fontsize=14)
```

```
plt.xlabel('G2', fontsize=14)
```

```
plt.ylabel('G3', fontsize=14)
```

```
plt.grid(True)
```

```
plt.show()
```

```
#Multiple Linear Regression
```

X = df[['G1', 'G2']] # here we have 2 variables for multiple regression. If you just want to use one variable for simple linear regression, then use X = df['Interest_Rate'] for example. Alternatively, you may add additional variables within the brackets

```
Y = df['G3']
```

```
# with sklearn
```

```
regr = linear_model.LinearRegression()
```

```
regr.fit(X, Y)
```

```
print('Intercept: \n', regr.intercept_)
```

```
print('Coefficients: \n', regr.coef_)
```

```
# prediction with sklearn
```

```
New_G1 = 0.15
```

```
New_G2 = 0.90
```

```
print ('Predicted G3: \n', regr.predict([[New_G1, New_G2]]))
```

Appendix G: Logistic Regression (Python)

```
import pandas as pd
```

```
import numpy as np
```

```
from pandas import DataFrame
```

```
from sklearn import preprocessing
```

```
import matplotlib.pyplot as plt
```

```
plt.rc("font", size=14)
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.cross_validation import train_test_split
```

```
from sklearn.feature_selection import RFE
```

```
import seaborn as sns
```

```
sns.set(style="white")
```

```
sns.set(style="whitegrid", color_codes=True)
```

```
School = pd.read_csv(r'C:\Users\school2\Log.csv', sep=";")
```

```
df = DataFrame(School,
columns=['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2', 'y'])
```

```
print (df)
```

```
df = df.dropna()
```

```
print(df.shape)
```

```
print(list(df.columns))
```

```
df.head()
```

```
#Create dummy variables
```

```
#That is variables with only two values, zero and one.
```

```
cat_vars=['school', 'sex', 'age', 'add2ess', 'famsize', 'Ps121us', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2']
```



```

for var in cat_vars:
    cat_list=var+'_'+var
    cat_list = pd.get_dummies(df[var], prefix=var)

    data1=df.join(cat_list)

    df=data1

cat_vars=['school','sex','age','add2ess','famsize','Ps121us','Medu','Fedu','Mjob','Fjob','reason','guardian','traveltime',
'studytime','failures','schoolsup','famsup','paid','activities','nursery','higher','internet','romantic','famrel','freetime',
'goout','Dalc','Walc','health','absences','G1','G2']

df_vars=df.columns.values.tolist()

to_keep=[i for i in df_vars if i not in cat_vars]

#Our final data columns will be:

df_final=df[to_keep]

df_final.columns.values

prosorin = df_final.columns.values

print(prosorin.shape)

prosorin[0]

prosorin[40]

prosorin[54]

prosorin[86]

prosorin[109]

prosorin[135]

prosorin[136]

prosorin[137]

prosorin[138]

prosorin[139]

prosorin[140]

prosorin[141]

prosorin[135]

prosorin[151]

prosorin[152]

prosorin[153]

prosorin[154]

prosorin[155]

prosorin[156]

prosorin[157]

prosorin[158]

```




```
X = df_final.loc[:, df_final.columns != 'y']
y = df_final.loc[:, df_final.columns == 'y']

#Over-sampling using SMOTE

#With our training data created, I'll up-sample the no-subscription using the SMOTE algorithm(Synthetic Minority
Oversampling Technique). At a high level, SMOTE:

#1. Works by creating synthetic samples from the minor class (no-subscription) instead of creating copies.

#2. Randomly choosing one of the k-nearest-neighbors and using it to create a similar, but randomly tweaked,
new observations.

from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

columns = X_train.columns

os_data_X,os_data_y=os.fit_sample(X_train, y_train)

os_data_X = pd.DataFrame(data=os_data_X,columns=columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=['y'])

# we can Check the numbers of our data

print("length of oversampled data is ",len(os_data_X))

print("Number of no subscription in oversampled data",len(os_data_y[os_data_y['y']==0]))

print("Number of subscription",len(os_data_y[os_data_y['y']==1]))

print("Proportion of no subscription data in oversampled data is
",len(os_data_y[os_data_y['y']==0])/len(os_data_X))

print("Proportion of subscription data in oversampled data is
",len(os_data_y[os_data_y['y']==1])/len(os_data_X))

#Now we have a perfect balanced data! You may have noticed that I over-sampled only on the training data,
because by oversampling only on the training data,

#none of the information in the test data is being used to create synthetic observations, therefore, no information
will bleed from test data into the model training.

#Recursive Feature Elimination

#Recursive Feature Elimination (RFE) is based on the idea to repeatedly construct a model and choose either the
best or worst performing feature, setting the feature aside

#and then repeating the process with the rest of the features. This process is applied until all features in the data
set are exhausted. The goal of RFE is to select features

#by recursively considering smaller and smaller sets of features.

df_final_vars=df_final.columns.values.tolist()

y=['y']
```



```
X=[i for i in df_final_vars if i not in y]
```

```
logreg = LogisticRegression()
```

```
rfe = RFE(logreg, 20)
```

```
rfe = rfe.fit(os_data_X, os_data_y.values.ravel())
```

```
print(rfe.support_)
```

```
print(rfe.ranking_)
```

```
cols=['reason_2','failures_0','goout_3','absences_5','G1_10','G1_11','G1_12','G1_13','G1_14','G1_15','G1_16','G1_17','G2_10','G2_11','G2_12','G2_13','G2_14','G2_15','G2_16','G2_17']
```

```
X=os_data_X[cols]
```

```
y=os_data_y['y']
```

```
#Implementing the model
```

```
import statsmodels.api as sm
```

```
logit_model=sm.Logit(y,X)
```

```
result = logit_model.fit(method='bfgs')
```

```
print(result.summary())
```

```
#Logistic Regression Model Fitting
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn import metrics
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```
logreg = LogisticRegression()
```

```
logreg.fit(X_train, y_train)
```

```
#Predicting the test set results and calculating the accuracy
```

```
y_pred = logreg.predict(X_test)
```

```
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
#Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix = confusion_matrix(y_test, y_pred)
```

```
print(confusion_matrix)
```

```
#The result is telling us that we have 6124+5170 correct predictions and 2505+1542 incorrect predictions.
```

```
#Compute precision, recall, F-measure and support
```

```
#To quote from Scikit Learn:
```

```
#The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.
```

```
#The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
```

#The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

#The F-beta score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, y_pred))
```

#Interpretation: Of the entire test set, 74% of the promoted term deposit were the term deposit that the customers liked. Of the entire test set, 74% of the customer's preferred term deposits that were promoted.

#ROC Curve

```
from sklearn.metrics import roc_auc_score
```

```
from sklearn.metrics import roc_curve
```

```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
```

```
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
```

```
plt.figure()
```

```
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
```

```
plt.plot([0, 1], [0, 1], 'r--')
```

```
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver operating characteristic')
```

```
plt.legend(loc="lower right")
```

```
plt.savefig('Log_ROC')
```

```
plt.show()
```

#The receiver operating characteristic (ROC) curve is another common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

Appendix H: Quantum Regression (Python)

```
import numpy as np
```

```
import pandas as pd
```

```
from pandas import DataFrame
```

```
import matplotlib.pyplot as plt
```

```
from sklearn import linear_model
```

```
import statsmodels.api as sm
```



```
School = pd.read_csv(r'C:\Users\schoo2\normNew.csv', sep=";")
```

```
df = DataFrame(School,
columns=['school','sex','age','add2ess','famsize','Ps121us','Medu','Fedu','Mjob','Fjob','reason','guardian','traveltim
e','studytime','failures','schoolsup','famsup','paid','activities','nursery','higher','internet','romantic','famrel','freeti
me','goout','Dalc','Walc','health','absences','G1','G2','G3'])
```

```
#print (df)
```

```
Parad = (-1, -0.1, 0.2, 0.4, 0.1, 0.2, 0.3, 0, 0.5, 0.2, 0.4, 0, 0.2, 0.2, 0.4, 0.7, 0.3, 0.3, -0.2, 0.2, 0.2, 0.1, 0.2, 0, 0, 0,
0.1, 0.1, 0.2, 0.2, 0, 0.5, 0.3, 0.3, 0.2, 0.1, 0.4, 0.3, 0.1, 0.4, 0.1, 0, 0.4, -0.1, 0, 0, 0.3, 0.7, 0.1, 0.3, 0.4, 0.6, 0, 0.3,
0.3, 0.2, 0.5, 0.5, 0.4, 0.6, 0.7, 0, 0.3, 0.4, 0.3, 0.6, 0.1, 0, 0.1, 0.5, 0.3, 0.1, 0.3, 0.3, 0.1, 0.1, 0.2, 0.3, -0.1, 0.2, 0.1,
0, 0.2, 0.3, 0.3, 0.2, 0.3, 0.5, 0.2, -0.1, -0.1, 0.4, 0.2, 0.3, 0.1, 0.3, -0.1, 0.3, 0.2, 0.2, -0.1, 0.6, 0.2, 0.1, 0.6, 0, 0, 0.3,
0.2, 0.5, 0.3, 0.1, -0.2, 0.8, 0, 0.6, 0.6, 0.4, 0.2, 0.4, 0.4, 0.4, 0.4, 0.2, 0.2, -0.1, 0, -0.1, 0.4, 0.3, 0, 0, 0.1, 0.1, 0.3,
0.5, -0.1, 0.3, 0.2, 0.3, 0, 0.4, 0.1, 0.2, 0.2, -0.1, 0, 0.3, -0.2, -0.1, -0.1, 0.5, 0.2, 0.3, 0, -0.1, 0.2, 0.1, 0.1, 0.3, 0.1, -
0.1, 0.2, 0.1, -0.1, 0.3, 0.1, 0.3, -0.3, -0.1, 0.1, 0.2, 0, -0.1, -0.2, -0.3, -0.2, -0.1, -0.2, -0.2, -0.1, 0.7, 0.2, 0.6, -0.1,
0.6, 0.1, 0.4, 0.4, 0, 0.3, 0.1, 0, 0.1, 0.1, 0.1, 0.7, 0.4, 0.4, 0.1, 0, 0.3, 0.2, -0.1, 0.2, 0.2, 0, 0.1, 0.3, 0.4, 0.3, 0, 0.4,
0.1, 0.4, 0.1, 0.4, 0.3, 0.3, -0.2, 0.1, 0.4, 0.2, 0.1, 0.2, 0.3, 0.3, 0.2, 0.2, 0.4, 0.1, 0, 0.2, 0.2, 0.1, 0.2, 0.3, 0, 0.5, -0.2,
0.7, 0, -0.1, 0.5, 0.4, 0.2, 0.3, 0.3, -0.1, 0.2, 0.6, -0.1, 0.4, 0, -0.1, -0.2, -0.3, 0, 0.4, 0.1, 0.2, 0.3, -0.1, -0.2, 0.1, 0.5,
0.5, 0.5, 0.2, 0.4, 0.4, 0.1, 0.3, 0.1, 0, 0.2, 0.5, 0.1, -0.1, -0.3, 0, 0.1, -0.3, -0.1, -0.3, 0.4, 0.2, -0.1, 0.2, 0.4, 0, 0.1,
0.1, 0.1, 0.1, 0.1, 0.6, -0.1, 0, 0, 0.5, 0, 0, 0.5, 0.1, -0.2, 0.5, 0.1, 0, 0, -0.2, 0.1, 0.6, 0.3, 0.6, 0.4, 0.5, 0.2, 0.2, 0.1,
0.4, 0.3, -0.1, 0, 0, 0.1, 0.3, 0.7, 0.2, 0.2, 0.2, 0.2, 0.8, 0.3, 0.4, 0.7, 0.5, 0.7, 0.8, 0.4, 0.4, 0.4, 0.3, 0.6, 0.8, 0.4, 0.2, -
0.3, 0.6, 0.8, -0.1, 0.4, 0, 0, 0.2, 0.1, 0.5, 0.4, 0.2, 0.5, 0.1, 0.1, 0.2, 0.1, 0.4, 0.5, 0.2, 0.1, -0.2, 0, -0.2, 0, 0.4, 0.1,
0.7, 0.4, 0.4, 0.3, 0.7, 0.1, 0.3, 0.5, 0.1, 0.1, 0.1, 0, 0.5, 0.5, 0, 0, 0.4, 0.1, 0.2, 0.4, 0.4, 0.2, 0.6, 0, 0.1, 0.4, 0.5, 0.4,
0.1, 0.2, 0.2, 0, 0, -0.1, -0.1, 0.3, 0.3, 0.2, 0.4, -0.1, 0.1, -0.1, 0.6, 0.4, 0.3, -0.3, 0.4, -0.2, 0, 0, 0.2, 0, 0, 0.6, -0.4, 0,
0, 0, -0.4, 0.3, 0, 0, -0.4, 0.2, 0, 0.4, -0.3, -0.2, 0.2, -0.3, -0.1, -0.3, 0.1, -0.2, 0.6, 0.1, 0.5, 0, -0.1, 0, -0.1, -0.1, 0.3, 0,
0.3, 0, 0.1, 0.3, 0.3, 0.4, 0.1, -0.1, 0, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0, 0, 0.2, 0, 0.2, -0.3, -0.1, -0.1, -0.1, -0.1, -0.1, -0.3,
0.1, -0.1, -0.3, -0.3, -0.2, -0.1, -0.3, -0.2, -0.1, -0.2, 0.4, 0.4, 0, 0.4, 0.7, -0.4, 0.4, 0, 0.4, 0.2, 0.1, 0.1, 0.1, 0, 0.6, 0.5,
0.1, -0.3, -0.2, -0.3, -0.1, 0.4, 0.4, -0.2, -0.2, -0.4, -0.2, -0.2, -0.5, -0.3, 0.5, 0.3, 0, -0.1, 0.1, -0.3, 0, -0.1, 0.4, 0.1, -
0.1, 0.3, 0.2, 0.1, -0.1, -0.1, 0.1, 0, -0.1, 0.2, -0.2, 0.3, 0, 0.1, 0.7, 0.2, 0.3, -0.1, 0.1, 0, 0.3, -0.1, -0.3, -0.2, 0.2, 0, 0,
0.1, -0.3, 0.1, 0, -0.1, -0.6, -0.4, -0.6, -0.3, -0.2, -0.5, -0.3, -0.2, 0, -0.1, 0, -0.2, -0.1, -0.2, -0.4, -0.4, -0.2, 0, -0.2, -0.2,
-0.3, -0.2, -0.3, -0.3, 0.2, 0.1, 0.2, 0.8, 0.7, 0.7, -0.1, 0, 0.2, 0.2, -0.2, 0.1, -0.5, -0.1, -0.5, 0.8, 0.1, -0.2, 0.1, -0.2, 0.4,
0, 0, 0.2, 0.5, 0.4, 0.9, 0.6, 0.3, 0.5, 0.3, -0.2, 0.5, -0.2, 0, -0.3, -0.1, 0, -0.3, 0.5, 0, -0.3, 0.3, 0.6, -0.2, 0.7, -0.3, 0.4, -
0.5, -0.3, 0.4, -0.4, -0.3, 0, 0.5, 0.1, 0, 0)
```

```
Parad = np.array(Parad)
```

```
Parad2 = (0.1, 0.1, 0.2, 0.4, 0.3, 0.3, 0.3, 0.3, 0.7, 0.3, 0.4, 0.3, 0.2, 0.3, 0.5, 0.7, 0.4, 0.4, -0.3, 0.2, 0.4, 0.2, 0.4, 0,
0, 0.2, 0.2, 0.1, 0.3, 0.2, 0.1, 0.5, 0.5, 0.2, 0.2, 0.1, 0.4, 0.3, 0.2, 0.2, 0, 0.1, 0.5, 0, 0.1, 0.1, 0.3, 0.7, 0.3, 0.2, 0.3,
0.6, -0.1, 0.2, 0.3, 0.2, 0.5, 0.6, 0.4, 0.6, 0.6, 0.6, 0, 0.3, 0.2, 0.6, 0.2, 0, 0.1, 0.5, 0.1, 0, 0.1, 0.4, 0.1, 0.1, 0.1, 0.3, 0,
0.1, 0.2, -0.1, 0.1, 0.3, 0.2, 0.2, 0.1, 0.5, 0.1, 0, 0.1, 0.3, 0.2, 0.4, 0.2, 0.3, 0.1, 0.2, 0.3, 0.3, -0.2, 0.6, 0.2, 0, 0.6, 0, 0,
0.4, 0.1, 0.4, 0.4, 0.1, 0, 0.8, 0, 0.4, 0.6, 0.5, 0.1, 0.4, 0.4, 0.3, 0.3, 0.3, 0.1, -0.1, 0.1, 0.1, 0.5, 0.3, 0.2, -0.2, 0.1, 0.3,
0.2, 0.4, 0.1, 0.1, 0.1, 0.5, 0, 0.3, 0.2, 0.1, 0.1, 0, 0, 0.4, -0.1, 0.1, -0.1, 0.3, 0.1, 0.3, 0.1, -0.4, 0.2, 0, 0.1, 0.3, 0.1, -
0.2, 0.1, -1, 0, 0.3, 0.1, 0.3, -0.2, 0, 0.1, 0.1, -0.9, 0, -0.1, -0.2, 0, -0.2, -0.2, -0.2, 0.1, 0.8, 0.3, 0.7, 0, 0.8, 0, 0.3, 0.5,
0.1, 0.4, 0, 0.1, 0.3, 0.1, 0.3, 0.7, 0.4, 0.6, 0.4, 0.1, 0.6, 0.4, 0, 0.3, 0.2, 0.2, 0, 0.2, 0.6, 0.4, 0.2, 0.6, 0.1, 0.5, 0.2,
0.5, 0.3, 0.3, -0.2, 0.2, 0.5, 0.3, 0.2, 0.2, 0.2, 0.3, 0.1, 0.1, 0.5, 0, 0, 0.3, 0.3, 0.1, 0.2, 0.4, 0, 0.6, -0.2, 0.7, 0.1, 0.1,
0.6, 0.2, 0.3, 0.3, 0.4, -0.1, 0.2, 0.6, 0, 0.3, 0, 0, -0.3, -0.2, -0.1, 0.5, 0, 0.1, 0.3, -0.2, -0.2, 0, 0.5, 0.4, 0.5, 0.2, 0.5,
0.5, 0.2, 0.5, 0.1, 0, 0.1, 0.6, 0.1, 0.3, -0.5, 0, 0.1, -0.3, 0, -0.4, 0.2, 0.3, 0, 0.3, 0.7, 0.1, 0.1, 0.4, 0.4, 0.3, 0.4, 0.6, 0,
0.2, 0.2, 0.5, 0.1, 0.2, 0.3, 0.3, -0.1, 0.6, 0.4, 0.2, 0.4, 0, 0.2, 0.6, 0.3, 0.8, 0.5, 0.6, 0.2, 0, 0.2, 0.3, 0.5, 0, 0, 0.1, 0,
0.3, 0.8, 0.3, 0.4, 0.4, 0.2, 0.8, 0.4, 0.5, 0.7, 0.6, 0.8, 0.9, 0.5, 0.5, 0.3, 0.4, 0.7, 0.7, 0.5, 0.3, -0.2, 0.6, 0.8, 0.1, 0.5,
0.1, 0.1, 0.5, 0.4, 0.7, 0.7, 0.5, 0.7, 0.4, 0, 0.3, 0.4, 0.7, 0.7, 0.3, 0.4, 0.1, 0.1, -0.1, 0, 0.3, 0, 0.7, 0.5, 0.4, 0.3, 0.7, 0,
0.3, 0.5, 0.1, 0.2, 0, 0, 0.5, 0.5, 0.2, 0.2, 0.4, 0.4, 0.5, 0.5, 0.6, 0.3, 0.7, 0.4, 0.4, 0.7, 0.7, 0.4, 0.3, 0.5, 0.6, 0.1, 0.3,
0.2, 0.2, 0.5, 0.7, 0.5, 0.7, 0, 0.5, 0.1, 0.8, 0.7, 0.4, 0.1, 0.7, 0, 0.3, 0.1, 0.2, 0, 0.1, 0.7, -0.1, 0.1, 0.1, 0, -0.3, 0.4, 0.1,
0, -0.2, 0.2, 0.2, 0.6, -1, -0.1, 0.4, -0.2, 0.1, -0.1, 0.1, -0.1, 0.7, 0.3, 0.5, 0.1, 0.1, -0.2, -0.2, -0.1, 0.5, 0.1, 0.3, 0, 0.1,
0.4, 0.4, 0.2, 0.1, -0.2, 0.1, 0.4, 0.3, 0.3, 0.2, 0.2, 0.6, 0, 0.1, 0.4, -0.2, 0.1, -0.2, 0, 0, 0.1, -0.1, 0.1, -0.2, 0.1, 0, 0, -
0.1, 0, 0, -0.1, 0, 0, -0.1, 0.3, 0.4, 0, 0.4, 0.6, -0.3, 0.3, -0.1, 0.4, 0.3, 0.1, 0, 0, -0.1, 0.8, 0.7, 0, -0.3, -0.2, -0.3, 0, 0.6,
0.5, -0.2, -1, -0.2, 0, -0.2, -0.4, -0.2, 0.6, 0.4, 0, -0.1, 0.1, -0.1, 0, -0.2, 0.6, 0.2, 0.1, 0.4, 0.2, 0.1, 0.1, 0.2, -0.2,
0.2, -0.2, 0.6, 0.1, 0.1, 0.8, 0.3, 0.3, 0, 0.2, 0, 0.3, 0.1, 0, 0, 0.3, 0, 0, 0.2, -1, 0, -0.1, -0.1, -1, -0.1, -0.2, -0.2, -0.1, -
```

0.3, 0, 0, 0, 0.1, 0.1, 0, -0.1, 0, -0.2, -0.3, -1, 0.1, -0.2, -1, -0.2, -0.1, 0, -0.3, 0.4, 0.3, 0.4, 0.8, 0.7, 0.8, -1, 0.1, 0.4, 0.4, 0, 0.3, -1, 0, -1, 0.8, 0.2, 0.1, 0.2, -1, 0.5, 0.1, 0, 0.2, 0.5, 0.4, 0.8, 0.5, 0.3, 0.5, 0.3, -0.1, 0.6, -0.1, 0, -1, 0, 0.2, -0.1, 0.7, 0.2, -0.1, 0.4, 0.6, -0.1, 0.9, -1, 0.6, -1, -1, 0.5, 0.1, 0, 0, 0.6, -0.1, 0, 0.1)

```
Parad2 = np.array(Parad2)
```

#X = df['G1'] # here we have 2 variables for multiple regression. If you just want to use one variable for simple linear regression, then use X = df['Interest_Rate'] for example. Alternatively, you may add additional variables within the brackets

```
#y = df['G3']
```

```
m=649
```

```
X = Parad
```

```
y = Parad2
```

```
#Xnum = pd.to_numeric(X, errors='raise', downcast='float')
```

```
#X = X.values.reshape(-1,1)
```

```
#y = y.values.reshape(-1,1)
```

```
X = np.array(X)
```

```
y = np.array(y)
```

```
#np.random.seed(0)
```

```
#m = 8
```

```
#X = np.linspace(-0.95,0.95,m)
```

```
#y = X**2
```

```
import qsimulator as pq
```

```
from qsimulator import RX, RY, RZ
```

```
n_qubits = 3
```

```
def input_prog(sample):
```

```
    p = pq.Program(n_qubits)
```

```
    for j in range(n_qubits):
```

```
        p.inst(RY(np.arcsin(sample[0])), j)
```

```
        p.inst(RZ(np.arccos(sample[0]**2)), j)
```

```
    return p
```

```
from qcl import ising_prog_gen
```

```
ising_prog = ising_prog_gen(trotter_steps=1000, T=10, n_qubits=n_qubits)
```

```
depth = 3
```

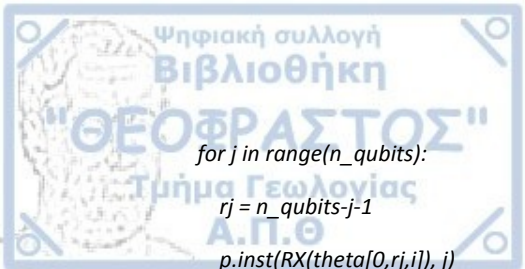
```
def output_prog(theta):
```

```
    p = pq.Program(n_qubits)
```

```
    theta = theta.reshape(3,n_qubits,depth)
```

```
    for i in range(depth):
```

```
        p += ising_prog
```



```

for j in range(n_qubits):
    rj = n_qubits-j-1
    p.inst(RX(theta[0,rj,i]), j)

    p.inst(RZ(theta[1,rj,i]), j)

    p.inst(RX(theta[2,rj,i]), j)

return p

def grad_prog(theta, idx, sign):

    theta = theta.reshape(3,n_qubits,depth)

    idx = np.unravel_index(idx, theta.shape)

    p = pq.Program(n_qubits)

    for i in range(depth):

        p += ising_prog

        for j in range(n_qubits):

            rj = n_qubits-j-1

            if idx == (0,rj,i):

                p.inst(RX(sign*np.pi/2.0), j)

            p.inst(RX(theta[0,rj,i]), j)

            if idx == (1,rj,i):

                p.inst(RZ(sign*np.pi/2.0), j)

            p.inst(RZ(theta[1,rj,i]), j)

            if idx == (2,rj,i):

                p.inst(RX(sign*np.pi/2.0), j)

            p.inst(RX(theta[2,rj,i]), j)

        return p

from qsimulator import Z

from qcl import QCL

state_generators = dict()

state_generators['input'] = input_prog

state_generators['output'] = output_prog

state_generators['grad'] = grad_prog

initial_theta = np.random.uniform(0.0, 2*np.pi, size=3*n_qubits*depth)

operator = pq.Program(n_qubits)

operator.inst(Z, 0)

operator_programs = [operator]

est = QCL(state_generators, initial_theta, loss="mean_squared_error",

```

```
operator_programs=operator_programs, epochs=20, batch_size=m,
verbose=True)
est.fit(X,y)
```

```
results = est.get_results()

print(results)

#X_test = np.array()

#X_test = np.linspace(-1.0,1.0,50)

#y_pred = est.predict(X_test)

X_test = X

y_pred = est.predict(X_test)

print(y_pred)

import matplotlib.pyplot as plt

plt.plot(X, y, 'bs', X_test, y_pred, 'r-')
```

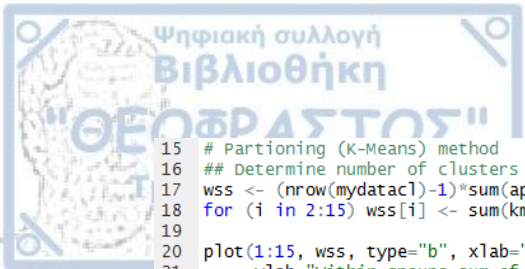
Appendix I: Classical methods of Clustering Analysis (k-Means, Hierarchical and Model-based method)

- Package installation and Data preparation:**

```
1 install.packages("devtools")
2 devtools::install_github("rdtaylor/quantumClustering")
3 library(quantumClustering)
4
5 #Firstly Import Dataset (see right on the screen) as Excel file
6 mydata <- Absenteeism_at_work2
7 str(mydata)
8 mydata
9 is.data.frame(mydata)
10
11 # Prepare Data
12 mydatacl <- na.omit(mydata) # listwise deletion of missing
13 mydatacl <- scale(mydatacl) # standardize variables

> str(mydata)
Classes 'tbl_df', 'tbl' and 'data.frame':    740 obs. of  20 variables:
 $ Reason for absence      : num  26 0 23 7 23 23 22 23 19 22 ...
 $ Month of absence       : num  7 7 7 7 7 7 7 7 7 7 ...
 $ Day of the week        : num  3 3 4 5 5 6 6 6 2 2 ...
 $ Seasons                : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Transportation expense : num  289 118 179 279 289 179 361 260 155 235 ...
 $ Distance from Residence to work: num  36 13 51 5 36 51 52 50 12 11 ...
 $ Service time           : num  13 18 18 14 13 18 3 11 14 14 ...
 $ Age                   : num  33 50 38 39 33 38 28 36 34 37 ...
 $ work load Average/day  : num  239554 239554 239554 239554 239554 ...
 $ Hit target            : num  97 97 97 97 97 97 97 97 97 97 ...
 $ Disciplinary failure   : num  0 1 0 0 0 0 0 0 0 0 ...
 $ Education              : num  1 1 1 1 1 1 1 1 1 3 ...
 $ Son                   : num  2 1 0 2 2 0 1 4 2 1 ...
 $ Social drinker         : num  1 1 1 1 1 1 1 1 1 0 ...
 $ Social smoker          : num  0 0 0 1 0 0 0 0 0 0 ...
 $ Pet                   : num  1 0 0 0 1 0 4 0 0 1 ...
 $ weight                : num  90 98 89 68 90 89 80 65 95 88 ...
 $ Height                : num  172 178 170 168 172 170 172 168 196 172 ...
 $ Body mass index        : num  30 31 31 24 30 31 27 23 25 29 ...
 $ Absenteeism time in hours : num  4 0 2 4 2 2 8 4 40 8 ...
```

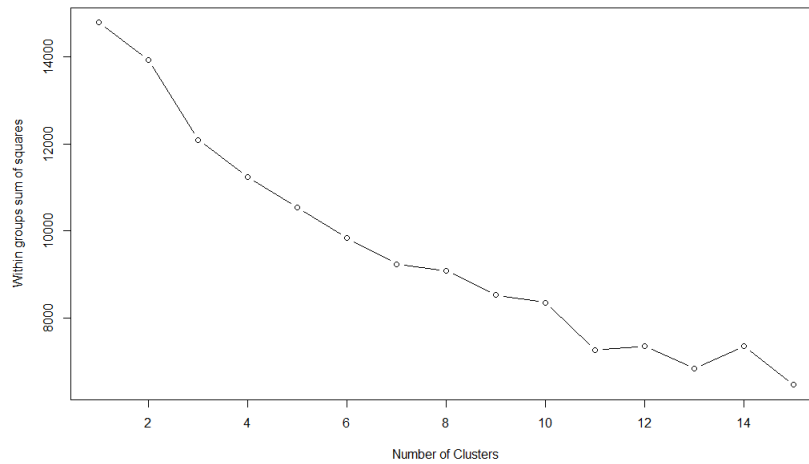
- k-Means Clustering method:**



```

15 # Partitioning (K-Means) method
16 ## Determine number of clusters (determine the suitable number of clusters)
17 wss <- (nrow(mydatacl)-1)*sum(apply(mydatacl,2,var))
18 for (i in 2:15) wss[i] <- sum(kmeans(mydatacl,
19                                centers=i)$withinss)
20 plot(1:15, wss, type="b", xlab="Number of Clusters",
21      ylab="Within groups sum of squares")

```



```

23 ## K-Means Cluster Analysis
24 fit <- kmeans(mydatacl, 14) # 14 cluster solution
25 ## get cluster means
26 aggregate(mydatacl,by=list(fit$cluster),FUN=mean)
27 ## append cluster assignment (column in the first dataset)
28 mydatacl <- data.frame(mydatacl, fit$cluster)
29 mydatacl
30 fit$cluster

```

Figure 27: Cluster means

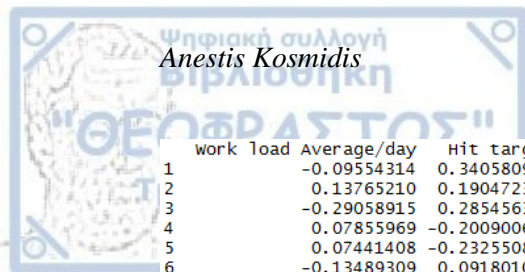
```

> aggregate(mydatacl,by=list(fit$cluster),FUN=mean)

```

Group.1	Reason for absence	Month of absence	Day of the week	Seasons
1	1	-0.75191640	-0.14894691	-0.15992749 -0.15253630
2	2	-0.27799223	-0.20630981	0.07791948 -0.23613598
3	3	-0.10468877	0.13196030	0.03383200 -0.02345327
4	4	0.10987743	1.31078900	0.01968965 1.25762141
5	5	0.09647757	0.13147769	0.03888681 -0.10051777
6	6	0.05169414	-0.26519347	0.25866946 0.03810105
7	7	-0.68571144	0.37123666	0.03643717 0.39460918
8	8	0.19669204	-0.21078692	-0.36215378 -0.10756544
9	9	-0.62763958	0.26378615	-0.58940472 0.06366976
10	10	-0.02563807	0.18753547	-0.20388972 -0.12442951
11	11	0.05905912	0.28497245	0.48694542 0.18474516
12	12	-0.81332186	0.07191025	-0.14108653 -0.04010915
13	13	0.13105172	-0.84269822	0.02220178 -0.36132958
14	14	0.46297706	-0.22985321	-0.01894514 -0.16416669

	Transportation expense	Distance from Residence to work	Service time	Age
1	1.17408753	-0.95664780	-1.4733160	-0.72062262
2	-0.32279408	-0.62666849	0.4467099	0.31443912
3	0.18205204	-1.21579468	0.3635439	0.17635943
4	0.02409371	-0.23318069	-0.7583988	-1.26015935
5	0.72203595	0.01279350	0.2565753	0.58019559
6	-0.63223785	-0.24473498	-0.8105260	-0.99555896
7	-0.62701024	-0.68957518	0.8466864	1.50491471
8	-1.36372962	-1.27679795	-0.2631898	-0.04051693
9	-0.34615555	-0.88244296	0.2420436	0.63105497
10	1.55448938	1.41288453	-2.2073737	-1.14508575
11	0.98458673	0.81681551	-0.2973983	-0.26239539
12	1.42466600	0.72775118	-0.2566739	0.72434357
13	-0.07933826	-0.08827062	-0.7698016	-1.24086558
14	-0.58253694	1.41702441	1.2970329	0.28980586



	work load	Average/day	Hit target	Disciplinary failure	Education	Son
1		-0.09554314	0.340580978	0.31353545	1.3767168	-0.927564046
2		0.13765210	0.190472301	0.10106637	0.3281583	0.566329486
3		-0.29058915	0.285456322	-0.07520427	2.7572064	0.117642714
4		0.07855969	-0.200900635	0.01365052	-0.4335642	0.034796832
5		0.07441408	-0.232550885	-0.20590388	-0.4335642	0.213759464
6		-0.13489309	0.091801012	-0.23888415	2.5371533	-0.927564046
7		-0.12821070	-0.512748684	1.16057883	-0.4335642	-0.002050318
8		0.27125514	0.079290112	-0.23888415	-0.4335642	-0.324462887
9		-0.28757946	0.516131803	-0.23888415	-0.4335642	0.472961141
10		-0.24375898	0.001564348	-0.23888415	-0.1086419	-0.017222675
11		-0.14703372	-0.117741155	0.15570128	-0.4335642	2.486216095
12		0.10387215	0.090157544	1.33945755	-0.4335642	0.502972395
13		0.43209139	0.283881332	-0.23888415	-0.4335642	-0.033478770
14		-0.17989247	0.129586608	-0.23888415	-0.4335642	-0.919716275

	Social drinker	Social smoker	Pet	weight	Height	Body mass index
1	-1.14486958	-0.28037622	-0.56585720	0.07246756	1.3065687	-0.41320657
2	-0.16215491	3.56181647	-0.48805446	-0.85854191	0.1721594	-0.91187489
3	-0.84603237	-0.28037622	0.30510121	0.76485234	0.2264487	0.50749016
4	-1.14486958	-0.28037622	0.95129616	-0.77893124	-0.5350710	-0.61134369
5	0.42068058	-0.28037622	0.01156684	-0.07107950	-0.3405410	0.06839937
6	-1.14486958	-0.28037622	-0.56585720	-1.78799649	-0.1847334	-1.79141585
7	0.77142403	-0.28037622	-0.40149892	1.53674414	0.1991388	1.48711796
8	-0.46408106	-0.28037622	-0.56585720	0.62502781	1.2858562	0.08994919
9	0.09645421	0.01517706	-0.21574489	0.33163761	1.0134068	-0.10413319
10	0.87228158	-0.28037622	3.22702621	-0.08519888	0.1880922	-0.17985897
11	0.87228158	-0.28037622	-0.56585720	-0.99238735	-0.6404090	-0.74135164
12	-1.14486958	3.56181647	2.79355382	-0.32429745	-0.4687909	-0.12464726
13	-1.14486958	-0.28037622	0.76165199	-0.88565930	-0.5930661	-0.68718166
14	0.82011388	-0.28037622	-0.52662048	0.82299330	-0.3461483	1.05301096

Absenteeism time in hours

1	-0.069336468
2	-0.092417434
3	-0.124901755
4	-0.317951440
5	0.019111859
6	-0.106843037
7	-0.260619970
8	-0.091840409
9	6.174064709
10	-0.003699972
11	0.028448515
12	-0.171140012
13	-0.197930419
14	-0.194789475

Figure 28: Append cluster assignment

	Absenteeism.time.in.hours	fit.cluster
1	-0.21936274	5
2	-0.51941530	7
3	-0.36938902	14
4	-0.21936274	2
5	-0.36938902	5
6	-0.36938902	14
7	0.08068981	10
8	-0.21936274	11
9	2.48111022	8
10	0.08068981	3
11	0.08068981	11
12	0.08068981	11
13	0.08068981	11
14	-0.44440216	14
15	-0.21936274	14
16	0.08068981	5
17	-0.36938902	14
18	0.08068981	14
19	0.08068981	13
20	-0.36938902	5
21	0.08068981	1
22	-0.44440216	14
23	2.48111022	10
24	-0.21936274	11
25	0.08068981	5
26	0.00567667	10
27	-0.44440216	5
28	-0.21936274	2
29	0.08068981	5
30	-0.36938902	14
31	0.08068981	14
32	0.08068981	12
33	-0.21936274	3
34	0.08068981	12
35	-0.36938902	14
36	-0.44440216	10
		37
	0.08068981	5
	-0.21936274	5
	0.08068981	12
	-0.21936274	11
	-0.36938902	13
	-0.21936274	8
	-0.21936274	14
	0.08068981	7
	-0.36938902	8
	-0.29437588	8
	-0.29437588	14
[reached getoption("max.print") -- omitted 693 rows]		

Figure 29: The cluster of each observation

```

> fit$cluster
[1] 5 7 14 2 5 14 10 11 8 3 11 11 11 14 14 5 14 14 13 5 1 14 10 11 5 10 5 2
[29] 5 14 14 12 3 12 14 10 5 5 12 11 13 8 14 7 8 8 14 5 11 5 11 4 4 8 7 7
[57] 4 14 11 5 14 11 14 8 7 6 14 4 8 4 5 5 14 4 11 5 4 5 10 11 14 4 14 2
[85] 5 8 4 5 4 8 4 2 4 11 5 4 5 5 5 9 7 3 4 11 9 10 4 11 4 10 8 5
[113] 4 13 8 8 8 13 13 13 13 8 13 8 8 8 8 8 6 5 8 13 8 8 13 5 13 5 14
[141] 8 13 11 5 1 1 13 13 6 13 13 1 1 13 13 5 11 2 2 5 11 6 5 11 2 8 11 8
[169] 5 2 11 13 2 14 13 13 6 13 13 14 2 13 5 13 5 13 8 5 8 13 11 14 7 5 5 5
[197] 5 11 5 8 12 5 3 12 5 8 8 14 13 13 8 14 5 2 1 12 1 14 7 10 5 10 5 5
[225] 5 14 8 5 7 3 7 9 12 14 14 13 8 7 8 8 1 6 8 1 1 2 10 5 14 5 5 11
[253] 5 2 5 5 2 2 11 5 5 1 6 7 5 3 7 3 10 13 14 14 5 7 5 5 7 5 14 5
[281] 14 5 11 7 7 7 5 5 14 5 3 5 5 7 7 6 8 11 14 6 7 5 11 7 7 7 7 5
[309] 6 7 10 11 7 2 6 6 7 8 5 14 11 5 12 9 7 1 3 8 3 14 5 5 5 14 11 11
[337] 12 2 14 4 14 14 3 7 11 5 14 14 6 8 3 14 7 3 11 2 14 5 13 8 8 3 8 5
[365] 5 14 8 14 14 8 14 13 5 11 14 14 14 14 5 14 13 14 5 5 2 14 14 5 14 14 8 5
[393] 5 11 14 3 2 12 14 10 11 3 7 7 7 3 7 14 5 11 5 14 11 1 11 5 1 14 7
[421] 9 2 11 14 8 7 5 11 6 14 10 11 2 2 5 13 1 6 8 3 6 8 14 8 13 11 7 5
[449] 14 5 14 7 14 13 7 14 6 5 1 3 6 13 11 1 1 8 11 8 2 12 5 9 14 7 10 5
[477] 7 5 5 2 14 2 5 8 5 7 5 14 2 1 3 5 8 5 11 5 5 8 8 5 6 5 14 13
[505] 6 11 10 7 1 5 7 5 8 14 5 4 11 11 4 11 14 5 5 11 4 11 4 4 1 5 4 4
[533] 14 8 1 14 8 5 1 4 10 5 8 4 14 8 8 4 7 7 5 7 4 11 10 14 5 4 6 4
[561] 4 10 2 7 10 6 4 4 4 9 6 6 2 2 6 2 14 2 6 8 6 14 5 14 14 13 5
[589] 14 13 14 6 3 14 10 14 13 14 14 13 6 2 14 10 6 14 14 14 8 1 14 13 14 5 13 14
[617] 14 1 14 5 5 5 9 14 13 14 13 6 1 10 14 14 14 6 14 5 14 13 14 5 5 14 5 14
[645] 6 14 14 2 8 14 14 6 9 11 13 8 10 5 13 6 11 2 7 10 8 3 6 13 1 6 8 13
[673] 7 6 3 3 1 8 7 6 5 3 9 3 6 7 7 8 7 6 12 2 9 1 5 2 13 1 1 5
[701] 6 10 8 2 8 13 2 6 8 8 12 7 10 6 12 1 7 6 5 6 8 10 8 8 10 7 10 5
[729] 8 9 13 8 10 13 9 5 3 10 7 5

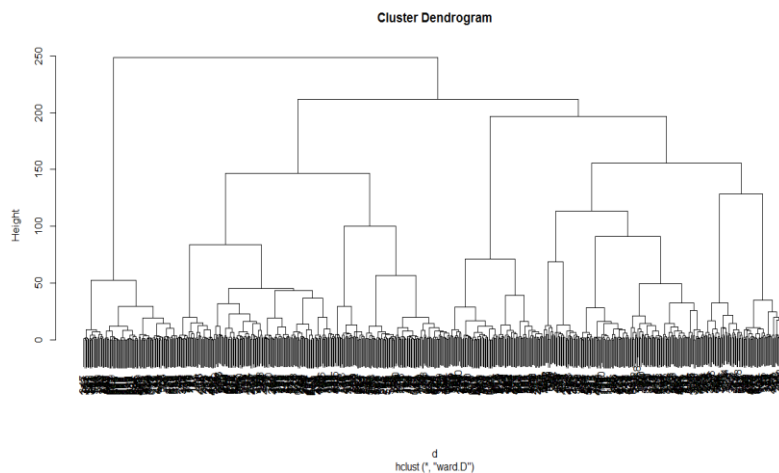
```

- **Hierarchical method:**

```

37 # Prepare Data για Hierarchical Method
38 mydatacl2 <- na.omit(mydata) # listwise deletion of missing
39 mydatacl2 <- scale(mydatacl2) # standardize variables
40 #Hierarchical Method (ward)
41 ## ward Hierarchical Clustering
42 d <- dist(mydatacl2, method = "euclidean") ## distance matrix
43 fit2 <- hclust(d, method="ward.D")
44 plot(fit2) ## display dendrogram
45 groups <- cutree(fit2, k=14) ## cut tree into 14 clusters
46 ## draw dendrogram with red borders around the 14 clusters
47 rect.hclust(fit2, k=14, border="red")

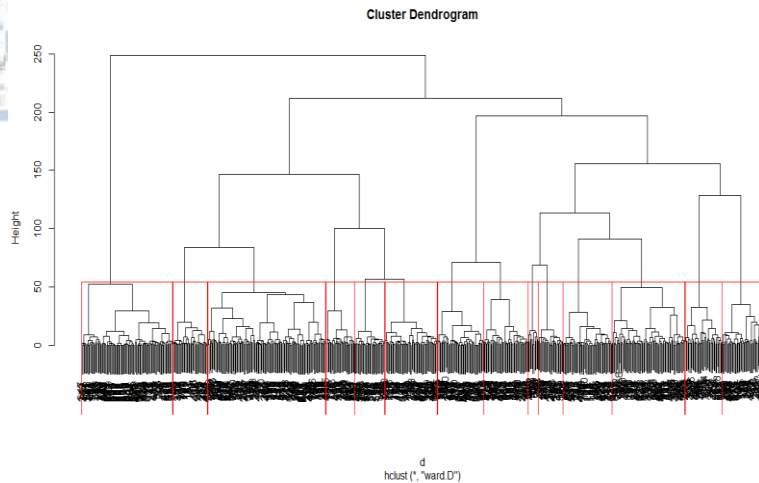
```



```

> groups
[1] 1 2 3 4 1 3 5 6 7 8 6 6 6 3 3 1 3 3 9 10 8 3 5 6 1 5 1 4
[29] 1 3 3 4 8 4 3 5 1 9 4 6 9 11 3 10 7 11 3 1 6 1 2 2 12 11 2 2
[57] 12 3 2 10 3 6 3 11 2 13 3 12 11 12 10 1 3 12 6 1 12 1 5 6 3 12 3 4
[85] 1 7 12 1 12 7 12 4 12 6 10 12 1 1 10 14 10 8 12 6 14 5 12 6 12 5 11 1
[113] 12 9 11 11 7 9 9 9 11 9 11 11 11 11 11 13 1 11 9 11 11 9 1 9 1 3
[141] 7 9 6 1 10 10 9 9 13 9 9 8 9 9 1 6 4 4 1 6 13 10 6 4 7 6 7
[169] 1 4 6 9 4 1 9 9 13 9 9 1 4 9 10 9 1 9 7 1 7 9 6 1 10 1 1 1
[197] 10 6 1 7 4 1 8 2 1 7 7 1 9 9 7 1 1 2 2 2 2 1 10 5 1 5 1 1
[225] 1 1 7 1 10 8 10 14 4 1 10 9 7 10 11 11 8 13 7 8 8 4 5 1 3 1 1 2
[253] 1 4 1 10 4 4 6 1 10 8 13 10 10 8 10 8 5 9 1 1 1 2 1 1 2 2 1 1
[281] 1 1 1 10 10 2 1 1 1 1 8 1 1 2 2 13 11 1 3 13 2 1 6 2 10 10 10 1
[309] 13 10 5 2 2 2 13 13 10 11 10 3 6 1 4 14 10 2 8 11 8 3 1 1 1 3 6 6
[337] 2 2 3 12 3 3 8 10 6 1 3 3 13 11 8 3 10 8 6 4 3 1 9 11 7 8 7 1
[365] 1 10 10 3 3 11 3 9 1 6 3 3 3 3 1 3 9 3 1 1 4 3 3 1 3 3 11 1
[393] 1 6 3 8 4 4 3 5 2 8 10 10 10 2 2 2 3 1 6 1 3 6 8 1 10 8 3 10
[421] 14 4 6 3 11 10 10 1 13 3 5 6 4 4 10 9 8 13 11 8 13 11 3 11 9 6 2 1
[449] 3 1 3 10 3 9 10 3 13 1 8 8 13 9 6 8 8 11 6 11 4 4 1 14 3 10 5 1
[477] 10 1 1 4 3 4 1 11 1 10 1 3 4 8 8 1 11 1 6 1 1 11 7 1 13 10 3 9
[505] 13 1 5 10 13 1 10 1 11 10 1 12 1 1 12 1 3 10 1 1 12 1 12 12 13 10 2 12
[533] 3 11 13 3 11 1 13 12 5 1 11 12 3 11 11 12 2 2 10 2 12 1 5 3 1 12 13 12
[561] 12 5 4 10 5 13 12 12 12 14 13 13 4 4 13 4 10 4 13 7 13 3 1 3 3 3 9 10
[589] 3 9 3 13 8 3 5 3 9 3 3 9 13 4 3 5 13 3 3 3 7 8 3 9 3 10 9 3
[617] 3 8 3 10 10 10 14 3 9 3 9 13 8 5 3 3 3 13 3 10 3 9 3 1 10 3 10 3
[645] 13 3 3 4 7 3 3 13 14 1 9 11 5 10 9 13 1 4 10 5 11 8 13 9 8 13 7 9
[673] 10 13 8 8 8 11 10 13 1 8 4 8 13 10 10 11 2 13 2 4 7 8 1 4 9 8 8 1
[701] 13 5 7 4 7 9 4 13 11 11 4 10 5 13 2 2 10 13 1 13 11 5 11 11 5 10 5 10
[729] 11 14 9 11 5 9 14 1 8 10 10 10

```



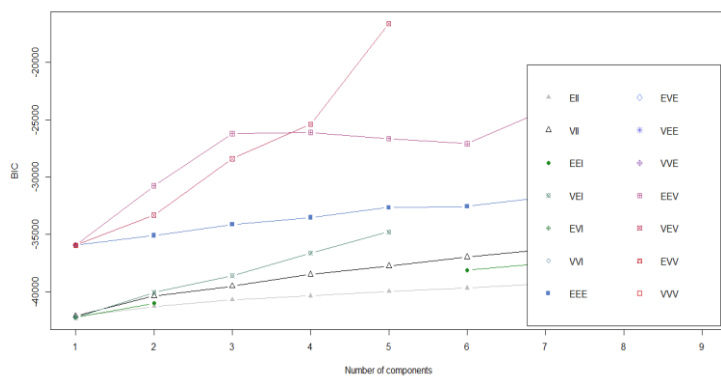
- **Model-based Method:**

```
50 # Prepare Data for Model based method
51 mydatacl3 <- na.omit(mydata) # listwise deletion of missing
52 mydatacl3 <- scale(mydatacl3) # standardize variables
53 # Model Based Clustering
54 library(mclust)
55 fit3 <- Mclust(mydatacl3)
56 plot(fit3) ## plot results
57 summary(fit3) ## display the best model
```

```
> fit3 <- Mclust(mydatacl3)
fitting ...
=====| 100%
> plot(fit3) ## plot results
Model-based clustering plots:

1: BIC
2: classification
3: uncertainty
4: density

selection: |
```



- **Classification:**

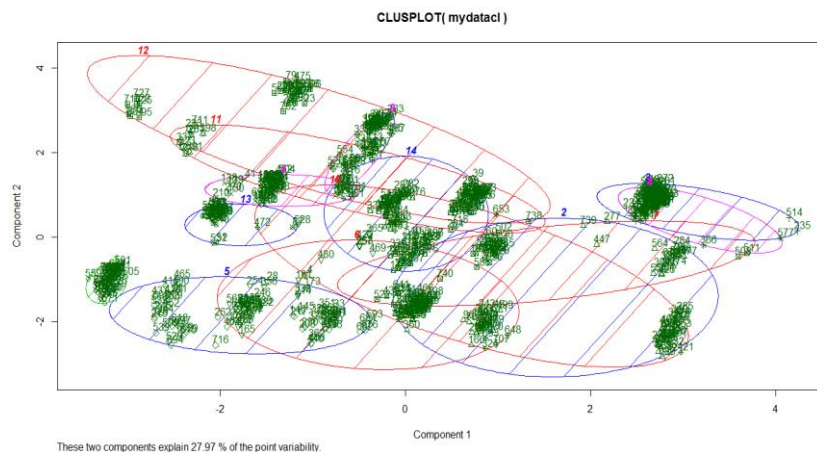


- **Plotting and Validating Cluster Solutions:**

```

60 #Plotting Cluster Solutions
61 ## K-Means Clustering with 5 clusters
62 fit1 <- kmeans(mydatacl, 14)
63
64 ## Cluster Plot against 1st 2 principal components
65 ## vary parameters for most readable graph
66 library(cluster)
67 clusplot(mydatacl, fit1$cluster, color=TRUE, shade=TRUE,
68         labels=2, lines=0)
69
70 ## Centroid Plot against 1st 2 discriminant functions
71 library(fpc)
72 plotcluster(mydatacl, fit1$cluster)
73
74
75 #validating cluster solutions
76 ##comparing 2 cluster solutions
77 library(fpc)
78 cluster.stats(d, fit1$cluster, fit2$cluster)

```



- **Centroid Plot against first 2 discriminant functions:**

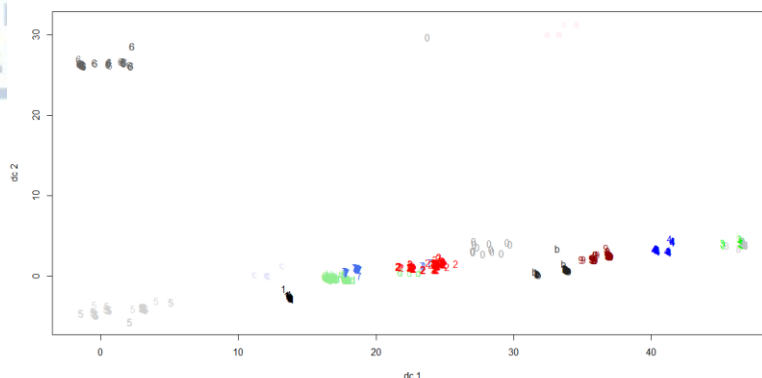


Figure 30: Comparison between *k*-Means and Hierarchical methods (*d* = Euclidean matrix distance)

```
> ccluster.stats(d, fit1$cluster, fit2$cluster)
$`n`
[1] 740

$cluster.number
[1] 14

$cluster.size
[1] 46 132 38 56 59 39 33 78 56 13 14 32 35 109

$min.cluster.size
[1] 13

$noisen
[1] 0

$diameter
[1] 7.083279 10.186640 7.298741 5.748245 9.365575 8.788980 8.071288 6.529532
[9] 8.234788 8.830087 9.061927 8.464899 6.431626 8.589190

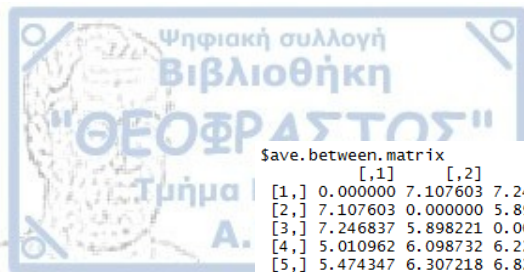
$average.distance
[1] 3.226061 5.313859 3.605969 3.024050 4.737803 4.860760 4.251936 2.492743 4.287303
[10] 6.319077 5.829528 4.245846 2.400233 4.378930

$median.distance
[1] 3.133211 5.423988 3.769823 3.019806 4.677781 4.834979 4.267165 2.438616 4.304418
[10] 6.491025 6.060566 4.165717 2.111520 4.325076

$separation
[1] 2.606884 2.314298 1.280342 1.715971 2.606884 4.168811 1.280342 1.281971 2.948516
[10] 2.314298 4.313251 3.952607 1.715971 2.894536

$average.toother
[1] 6.629284 6.461962 6.082998 5.685109 6.385154 6.995295 6.118337 5.911019 6.483181
[10] 8.916640 8.055619 7.187842 5.718022 5.738673

$separation.matrix
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,] 0.000000 4.577528 5.993571 3.372399 2.606884 5.190021 5.931441 5.991693 5.731458
[2,] 4.577528 0.000000 3.643200 3.335939 3.274370 5.005622 3.173328 2.994628 5.308592
[3,] 5.993571 3.643200 0.000000 4.714387 4.849510 5.822048 1.280342 1.281971 4.943906
[4,] 3.372399 3.335939 4.714387 0.000000 3.556872 4.814984 3.398240 4.348781 3.751313
[5,] 2.606884 3.274370 4.849510 3.556872 0.000000 5.195618 3.910960 4.844578 4.878997
[6,] 5.190021 5.005622 5.822048 4.814984 5.195618 0.000000 5.189940 5.501745 4.864360
[7,] 5.931441 3.173328 1.280342 3.398240 3.910960 5.189940 0.000000 3.628618 4.495744
[8,] 5.991693 2.994628 1.281971 4.348781 4.844578 5.501745 3.628618 0.000000 4.944759
[9,] 5.731458 5.308592 4.943906 3.751313 4.878997 4.864360 4.495744 4.944759 0.000000
[10,] 5.539840 2.314298 4.556827 5.589505 5.336025 4.352215 5.604864 5.317279 4.759233
[11,] 7.095956 6.195789 6.749593 5.384302 6.252521 4.313251 5.135918 6.539746 6.091020
[12,] 6.544866 5.277740 5.743610 4.111725 5.064696 6.087484 5.856871 5.743610 4.858986
[13,] 3.938615 3.335939 4.425671 1.715971 3.705652 4.945919 3.383374 4.771910 3.927849
[14,] 4.620265 2.894536 3.131208 3.145586 3.773608 4.168811 3.381795 3.110936 2.948516
[,10] [,11] [,12] [,13] [,14]
[1,] 5.539840 7.095956 6.544866 3.938615 4.620265
[2,] 2.314298 6.195789 5.277740 3.335939 2.894536
[3,] 4.556827 6.749593 5.743610 4.425671 3.131208
[4,] 5.589505 5.384302 4.111725 1.715971 3.145586
[5,] 5.336025 6.252521 5.064696 3.705652 3.773608
[6,] 4.352215 4.313251 6.087484 4.945919 4.168811
[7,] 5.604864 5.135918 5.856871 3.383374 3.381795
[8,] 5.317279 6.539746 5.743610 4.771910 3.110936
[9,] 4.759233 6.091020 4.858986 3.927849 2.948516
[10,] 0.000000 6.004223 5.917028 5.263860 4.529564
[11,] 6.004223 0.000000 5.342212 5.681018 5.480931
[12,] 5.917028 5.342212 0.000000 4.114461 3.952607
[13,] 5.263860 5.681018 4.114461 0.000000 3.171334
[14,] 4.529564 5.480931 3.952607 3.171334 0.000000
```



Save.between.matrix

```

[1,] [1,] [2,] [3,] [4,] [5,] [6,] [7,] [8,] [9,]
[1,] 0.000000 7.107603 7.246837 5.010962 5.474347 6.794313 7.151782 6.826761 6.856499
[2,] 7.107603 0.000000 5.898221 6.098732 6.307218 7.161153 5.953644 5.592615 7.153344
[3,] 7.246837 5.898221 0.000000 6.224383 6.828857 7.324430 5.783390 4.158386 6.540859
[4,] 5.010962 6.098732 6.224383 0.000000 5.603635 6.402301 5.704294 5.545343 5.673054
[5,] 5.474347 6.307218 6.828857 5.603635 0.000000 7.021704 6.286238 6.434934 7.003011
[6,] 6.794313 7.161153 7.324430 6.402301 7.021704 0.000000 6.890566 6.927239 6.959227
[7,] 7.151782 5.953644 5.783390 5.704294 6.286238 6.890566 0.000000 5.466495 6.244896
[8,] 6.826761 5.592615 4.158386 5.545343 6.434934 6.927239 5.466495 0.000000 6.305594
[9,] 6.856499 7.153344 6.540859 5.673054 7.003011 6.959227 6.244896 6.305594 0.000000
[10,] 9.451331 8.665366 9.003097 8.811275 8.928902 9.285055 8.847221 8.789690 9.080441
[11,] 8.740434 8.726849 8.427128 7.261609 8.248881 7.083493 7.499713 8.240449 8.082704
[12,] 7.562753 8.043194 7.433173 5.882529 7.215786 8.402539 7.551099 7.142377 6.861106
[13,] 5.316399 6.155951 5.796720 3.958815 5.687657 6.652745 5.713745 6.141965 5.584157
[14,] 6.355833 5.907370 5.173976 5.089624 5.997228 6.566253 5.451789 4.915527 5.320202
[1,] [10,] [11,] [12,] [13,] [14,]
[1,] 9.451331 8.740434 7.562753 5.316399 6.355833
[2,] 8.665366 8.726849 8.043194 6.155951 5.907370
[3,] 9.003097 8.427128 7.433173 5.796720 5.173976
[4,] 8.811275 7.261609 5.882529 3.958815 5.089624
[5,] 8.928902 8.248881 7.215786 5.687657 5.997228
[6,] 9.285055 7.083493 8.402539 6.652745 6.566253
[7,] 8.847221 7.499713 7.551099 5.713745 5.451789
[8,] 8.789690 8.240449 7.142377 6.141965 4.915527
[9,] 9.080441 8.082704 6.861106 5.584157 5.320202
[10,] 0.000000 10.463745 9.872630 8.847530 8.451350
[11,] 10.463745 0.000000 7.564186 7.223570 7.621738
[12,] 9.872630 7.564186 0.000000 5.805351 6.294936
[13,] 8.847530 7.223570 5.805351 0.000000 5.089995
[14,] 8.451350 7.621738 6.294936 5.089995 0.000000

```

Saverage.between
[1] 6.318257

Saverage.within
[1] 4.318725

\$n.between
[1] 246837

\$n.within
[1] 26593

\$max.diameter
[1] 10.18664

\$min.separation
[1] 1.280342

\$within.cluster.ss
[1] 7301.809

```

$clus.avg.silwidths
      1      2      3      4      5      6      7
0.33850169 -0.03718046 0.13474953 0.22717601 0.08936052 0.21021035 0.17208535
      8      9     10     11     12     13     14
0.40155355 0.18216492 0.18491071 0.14170839 0.24442184 0.40477645 0.03719016

```

\$avg.silwidth
[1] 0.1616317

\$g2
NULL

\$g3
NULL

\$spearsongamma
[1] 0.3747736

\$dunn
[1] 0.1256883

\$dunn2
[1] 0.6264862

\$entropy
[1] 2.464264

\$wb.ratio
[1] 0.6835311

\$ch
[1] 57.19517

```

$widgegap
[1] 4.425775 4.464624 4.754578 2.972990 4.959918 5.383833 4.971354 4.820274 4.774110
[10] 6.137389 5.587074 6.419219 4.738387 4.783293

```

\$widestgap
[1] 6.419219

\$sindex
[1] 2.062431

\$corrected.rand
NULL

\$vi
NULL

Appendix J: Quantum Clustering Statistics

```

> cluster.stats(d, fit50$cluster)
$`n`
[1] 740

$cluster.number
[1] 50

$cluster.size
[1] 10 6 8 15 5 2 13 19 27 13 21 1 19 20 37 26 7 2 14 11 73 15 13 16 4 10 16 4
[29] 16 21 5 7 15 32 18 13 6 18 2 22 21 35 11 15 6 1 20 13 7 9

$min.cluster.size
[1] 1

$noisen
[1] 0

$diameter
[1] 13.832977 8.886341 12.232701 13.808416 2.837661 5.662069 9.480929 13.922394
[9] 12.715959 5.632132 13.463793 NA 12.708883 14.471445 14.831838 14.799597
[17] 9.271680 10.656476 14.423607 12.331761 15.208115 13.609850 11.659058 14.484483
[25] 11.045163 12.890754 13.180605 8.656734 15.149545 13.760696 12.504036 11.977694
[33] 14.487477 14.730695 14.400095 13.561771 10.935733 13.498085 5.174137 13.659125
[41] 14.461169 15.227484 9.545256 10.863645 9.252812 NA 12.914762 12.808472
[49] 13.660893 9.860713

$average.distance
[1] 8.261582 4.500943 7.841124 6.660253 1.661123 5.662069 6.005294 7.673132
[9] 6.941237 1.839546 5.617246 NaN 6.713623 7.735085 7.248312 7.186483
[17] 6.689973 10.656476 5.756755 7.163586 8.248673 7.678911 5.986321 7.604239
[25] 6.968027 7.188442 7.546739 5.322820 8.636576 6.309452 7.850911 6.707375
[33] 6.780551 7.967044 7.836291 7.503970 6.195921 6.397894 5.174137 7.632097
[41] 7.713686 7.624837 4.942301 6.554822 4.613440 NaN 7.011461 6.692306
[49] 7.384147 6.360089

$median.distance
[1] 8.351320 3.175154 8.887820 6.984795 2.103642 5.662069 6.374031 7.530100
[9] 7.351727 1.406792 2.919236 NA 6.760472 7.619259 7.257573 7.083274
[17] 6.753574 10.656476 5.326361 7.209744 8.652966 7.581326 5.904525 7.383295
[25] 7.482155 6.324571 8.098367 5.462293 8.536102 6.323408 8.197390 7.567219
[33] 6.450849 7.687667 7.925559 7.282169 6.878555 6.134333 5.174137 7.576797
[41] 7.427500 7.915854 2.904001 6.866299 3.196389 NA 7.343904 7.176090
[49] 7.389406 7.294177

$separation
[1] 1.1572835 2.0184651 0.5828658 0.5592334 1.1818870 2.5941443 1.9309154 0.7281363
[9] 0.5592334 0.2932619 0.6071292 1.7593631 0.5828658 0.6401271 0.4887410 0.4887410
[17] 0.7426936 0.9655440 0.7345924 0.8599965 0.4138933 0.6505448 0.5335437 0.4287414
[25] 0.7748724 1.1168768 0.8601851 1.2819709 0.8601851 0.4138933 0.9991224 1.2027398
[33] 0.5335437 0.4029162 0.8599965 0.2932619 0.7345924 0.8255786 0.9428895 0.6376213
[41] 0.8360932 0.9586612 0.9655440 0.4029162 1.3653881 1.5216948 1.3462900 1.1432062
[49] 0.8171653 0.4989190

$average.toother
[1] 8.808362 9.308607 8.082541 8.155753 7.736758 10.506106 8.560653 8.214101
[9] 7.773072 6.986174 8.602329 8.614522 8.088263 8.162165 7.757176 7.924700
[17] 8.027200 8.654434 8.051288 8.188652 8.267195 8.363610 7.964605 8.055791
[25] 8.109972 8.792730 8.371525 8.030918 8.982156 8.129349 8.066063 8.658898
[33] 7.855743 8.302372 8.440138 8.537470 8.072068 7.949845 7.375444 8.033449
[41] 8.408345 8.106458 7.962033 8.026651 7.363817 7.292175 7.820730 7.913594
[49] 8.205961 7.964477

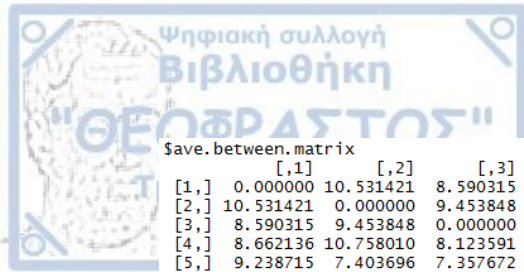
```

Sseparation.matrix

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	0.000000	4.592557	4.8596727	3.5548460	6.785440	9.122880	6.710671	4.0288075
[2,]	4.592557	0.000000	6.3746888	4.9853886	6.783060	10.815918	3.598152	3.8725872
[3,]	4.859673	6.374689	0.000000	4.0296922	4.561938	8.668794	4.667644	3.4853735
[4,]	3.554846	4.985389	4.0296922	0.000000	5.356508	6.190411	4.805961	2.1729425
[5,]	6.785440	6.783060	4.5619383	5.3565083	0.000000	8.787963	5.337872	3.9170837
[6,]	9.122880	10.815918	8.6687935	6.1904111	8.787963	0.000000	8.735393	8.8421617
[7,]	6.710671	3.598152	4.6676440	4.8059610	5.337872	8.735393	0.000000	3.9940149
[8,]	4.028808	3.872587	3.4853735	2.1729425	3.917084	8.842162	3.994015	0.0000000
[9,]	3.781643	2.307785	3.0143966	0.5592334	1.246827	6.112687	1.930915	1.2349079
[10,]	1.215619	6.217789	5.0234257	5.9072397	7.030464	9.226504	6.540681	4.1733259
[11,]	3.275996	5.481416	5.5945120	3.4012324	7.009518	8.006144	7.397541	3.0871943
[12,]	6.730999	8.406739	4.0440341	7.0375428	5.505955	6.707031	6.175849	5.2707102
[13,]	3.377924	6.510775	0.5828658	3.7529777	4.520628	7.698684	4.376792	3.9718678
[14,]	3.573144	3.717478	3.4324047	3.7508340	4.434026	4.843394	3.469075	1.7745543
[15,]	2.897448	3.646442	1.9631092	1.6030912	3.448858	6.884068	3.481933	0.7281363
[16,]	3.404009	3.538892	3.3360776	1.3891942	2.034467	8.879849	3.294451	0.8667874
[17,]	6.126670	6.482863	3.0217902	3.5299961	3.650213	7.638469	3.300160	1.0950389
[18,]	4.948899	6.691821	3.9914273	4.8174670	7.592350	8.808497	4.916601	4.1520224
[19,]	3.387929	3.525152	2.9609191	1.9446070	6.134433	9.549413	3.267705	1.5498484
[20,]	1.923329	5.720106	1.6703018	4.7930734	6.086531	7.611197	6.037725	4.4118942
	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]
[1,]	3.7816427	1.2156189	3.2759956	6.730999	3.3779241	3.5731438	2.8974477	3.4040085
[2,]	2.3077853	6.2177885	5.4814157	8.406739	6.5107754	3.7174780	3.6464424	3.5388916
[3,]	3.0143966	5.0234257	5.5945120	4.044034	0.5828658	3.4324047	1.9631092	3.3360776
[4,]	0.5592334	5.9072397	3.4012324	7.037543	3.7529777	3.7508340	1.6030912	1.3891942
[5,]	1.2468275	7.0304639	7.0095184	5.505955	4.5206276	4.4340261	3.4488584	2.0344675
[6,]	6.1126874	9.2265043	8.0061442	6.707031	7.6986836	4.8433942	6.8840676	8.8798490
[7,]	1.9309154	6.5406811	7.3975409	6.175849	4.3767920	3.4690751	3.4819326	3.2944512
[8,]	1.2349079	4.1733259	3.0871943	5.270710	3.9718678	1.7745543	0.7281363	0.8667874
[9,]	0.0000000	3.4881853	3.8946012	4.768066	3.5634041	1.6564145	1.2244384	1.1130987
[10,]	3.4881853	0.0000000	3.0737064	7.815495	5.1357461	2.9535751	2.2129070	3.6134503
[11,]	3.8946012	3.0737064	0.0000000	7.647031	5.1981305	2.3784849	1.7471300	3.2687452
[12,]	4.7680661	7.8154954	7.6470305	0.000000	1.7593631	4.9649233	4.1806336	5.5598608
[13,]	3.5634041	5.1357461	5.1981305	1.759363	0.0000000	3.4252405	2.6072991	3.2665785
[14,]	1.6564145	2.9535751	2.3784849	4.964923	3.4252405	0.0000000	1.2579182	3.1872152

[15,]	1.2244384	2.2129070	1.7471300	4.180634	2.6072991	1.2579182	0.0000000	0.4887410
[16,]	1.1130987	3.6134503	3.2687452	5.559861	3.2665785	3.1872152	0.4887410	0.0000000
[17,]	1.5472291	5.8953373	5.7281194	4.921382	3.4458867	0.9071508	0.7784948	1.1759751
[18,]	4.7760474	5.5464051	5.5204843	8.086662	1.4736069	3.5549987	3.9328004	3.5182859
[19,]	1.7951732	3.4365853	3.4679905	7.687851	4.0352159	3.2529314	0.9651352	0.7345924
[20,]	4.5099896	1.5792253	2.8381655	6.493827	1.5012590	3.8843981	2.9195336	4.1973650
	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]
[1,]	6.1266704	4.948899	3.3879290	1.9233292	2.1584120	5.5715957	4.6008687	3.1083527
[2,]	6.4828628	6.691821	3.5251516	5.7201061	2.0184651	5.9358851	6.4098344	5.4489455
[3,]	3.0217902	3.991427	2.9609191	1.6703018	2.0626208	2.4927711	2.7012931	2.2260750
[4,]	3.5299961	4.817467	1.9446070	4.7930734	2.8361108	1.5789925	3.5323112	2.6452375
[5,]	3.6502129	7.592350	6.1344327	6.0865312	1.1818870	3.5251718	4.7498365	3.3561355
[6,]	7.6384691	8.808497	9.5494129	7.6111969	2.5941443	8.7679077	8.8662015	7.7529101
[7,]	3.3001599	4.916601	3.2677050	6.0377248	2.1375142	4.5065475	3.5288833	5.3665369
[8,]	1.0950389	4.152022	1.5498484	4.4118942	1.6178126	1.6363911	1.8125686	2.7464891
[9,]	1.5472291	4.776047	1.7951732	4.5099896	0.8188387	0.6505448	2.3460848	2.3218080
[10,]	5.8953373	5.546405	3.4365853	1.5792253	1.6994358	5.7137468	5.4280432	2.4219770
[11,]	5.7281194	5.520484	3.4679905	2.8381655	1.2344925	5.5836260	5.0221093	0.8556763
[12,]	4.9213822	8.086662	7.6878506	6.4938273	5.2685480	4.4042918	4.8632496	4.6791908
[13,]	3.4458867	1.473607	4.0352159	1.5012590	2.2263342	3.3526898	2.6357795	2.9092915
[14,]	0.9071508	3.554999	3.2529314	3.8843981	1.1944190	3.5041464	1.2134649	2.7692708
[15,]	0.7784948	3.932800	0.9651352	2.9195336	0.7650601	1.3480771	1.4763477	1.1185909
[16,]	1.1759751	3.518286	0.7345924	4.1973650	1.3575839	1.9010846	2.9524821	2.7084341
[17,]	0.0000000	4.246874	3.5661698	4.6970877	1.3875469	1.9061135	1.1588343	2.4424478
[18,]	4.2468741	0.000000	5.0857886	2.3779269	2.5624015	4.4408060	2.9554358	4.3313129
[19,]	3.5661698	5.085789	0.0000000	4.1386865	0.9542216	2.2319858	3.3638852	2.8979855
[20,]	4.6970877	2.377927	4.1386865	0.0000000	1.8904049	3.8170456	3.2563587	2.6283909
	[,25]	[,26]	[,27]	[,28]	[,29]	[,30]	[,31]	[,32]
[1,]	5.6742094	5.715651	1.9089778	1.281971	1.5272219	2.6843824	2.9375301	3.183150
[2,]	6.1951639	6.315988	4.2440023	4.772799	4.0148467	4.6786546	4.4816388	4.417423
[3,]	4.2351093	3.144366	1.5985299	5.253006	2.0006000	2.1919530	4.2267389	5.469408
[4,]	2.0696365	2.865962	3.4702478	2.957216	3.9118229	3.6575459	3.4355134	2.023051
[5,]	6.1811026	5.399064	5.7906115	6.765345	5.3328582	5.0240482	6.2244292	6.316902
[6,]	10.2434323	8.386614	6.6335368	8.492399	8.6232615	8.7164846	9.7284756	5.996354
[7,]	4.9567155	5.190857	4.3367285	7.488415	4.2740462	4.6792560	5.7228122	6.456339
[8,]	1.7462216	2.763064	4.1589387	4.333916	4.5094675	3.0437125	3.3519236	1.643254

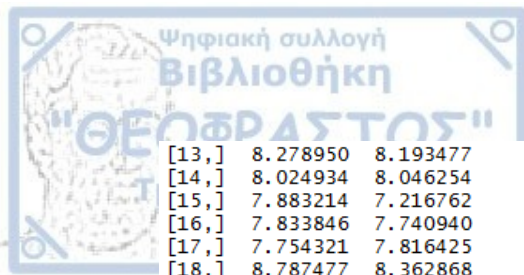
[9,]	1.6644105	2.794525	3.7193946	3.758298	4.0224709	3.1275434	2.0725681	2.450986
[10,]	6.2922371	5.181651	2.6550791	1.879355	2.8523987	1.5105372	4.7536881	3.681804
[11,]	5.3333922	4.957869	4.3870397	2.802458	4.5783760	1.1357736	2.5323143	3.004906
[12,]	7.6487101	6.976736	6.7998656	6.456302	6.5615942	4.6630542	7.1292970	7.555440
[13,]	4.2428639	3.007333	1.7352448	4.895659	1.6434509	2.3749952	4.0391850	5.689546
[14,]	2.9508946	3.003296	3.8378108	4.220752	4.2483255	2.1337088	1.0901111	3.504146
[15,]	0.7748724	1.368880	3.3059502	2.999404	3.5966600	1.4956499	0.9991224	1.480313
[16,]	0.9424180	3.808550	3.3590827	3.487132	3.5091810	3.3057762	4.8771212	1.331425
[17,]	4.1936865	5.571009	4.0516049	6.917526	4.7102352	2.6068771	4.9847049	5.598160
[18,]	5.2730289	4.925222	3.6298755	5.067074	2.4665256	2.4165851	5.5967476	6.040032
[19,]	1.1781676	3.015307	3.4678665	3.519576	3.2914298	3.3369984	5.1579273	1.760778
[20,]	4.7569240	3.120979	2.6357923	1.761257	2.9255381	2.2922577	3.9789048	3.770720
[,33]	[,34]	[,35]	[,36]	[,37]	[,38]	[,39]	[,40]	
[1,]	3.1354856	3.3094863	1.8151819	1.1572835	5.6704308	2.8350301	7.0947363	2.7182463
[2,]	3.4009919	4.4973556	5.2673313	4.7388690	5.6297186	3.8690863	5.6255889	5.1546879
[3,]	3.8706319	2.2905214	1.4667012	4.9505984	3.3903617	2.4194592	3.0670732	2.3639604
[4,]	3.3920747	3.4718964	3.8425258	3.8054992	1.9460532	1.5070473	4.0872731	3.5162263
[5,]	4.3435432	3.0259700	5.2463492	7.0710813	6.6894709	3.8388070	7.2703590	4.4330729
[6,]	8.7594743	4.8884196	9.0017076	9.2311275	9.7450682	7.6212325	8.8105307	8.2773630
[7,]	3.4575464	4.5356463	5.2641282	6.6837015	2.7417894	3.3475202	4.3311817	5.0713812
[8,]	1.5523495	2.2094299	4.0098237	4.0923319	3.3655688	1.8084973	1.4706205	2.9419557
[9,]	1.6428266	1.8941509	3.6203898	3.7258807	3.0544895	1.5938052	2.6875924	2.9084056
[10,]	2.7461504	2.8676360	2.1945859	0.2932619	6.0245168	4.0772186	5.5346723	2.1108479
[11,]	2.4510882	0.6071292	2.2072241	2.5371788	5.8460547	3.6930435	6.9674303	0.6378643
[12,]	5.8252870	4.6476553	4.4382814	7.2331133	8.4356902	4.7125359	8.9494511	4.6842844
[13,]	3.1525563	2.9312900	1.5148088	3.7757781	3.3213153	3.1350932	3.5269426	2.6528015
[14,]	0.6401271	1.3440848	3.1286465	3.1703950	3.2503356	3.5876575	1.2666855	2.3054409
[15,]	0.8827202	1.0524240	2.1999652	2.4142998	1.2290876	1.3848028	1.4945266	1.2074613
[16,]	2.9068371	2.7046401	3.9258093	3.6989942	0.7345924	1.1379972	1.6038837	2.9388331
[17,]	0.7426936	1.4817462	3.3979209	6.0213739	3.0354652	1.8758028	4.3988387	2.6419525
[18,]	3.3350395	3.1013730	1.9777001	5.1124320	3.8682984	3.7835480	5.2026178	2.7712266
[19,]	3.0131288	2.9406918	4.1888921	3.5212262	1.9338694	0.8255786	3.4577793	3.4032905
[20,]	3.6203596	2.6997501	0.8599965	1.1484657	4.7964625	4.0435325	5.4997676	2.5524338
[,41]	[,42]	[,43]	[,44]	[,45]	[,46]	[,47]	[,48]	
[1,]	4.1903663	2.2112052	1.748476	3.1389491	5.286989	4.978921	2.897185	6.031529
[2,]	3.6976639	4.3272548	6.598917	5.6351597	6.626066	4.439769	3.712988	7.294866
[3,]	3.9031609	1.4380830	1.360230	5.3503973	2.148754	4.650604	2.637844	3.026134
[4,]	4.3467862	2.1581700	3.916671	2.5092917	2.808956	4.572443	3.584637	4.158367
[5,]	4.8692602	6.0118862	5.570461	6.8428586	6.659947	7.846442	3.219833	3.538605
[6,]	6.9805292	7.1889101	8.690678	6.6242726	8.507293	9.806042	7.457932	8.160960
[7,]	3.6969029	4.2501924	4.694272	6.4065617	4.402562	5.690083	4.931040	5.622037
[8,]	1.6930215	2.4061398	3.798797	2.8074996	3.555643	3.805049	2.635184	3.423556
[9,]	1.9751886	2.0661281	3.485362	2.7539804	3.575498	3.095376	2.080799	3.384816
[10,]	5.9910162	1.9477800	5.064811	2.5684166	5.154173	4.267120	2.578869	5.438912
[11,]	2.5681055	2.0087670	2.865264	1.1420931	5.780691	5.718158	2.398937	4.818403
[12,]	5.5384979	6.9813906	6.511667	7.7046916	7.184279	8.480967	5.579917	5.690309
[13,]	3.9899742	2.4270072	1.063884	4.5428238	1.389452	4.743629	2.724765	3.358203
[14,]	0.8360932	1.8117497	2.942112	2.7048931	3.416799	3.416799	2.198176	2.379078
[15,]	1.6832040	1.3555856	2.211001	1.1767304	2.776447	2.754995	1.346290	1.852387
[16,]	3.3089154	2.0197324	3.705269	2.7422284	2.791850	4.978462	2.385207	3.515685
[17,]	1.0087768	2.4965620	3.888976	5.0545517	3.421543	5.941982	4.215657	3.661869
[18,]	3.7445668	2.4690572	0.965544	5.6610599	1.592311	5.272708	4.262198	2.800590
[19,]	3.7563169	2.6552366	4.935022	3.0371446	3.216701	4.985434	2.621849	3.843028
[20,]	4.4608382	2.6323646	1.463709	2.8893617	1.970684	2.886166	3.491982	2.938902
[,49]	[,50]							
[1,]	5.9977517	3.5614549						
[2,]	7.1414685	3.5050933						
[3,]	3.4351519	5.6580560						
[4,]	2.5952013	3.3040084						
[5,]	6.1283766	6.2328613						
[6,]	6.1373888	9.2935010						
[7,]	5.2388351	4.9390167						
[8,]	3.8571883	1.0518627						
[9,]	3.2784099	1.5857707						
[10,]	5.5939567	3.1689957						
[11,]	5.1551237	2.7851292						
[12,]	5.9729338	7.6319994						
[13,]	3.0947181	5.9030880						
[14,]	2.6544966	0.8915087						
[15,]	2.4203851	0.4989190						
[16,]	2.7381214	1.1234619						
[17,]	2.8991367	4.4659119						
[18,]	3.6054571	6.0935011						
[19,]	3.5997608	1.1781676						
[20,]	3.5926173	4.0461088						



Save. between. matrix

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	0.000000	10.531421	8.590315	8.662136	9.238715	11.066364	9.101520	9.342027
[2,]	10.531421	0.000000	9.453848	10.758010	7.403696	11.685937	9.033263	8.628090
[3,]	8.590315	9.453848	0.000000	8.123591	7.357672	10.512727	8.599461	8.553744
[4,]	8.662136	10.758010	8.123591	0.000000	8.105581	10.582832	8.578485	7.987384
[5,]	9.238715	7.403696	7.357672	8.105581	0.000000	9.794178	8.081789	7.446031
[6,]	11.066364	11.685937	10.512727	10.582832	9.794178	0.000000	11.105095	10.846328
[7,]	9.101520	9.033263	8.599461	8.578485	8.081789	11.105095	0.000000	8.515742
[8,]	9.342027	8.628090	8.553744	7.987384	7.446031	10.846328	8.515742	0.000000
[9,]	8.714579	9.124964	7.841857	7.121596	7.036352	10.281518	7.683935	7.529577
[10,]	7.233263	8.448395	6.722044	7.383692	7.292562	9.782218	7.738348	7.540801
[11,]	9.748555	7.917766	8.750239	9.634017	7.845207	10.551387	10.454822	8.350863
[12,]	9.442916	8.730040	7.816061	9.099308	5.758211	7.316992	9.420596	9.040644
[13,]	8.487534	8.892924	7.148681	8.447525	6.616774	10.273368	8.697017	8.656666
[14,]	8.957158	9.530440	8.413457	8.256017	8.065527	10.307720	8.013692	8.201127
[15,]	8.775433	8.622418	7.812294	7.610090	7.120006	10.307919	8.162160	7.477313
[16,]	8.810605	9.293520	7.910858	7.264056	7.347350	10.666498	7.847439	7.654628
[17,]	9.264920	8.885169	7.952495	8.099207	5.987846	10.125406	7.634267	7.884617
[18,]	9.229517	8.818117	8.242043	9.389880	8.373157	10.547345	9.128211	9.091871
[19,]	8.308109	10.606741	8.032359	6.635692	8.434472	11.149658	7.339491	7.860026
[20,]	8.366499	9.200238	7.810477	8.762254	7.987540	10.229772	9.399396	8.726288
	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]
[1,]	8.714579	7.233263	9.748555	9.442916	8.487534	8.957158	8.775433	8.810605
[2,]	9.124964	8.448395	7.917766	8.730040	8.892924	9.530440	8.622418	9.293520
[3,]	7.841857	6.722044	8.750239	7.816061	7.148681	8.413457	7.812294	7.910858
[4,]	7.121596	7.383692	9.634017	9.099308	8.447525	8.256017	7.610090	7.264056
[5,]	7.036352	7.292562	7.845207	5.758211	6.616774	8.065527	7.120006	7.347350
[6,]	10.281518	9.782218	10.551387	7.316992	10.273368	10.307720	10.307919	10.666498
[7,]	7.683935	7.738348	10.454822	9.420596	8.697017	8.013692	8.162160	7.847439
[8,]	7.529577	7.540801	8.350863	9.040644	8.656666	8.201127	7.477313	7.654628
[9,]	0.000000	6.669943	8.833665	8.436634	7.933845	7.528037	7.182861	7.171251
[10,]	6.669943	0.000000	7.528436	8.097373	6.689695	6.973970	6.595511	6.878914
[11,]	8.833665	7.528436	0.000000	8.327106	8.441207	8.947751	7.954633	8.976253
[12,]	8.436634	8.097373	8.327106	0.000000	6.746948	9.014403	8.320735	8.905855
[13,]	7.933845	6.689695	8.441207	6.746948	0.000000	8.376048	7.858989	8.153855
	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]
[1,]	9.264920	9.229517	8.308109	8.366499	9.114551	8.971874	8.267077	8.496460
[2,]	8.885169	8.818117	10.606741	9.200238	8.749706	10.579064	10.718004	10.489110
[3,]	7.952495	8.242043	8.032359	7.810477	8.265711	8.094286	8.020084	7.994709
[4,]	8.099207	9.389880	6.635692	8.762254	8.574040	7.450741	7.431572	7.661716
[5,]	5.987846	8.373157	8.434472	7.987540	7.588616	7.633044	8.560384	7.972608
[6,]	10.125406	10.547345	11.149658	10.229772	10.504225	10.908647	10.707835	10.373427
[7,]	7.634267	9.128211	7.339491	9.399396	8.597968	8.447879	7.361431	8.992030
[8,]	7.884617	9.091871	7.860026	8.726288	8.258328	8.197231	8.123942	8.202774
[9,]	7.439048	8.880243	7.039435	8.353583	7.978748	7.644062	7.216927	7.668888
[10,]	7.760263	7.519056	7.036496	6.002703	7.139798	7.860824	7.030011	6.785734
[11,]	8.712841	8.392076	10.227828	7.868898	8.143534	9.797183	9.934037	8.641620
[12,]	7.345481	8.700954	9.874322	8.252393	8.569450	8.881293	9.510488	8.904395
[13,]	7.769092	8.120077	8.515909	7.542938	8.147270	8.404311	8.252527	8.140878
[14,]	7.962532	8.981899	7.971591	8.602885	8.356431	8.509876	7.223323	8.047995
[15,]	7.474967	8.439283	7.492994	8.002387	7.823731	7.906437	7.722949	7.686901
[16,]	7.520851	8.678677	6.902265	8.418819	8.081952	7.635433	7.573395	7.828584
[17,]	0.000000	8.767393	7.958940	8.665671	8.050127	7.680275	7.872978	8.100626
[18,]	8.767393	0.000000	9.314336	7.734250	8.680272	9.237422	8.584268	8.938170
[19,]	7.958940	9.314336	0.000000	8.955835	8.445540	7.302612	6.839864	7.643483
[20,]	8.665671	7.734250	8.955835	0.000000	8.262550	8.880096	8.444836	8.166679
	[,25]	[,26]	[,27]	[,28]	[,29]	[,30]	[,31]	[,32]
[1,]	8.848861	8.968465	8.400615	7.931101	8.922903	9.186212	8.327799	9.774386
[2,]	10.387961	11.929702	8.857852	8.286358	9.237534	7.650671	9.488573	7.585882
[3,]	8.267080	8.838914	7.725696	8.008496	8.460989	8.036887	8.214516	8.723337
[4,]	6.973940	8.078713	8.455010	8.540604	9.216060	9.178202	8.042628	9.031481
[5,]	8.131311	9.643511	7.766931	7.244929	8.122911	7.349708	8.468212	7.180404
[6,]	11.131486	11.262456	10.519054	9.956982	11.480724	10.300485	10.900799	9.986018
[7,]	7.950416	9.382597	8.447404	8.967083	8.917297	9.400673	8.195705	10.002279

[8,]	8.031768	8.884540	8.700084	8.324653	9.376684	8.183461	8.210615	7.938145
[9,]	7.195560	8.372565	8.133136	8.022064	8.860034	8.273880	7.532971	8.490110
[10,]	7.215371	7.796276	6.834359	5.943502	7.587610	6.831364	6.477659	7.557357
[11,]	9.645617	10.125740	9.051208	7.827117	9.661488	6.211641	8.884821	7.283652
[12,]	9.619894	10.629811	8.253702	7.669608	8.846394	7.800500	9.229847	8.047039
[13,]	8.406952	9.361368	7.640789	7.698944	8.150165	7.682419	8.305132	8.490115
[14,]	7.955725	8.622853	8.785646	8.591423	9.456822	8.324537	7.649809	9.356304
[15,]	7.510995	8.557011	8.128238	7.788967	8.796203	7.593335	7.763943	7.898018
[16,]	7.230414	8.428389	8.180462	8.084453	8.755217	8.448200	7.946352	8.293303
[17,]	7.686521	9.218988	8.512419	8.232700	8.843967	8.123191	8.282950	8.542582
[18,]	9.629223	9.709322	8.119266	7.928398	8.867567	7.825275	9.020831	8.763097
[19,]	6.186682	7.751969	8.297456	8.819682	8.953443	9.444851	7.870551	9.574376
[20,]	9.093331	8.997683	7.700054	6.696315	8.389819	7.467413	8.280024	8.226841
	[,33]	[,34]	[,35]	[,36]	[,37]	[,38]	[,39]	[,40]
[1,]	8.429998	9.302972	8.880936	7.759674	9.061365	8.233502	8.631840	8.715406
[2,]	9.521942	8.900706	9.848047	10.425031	8.885027	9.943005	9.081408	9.692485
[3,]	8.159521	8.385425	7.873252	8.271626	7.930662	7.817940	7.496347	7.927177
[4,]	7.755420	8.785766	8.759051	8.496533	7.633063	6.659774	7.135040	8.264125
[5,]	8.187331	7.520936	7.878757	9.088947	7.753808	7.672677	7.611069	7.946059
[6,]	10.567012	10.120065	10.731034	11.122556	10.858780	10.683396	10.027035	10.146038
[7,]	7.639092	9.232919	9.686234	9.225574	7.394317	7.607022	6.571115	8.737104
[8,]	7.794445	8.275315	9.015630	9.096967	7.744309	7.753199	7.099712	8.273147
[9,]	7.148644	8.151342	8.420117	8.428707	7.362011	7.016573	6.560533	7.879084
[10,]	6.513798	7.384396	6.902186	6.813890	7.449468	7.116189	6.097266	6.700091
[11,]	8.853559	7.610217	8.365614	8.963054	9.192240	9.666314	8.967666	8.127739
[12,]	9.295677	8.107705	8.237779	9.244448	9.272123	8.815380	9.044251	8.564175
[13,]	8.237499	8.145840	7.496245	8.259885	8.402423	8.077526	8.009898	8.068723
[14,]	7.118821	8.257451	8.670835	8.600220	8.171183	8.050469	6.895600	8.020687
[15,]	7.436036	7.851735	8.226010	8.427651	7.459331	7.409592	6.815384	7.664426
[16,]	7.690597	8.375840	8.636251	8.721521	7.243399	7.038982	6.594068	7.955769
[17,]	7.885338	7.968144	8.613647	9.125221	7.797442	7.721153	7.254987	7.837848
[18,]	8.705996	8.507474	8.721059	9.116090	7.786784	8.897775	7.741301	8.317603
[19,]	7.352316	9.046810	8.933214	8.272073	7.078567	6.180711	6.460101	8.224126
[20,]	8.223411	8.098212	7.769063	8.161129	8.569622	8.528185	7.943661	7.823812
	[,41]	[,42]	[,43]	[,44]	[,45]	[,46]	[,47]	[,48]
[1,]	9.644497	8.407423	8.821823	8.035235	7.672916	7.596990	8.455064	8.881441
[2,]	8.846191	10.186590	8.193286	10.718346	9.982727	8.525964	9.083468	10.218499
[3,]	8.753471	8.058469	7.114294	8.142600	6.667981	6.988570	7.777955	7.801383
[4,]	8.735910	7.773235	8.746126	7.498474	6.771977	7.629797	7.871515	7.954246
[5,]	7.836766	8.381879	6.537393	8.664505	7.269070	7.877291	7.248266	7.763378
[6,]	10.547562	10.558675	10.461890	10.313783	10.107649	10.087825	10.303763	9.909173
[7,]	8.501008	8.517697	9.297557	8.810178	7.252115	7.477955	8.086295	8.842458
[8,]	8.192112	8.402746	8.546286	8.212689	7.950480	7.571294	7.951899	8.232471
[9,]	7.902489	7.629956	7.945502	7.601262	6.869410	7.080421	7.306295	7.585641
[10,]	7.802538	6.455348	6.146447	6.256446	6.152126	4.536239	6.254996	6.575584
[11,]	8.435889	9.051464	7.346746	8.862729	9.084578	8.225496	8.456330	8.420874
[12,]	8.965922	9.064087	7.132818	9.384353	7.859876	8.480967	8.507163	8.670435
[13,]	8.596822	8.204811	6.330598	8.540714	6.833529	7.482281	7.763776	7.838720
[14,]	7.797756	7.910397	8.285235	7.955660	7.405387	7.551056	7.695706	7.663780
[15,]	7.929864	7.824660	7.646453	7.699183	7.115672	6.991987	7.390107	7.563552
[16,]	8.274936	7.829150	8.286739	7.723394	6.960602	7.043169	7.529709	7.841495
[17,]	7.923259	8.212134	7.978466	8.283221	7.252614	7.916354	7.727865	7.848957
[18,]	8.818421	8.814132	8.076594	9.079488	7.185410	7.160986	8.720593	8.379220
[19,]	8.718778	7.733455	9.050234	7.439420	6.536597	7.276472	7.496090	7.914147
[20,]	8.847199	8.099609	7.222012	8.068569	7.242549	6.343108	8.134636	7.866907
	[,49]	[,50]						
[1,]	8.920582	9.226344						
[2,]	10.908835	6.839119						
[3,]	8.032920	8.361455						
[4,]	7.602328	8.755245						
[5,]	8.044629	7.074382						
[6,]	9.909595	10.463587						
[7,]	8.733742	8.614691						
[8,]	8.555543	7.365350						
[9,]	7.730846	7.657827						
[10,]	7.545885	6.372153						
[11,]	9.293189	6.969611						
[12,]	8.640813	8.538672						



```
[13,] 8.278950 8.193477
[14,] 8.024934 8.046254
[15,] 7.883214 7.216762
[16,] 7.833846 7.740940
[17,] 7.754321 7.816425
[18,] 8.787477 8.362868
[19,] 7.759546 8.748226
[20,] 8.524312 7.852285
[ reached getOption("max.print") -- omitted 30 rows ]
```

```
$average.between
[1] 8.155469
```

```
$average.within
[1] 7.452884
```

```
$n.between
[1] 264812
```

```
$n.within
[1] 8618
```

```
$max.diameter
[1] 15.22748
```

```
$min.separation
[1] 0.2932619
```

```
$within.cluster.ss
[1] 20690.04
```

```
$clus.avg.silwidths
      1      2      3      4      5      6
-0.255213123 0.297268777 -0.370231678 -0.145225479 0.709201242 0.220760176
      7      8      9     10     11     12
 0.041606660 -0.338108099 -0.293130978 0.580200982 0.036198625 0.000000000
     13     14     15     16     17     18
-0.350515295 -0.322423430 -0.366409941 -0.308136076 -0.213400620 -0.473024603
     19     20     21     22     23     24
-0.063499396 -0.368875047 -0.399282431 -0.286176574 -0.087142747 -0.307882603
     25     26     27     28     29     30
-0.247748559 -0.078425431 -0.344663759 -0.008130706 -0.366886588 -0.255980824
     31     32     33     34     35     36
-0.390369106 -0.138572777 -0.260566757 -0.371176458 -0.374469777 -0.198074499
     37     38     39     40     41     42
-0.141791789 -0.230894183 0.016833289 -0.359106302 -0.244496363 -0.315780287
     43     44     45     46     47     48
 0.138542007 -0.192758270 0.174282322 0.000000000 -0.302067963 -0.242209418
     49     50
-0.234380641 -0.269472739
```

```
$avg.silwidth
[1] -0.2408222
```

```
$g2
NULL
```

```
$g3
NULL
```

```
$spearsongamma
[1] 0.04777693
```

\$dunn
[1] 0.01925872

\$dunn2
[1] 0.4256791

\$entropy
[1] 3.646736

\$wb.ratio
[1] 0.9138512

\$sch
[1] 4.214156

\$cwiddegap
[1] 8.351320 8.398008 8.217636 8.593945 2.086924 5.662069 6.868184 6.757830
[9] 6.082538 5.168279 8.585887 0.000000 6.913267 8.835472 6.203351 6.308852
[17] 6.151083 10.656476 7.318232 6.514183 7.476343 6.883933 6.593165 6.185192
[25] 9.721613 8.555570 6.783535 8.265477 7.564265 6.568892 8.057994 9.874780
[33] 6.513373 7.908605 6.998036 8.371282 7.116128 6.929925 5.174137 6.101178
[41] 6.838734 7.534261 7.913943 6.568020 8.963878 0.000000 6.021215 6.515933
[49] 7.389406 7.293791

\$widestgap
[1] 10.65648

\$sindex
[1] 0.5152458

\$corrected.rand
NULL

\$vi
NULL

```
clusters50s100 <- qc(mydata, sigma=100, steps=21, min_d_factor=2,
  n_clusters_max=50, verbose=FALSE)
```

```
> cluster.stats(d, clusters50s100)
$`n`
[1] 740

$cluster.number
[1] 37

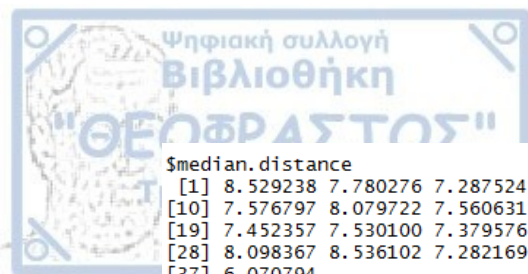
$cluster.size
[1] 36 33 32 32 29 28 25 24 23 22 22 22 21 21 20 20 19 19 19 19 18 18 18 18 16 16 16
[29] 16 15 15 15 15 12 12 11 3

$min.cluster.size
[1] 3

$noisen
[1] 0

$diameter
[1] 14.390554 14.390749 14.140623 14.730695 13.832977 14.619498 15.430741 9.545256
[9] 14.799597 13.659125 15.399716 12.535824 13.760696 14.461169 12.914762 14.471445
[17] 12.232701 13.930992 14.674408 13.922394 11.179792 12.738068 14.400095 13.498085
[25] 11.270377 14.484483 14.758325 13.180605 15.149545 14.443628 13.808416 14.487477
[33] 10.011573 12.564675 10.387813 12.331761 9.054391

$average.distance
[1] 7.471395 7.186359 7.427193 7.967044 7.715350 6.990234 6.836806 4.689323 6.917003
[10] 7.632097 8.303925 7.464250 6.309452 7.713686 7.011461 7.735085 7.037241 7.447345
[19] 7.510312 7.673132 7.254512 6.695469 7.836291 6.397894 6.577790 7.604239 7.516925
[28] 7.546739 8.636576 7.933186 6.660253 6.780551 6.312699 7.766697 6.685629 7.163586
[37] 6.783591
```

\$median.distance

```
[1] 8.529238 7.780276 7.287524 7.687667 7.766872 7.133617 6.909474 5.667276 6.870111
[10] 7.576797 8.079722 7.560631 6.323408 7.427500 7.343904 7.619259 7.368268 8.407601
[19] 7.452357 7.530100 7.379576 6.313119 7.925559 6.134333 6.879452 7.383295 7.254138
[28] 8.098367 8.536102 7.282169 6.984795 6.450849 7.118027 7.763106 7.180393 7.209744
[37] 6.070794
```

\$separation

```
[1] 0.6071292 0.4138933 0.4029162 0.4029162 0.5828658 0.8188387 0.7345924 0.2932619
[9] 0.4887410 0.6376213 0.6505448 0.4989190 0.4138933 0.8360932 1.3462900 0.6401271
[17] 0.5828658 0.8329964 0.7650601 0.7281363 1.9309154 0.5335437 0.8599965 0.8255786
[25] 0.4887410 0.4287414 0.8329964 0.8601851 0.8601851 0.2932619 0.5592334 0.5335437
[33] 0.5592334 0.7650601 0.8188387 0.8599965 3.5182859
```

\$average.toothor

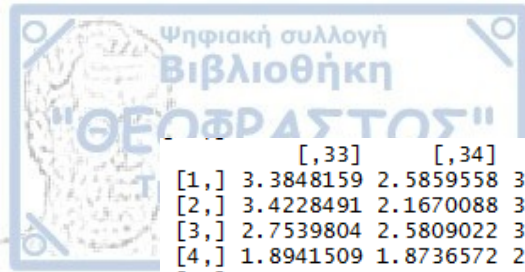
```
[1] 8.451135 8.487476 8.322183 8.302372 8.335850 8.176421 7.998882 7.455540 7.880133
[10] 8.033449 8.447767 7.959746 8.129349 8.408345 7.820730 8.162165 7.856035 8.015633
[19] 7.852352 8.214101 8.795726 7.999256 8.440138 7.949845 7.638963 8.055791 8.196063
[28] 8.371525 8.982156 8.550578 8.155753 7.855743 7.725920 8.123626 7.816461 8.188652
[37] 8.273792
```

\$separation.matrix

```
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
[1,] 0.0000000 1.2344925 1.1420931 0.6071292 3.2759956 3.8299705 3.4679905 2.5230939
[2,] 1.2344925 0.0000000 1.5214328 1.2813536 2.1584120 3.7877298 3.3865336 1.2881564
[3,] 1.1420931 1.5214328 0.0000000 0.4029162 3.0073327 3.4662523 3.0153073 2.5684166
[4,] 0.6071292 1.2813536 0.4029162 0.0000000 2.9312900 3.3598816 2.9406918 2.6984901
[5,] 3.2759956 2.1584120 3.0073327 2.9312900 0.0000000 3.8343103 3.3213153 1.0638841
[6,] 3.8299705 3.7877298 3.4662523 3.3598816 3.8343103 0.0000000 0.9542216 3.9250841
[7,] 3.4679905 3.3865336 3.0153073 2.9406918 3.3213153 0.9542216 0.0000000 3.4365853
[8,] 2.5230939 1.2881564 2.5684166 2.6984901 1.0638841 3.9250841 3.4365853 0.0000000
[9,] 3.2687452 3.3537080 2.7381214 2.7046401 3.2665785 1.3575839 0.7345924 3.6134503
[10,] 0.6378643 0.6537258 1.1168768 0.9538779 2.6528015 3.4984092 3.4032905 2.1108479
[11,] 3.0049059 3.2196429 2.3820298 2.5167169 3.1831496 2.4479930 1.7607778 3.6818042
[12,] 2.7851292 2.6980562 2.3673032 1.4817462 3.4458867 1.3875469 1.1781676 3.1689957
[13,] 1.1357736 0.4138933 1.5291663 1.2364017 2.3749952 3.6401666 3.3369984 1.5105372
[14,] 2.5681055 2.8396178 2.9297949 1.6052000 3.9899742 3.3218517 3.7563169 3.6047365
[15,] 2.3989368 2.0915008 2.1480961 1.8643179 2.7247646 2.2327063 2.6218488 2.4962796
[16,] 2.3784849 2.3319566 2.6544966 1.3440848 3.4252405 3.1638124 3.2503356 2.9421124
[17,] 2.8024580 2.0626208 2.5265960 2.2905214 0.5828658 3.2882449 2.5976890 1.3602295
[18,] 2.0087670 1.5201958 1.2780119 1.2922590 2.2112052 2.6379374 2.1145976 1.9477800
[19,] 1.7471300 1.5898117 1.1767304 1.0524240 2.6072991 2.3833527 2.0643118 2.2110007
[20,] 3.0871943 3.1794234 2.7630644 2.2094299 3.9718678 1.6178126 1.5498484 3.7987969
[21,] 5.4814157 4.8677987 5.1908574 4.4973556 4.3767920 2.0184651 2.7417894 4.6942716
[22,] 2.4430441 2.1724731 2.6018345 1.1342599 2.6357795 3.1843351 3.3638852 2.5847631
[23,] 2.0800140 1.1150850 2.2494376 1.9594914 1.5148088 4.2488196 4.1888921 1.1083978
[24,] 3.6930435 3.5181423 3.1138274 3.1057875 2.8350301 1.7205320 0.8255786 3.9527860
[25,] 2.8887689 3.1744640 2.4463099 2.3953143 3.4741950 1.3556488 0.9651352 3.5494101
[26,] 0.8556763 1.4447881 0.5263483 0.4287414 2.9092915 3.4106971 2.8979855 2.4219770
[27,] 2.0518042 1.7975322 2.2806337 1.1557257 2.4842267 3.9336475 3.4963848 2.2913858
```

```
      [,9]      [,10]      [,11]      [,12]      [,13]      [,14]      [,15]      [,16]
[1,] 3.2687452 0.6378643 3.0049059 2.7851292 1.1357736 2.5681055 2.398937 2.3784849
[2,] 3.3537080 0.6537258 3.2196429 2.6980562 0.4138933 2.8396178 2.091501 2.3319566
[3,] 2.7381214 1.1168768 2.3820298 2.3673032 1.5291663 2.9297949 2.148096 2.6544966
[4,] 2.7046401 0.9538779 2.5167169 1.4817462 1.2364017 1.6052000 1.864318 1.3440848
[5,] 3.2665785 2.6528015 3.1831496 3.4458867 2.3749952 3.9899742 2.724765 3.4252405
[6,] 1.3575839 3.4984092 2.4479930 1.3875469 3.6401666 3.3218517 2.232706 3.1638124
[7,] 0.7345924 3.4032905 1.7607778 1.1781676 3.3369984 3.7563169 2.621849 3.2503356
[8,] 3.6134503 2.1108479 3.6818042 3.1689957 1.5105372 3.6047365 2.496280 2.9421124
[9,] 0.0000000 2.9388331 1.3314247 0.9424180 3.3057762 3.3089154 2.385207 3.1872152
[10,] 2.9388331 0.0000000 2.8221715 2.6221923 0.6376213 2.5331210 2.056467 2.3054409
[11,] 1.3314247 2.8221715 0.0000000 1.9061135 3.2932870 3.7146276 2.902921 3.5041464
[12,] 0.9424180 2.6221923 1.9061135 0.0000000 2.6068771 1.0087768 2.343307 0.8915087
[13,] 3.3057762 0.6376213 3.2932870 2.6068771 0.0000000 2.5228212 2.383067 2.1337088
[14,] 3.3089154 2.5331210 3.7146276 1.0087768 2.5228212 0.0000000 2.481273 0.8360932
[15,] 2.3852067 2.0564668 2.9029211 2.3433069 2.3830674 2.4812730 0.0000000 2.1981760
[16,] 3.1872152 2.3054409 3.5041464 0.8915087 2.1337088 0.8360932 2.198176 0.0000000
[17,] 2.7918503 2.2977706 2.4927711 3.0217902 2.1919530 3.9031609 2.452226 3.4167994
[18,] 2.0197324 1.5445020 1.7897789 1.6368392 1.7736615 3.9021881 1.622591 1.8117497
[19,] 1.8351312 2.2074613 1.9505626 1.5317679 1.4956499 1.6832040 1.346290 1.2579182
[20,] 0.8667874 2.9419557 1.6363911 1.0518627 3.0437125 1.6930215 2.635184 1.7745543
[21,] 3.2944512 5.0713812 4.4174232 3.3001599 4.6786546 3.6969029 3.712988 3.4690751
[22,] 2.9524821 2.0836829 2.9834217 1.1588343 2.0659552 1.6174607 2.342106 1.0901111
[23,] 3.9258093 1.5483589 3.2568170 3.3226038 1.5248503 3.6655618 2.865692 3.1286465
[24,] 1.1379972 3.4084648 1.2027398 1.8758028 3.5609281 3.8371953 2.941405 3.5876575
[25,] 0.4887410 2.6935414 1.3480771 0.4989190 3.0475756 3.0284161 2.144543 2.9104041
[26,] 2.7084341 1.1068910 2.2543679 2.4424478 1.4909326 2.9130154 2.234511 2.7692708
[27,] 3.2274019 1.8126281 2.9920441 2.3307260 2.0536592 2.5041285 2.847338 1.9847352
```

	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]
[1,]	2.8024580	2.0087670	1.7471300	3.0871943	5.481416	2.4430441	2.0800140	3.6930435
[2,]	2.0626208	1.5201958	1.5898117	3.1794234	4.867799	2.1724731	1.1150850	3.5181423
[3,]	2.5265960	1.2780119	1.1767304	2.7630644	5.190857	2.6018345	2.2494376	3.1138274
[4,]	2.2905214	1.2922590	1.0524240	2.2094299	4.497356	1.1342599	1.9594914	3.1057875
[5,]	0.5828658	2.2112052	2.6072991	3.9718678	4.376792	2.6357795	1.5148088	2.8350301
[6,]	3.2882449	2.6379374	2.3833527	1.6178126	2.018465	3.1843351	4.2488196	1.7205320
[7,]	2.5976890	2.1145976	2.0643118	1.5498484	2.741789	3.3638852	4.1888921	0.8255786
[8,]	1.3602295	1.9477800	2.2110007	3.7987969	4.694272	2.5847631	1.1083978	3.9527860
[9,]	2.7918503	2.0197324	1.8351312	0.8667874	3.294451	2.9524821	3.9258093	1.1379972
[10,]	2.2977706	1.5445020	1.2074613	2.9419557	5.071381	2.0836829	1.5483589	3.4084648
[11,]	2.4927711	1.7897789	1.9505626	1.6363911	4.417423	2.9834217	3.2568170	1.2027398
[12,]	3.0217902	1.6368392	1.5317679	1.0518627	3.300160	1.1588343	3.3226038	1.8758028
[13,]	2.1919530	1.7736615	1.4956499	3.0437125	4.678655	2.0659552	1.5248503	3.5609281
[14,]	3.9031609	3.9021881	1.6832040	1.6930215	3.696903	1.6174607	3.6655618	3.8371953
[15,]	2.4522262	1.6225906	1.3462900	2.6351840	3.712988	2.3421064	2.8656922	2.9414045
[16,]	3.4167994	1.8117497	1.2579182	1.7745543	3.469075	1.0901111	3.1286465	3.5876575
[17,]	0.0000000	1.4380830	1.9631092	3.4853735	4.402562	2.7012931	1.4667012	2.4194592
[18,]	1.4380830	0.0000000	1.8020222	2.4899137	4.250192	2.0398320	2.0079564	2.1339592
[19,]	1.9631092	1.8020222	0.0000000	2.0819668	4.055484	0.9991224	2.1999652	2.2372389
[20,]	3.4853735	2.4899137	2.0819668	0.0000000	3.872587	1.8125686	4.0098237	1.8084973
[21,]	4.4025622	4.2501924	4.0554835	3.8725872	0.000000	3.5288833	5.2641282	3.3475202
[22,]	2.7012931	2.0398320	0.9991224	1.8125686	3.528883	0.0000000	2.5546945	3.0331844
[23,]	1.4667012	2.0079564	2.1999652	4.0098237	5.264128	2.5546945	0.0000000	3.5563660
[24,]	2.4194592	2.1339592	2.2372389	1.8084973	3.347520	3.0331844	3.5563660	0.0000000
[25,]	2.8258459	1.8715584	1.5651811	0.7281363	3.481933	2.7903572	3.5726025	1.3848028
[26,]	2.2260750	1.2213168	1.1185909	2.7464891	5.366537	2.4503543	1.9157574	2.9605699
[27,]	2.5591066	0.8329964	1.3555856	2.4061398	4.605306	1.3271355	1.9712943	3.3792637
	[,25]	[,26]	[,27]	[,28]	[,29]	[,30]	[,31]	[,32]
[1,]	2.8887689	0.8556763	2.0518042	4.3767666	4.5783760	2.5371788	3.4012324	2.4510882
[2,]	3.1744640	1.4447881	1.7975322	3.3488081	3.4803410	1.3216675	3.4878929	2.2752337
[3,]	2.4463099	0.5263483	2.2806337	3.8778217	4.0626247	2.5627977	2.5092917	2.5719764
[4,]	2.3953143	0.4287414	1.1557257	3.8029875	4.1542369	2.3393115	3.4718964	1.1677719
[5,]	3.4741950	2.9092915	2.4842267	1.7352448	1.5272219	1.1572835	3.5548460	3.1354856
[6,]	1.3556488	3.4106971	3.9336475	3.6519907	3.7291434	4.0194212	2.8361108	3.1648433
[7,]	0.9651352	2.8979855	3.4963848	2.9953421	3.2914298	3.5212262	1.9446070	3.0131288
[8,]	3.5494101	2.4219770	2.2913858	2.4724767	2.5560711	0.2932619	3.9166713	2.7461504
[9,]	0.4887410	2.7084341	3.2274019	3.3590827	3.5091810	3.6989942	1.3891942	2.9068371
[10,]	2.6935414	1.1068910	1.8126281	3.6750635	4.0196339	2.1393175	3.5162263	2.1959469
[11,]	1.3480771	2.2543679	2.9920441	3.0685073	3.4641658	3.5155594	1.5789925	3.1682907
[12,]	0.4989190	2.4424478	2.3307260	4.0023688	3.9404957	3.3870840	2.0696365	0.7426936
[13,]	3.0475756	1.4909326	2.0536592	3.6766848	3.8793463	1.6066312	3.6575459	2.1243846
[14,]	3.0284161	2.9130154	2.5041285	4.8266583	4.8949660	3.7276317	4.3467862	1.1783049
[15,]	2.1445428	2.2345110	2.8473382	3.5050776	3.4891825	2.7895433	3.5846370	2.4940700
[16,]	2.9104041	2.7692708	1.9847352	3.8378108	4.2483255	3.1703950	3.7508340	0.6401271
[17,]	2.8258459	2.2260750	2.5591066	1.5985299	1.8341436	1.5923113	2.8089559	3.0940373
[18,]	1.8715584	1.2213168	0.8329964	2.6761411	3.0443163	1.9735497	2.1581700	1.4195476
[19,]	1.5651811	1.1185909	1.3555856	3.3059502	3.5966600	2.4142998	2.9675270	0.8827202
[20,]	0.7281363	2.7464891	2.4061398	4.1589387	4.5094675	4.0923319	2.1729425	1.5523495
[21,]	3.4819326	5.3665369	4.6053060	4.2440023	4.0148467	4.7388690	4.8059610	3.4009919
[22,]	2.7903572	2.4503543	1.3271355	3.5246470	3.5766193	2.7329714	3.4355134	0.5335437
[23,]	3.5726025	1.9157574	1.9712943	2.7803274	2.9694786	1.2323593	3.8425258	3.3837710
[24,]	1.3848028	2.9605699	3.3792637	2.5150165	2.8076884	3.5519056	1.5070473	3.2008962
[25,]	0.0000000	2.3414522	3.1789822	3.5117655	3.6874138	3.6498387	1.6030912	2.8258214
[26,]	2.3414522	0.0000000	2.2013574	4.1265956	3.9720661	2.4076324	2.6452375	2.6594985
[27,]	3.1789822	2.2013574	0.0000000	3.4332513	3.7078345	2.2912970	3.2461871	1.5381261



```

      [,33]      [,34]      [,35]      [,36]      [,37]
[1,] 3.3848159 2.5859558 3.8946012 2.8381655 4.423738
[2,] 3.4228491 2.1670088 3.4944858 1.8904049 3.751002
[3,] 2.7539804 2.5809022 3.8798589 2.8893617 4.493669
[4,] 1.8941509 1.8736572 2.7945252 2.6997501 4.707954
[5,] 3.8448839 2.5914060 3.5634041 1.5012590 3.719989
[6,] 2.4533189 3.4196440 0.8188387 4.7906742 5.490291
[7,] 1.7951732 3.0143556 1.2468275 4.1386865 4.544984
[8,] 3.9024145 2.3329721 3.4853622 1.4637095 3.837244
[9,] 1.1130987 3.0004581 1.6997574 4.1973650 5.014878
[10,] 2.9084056 2.0450398 3.5917921 2.5524338 4.467062
[11,] 0.6505448 3.3522411 3.9802894 3.7707201 5.107465
[12,] 1.6644105 1.0539945 1.5472291 4.0461088 4.511402
[13,] 3.1275434 2.0446749 3.2710883 2.2922577 4.219489
[14,] 3.5217267 1.7697752 1.9751886 4.4608382 4.783698
[15,] 3.0655591 2.2625782 2.0807986 3.4919819 4.556859
[16,] 3.2486927 1.1944190 1.6564145 3.8843981 4.906549
[17,] 3.0143966 2.7249437 3.7104810 1.6703018 4.105957
[18,] 2.0661281 0.9586612 2.6778765 3.3564397 4.909568
[19,] 2.0442096 0.7650601 2.2996684 2.9195336 4.467226
[20,] 1.2349079 1.8883036 2.2051266 4.4118942 4.694028
[21,] 4.4996930 3.6444658 1.9309154 5.7201061 6.194222
[22,] 3.0468798 1.1765562 2.0725681 3.2563587 4.695554
[23,] 3.6203898 2.5849522 4.0166106 0.8599965 4.324702
[24,] 1.5938052 3.5216316 3.4351354 4.0435325 5.178385
[25,] 1.2244384 2.9367463 2.7189819 4.2125959 5.447132
[26,] 2.3218080 2.5240280 3.8308976 2.6283909 3.974919
[27,] 3.1036959 1.5232314 3.1268146 2.6323646 4.454673
[ reached getOption("max.print") -- omitted 10 rows ]

```

Save.between.matrix

```

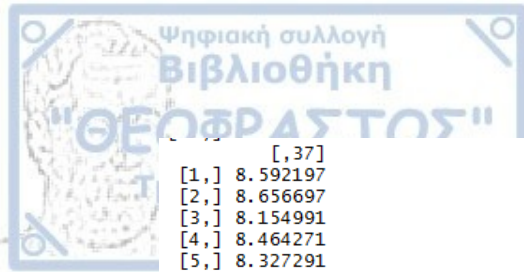
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
[1,] 0.000000 7.474736 8.779108 7.843537 8.733434 9.146619 9.004143 7.439420 8.669275
[2,] 7.474736 0.000000 9.135763 7.953938 8.413628 9.311369 9.165676 7.050241 8.817487
[3,] 8.779108 9.135763 0.000000 8.655234 8.664263 8.243661 8.017266 7.859345 7.943459
[4,] 7.843537 7.953938 8.655234 0.000000 8.544851 8.740952 8.631192 7.601959 8.364305
[5,] 8.733434 8.413628 8.664263 8.544851 0.000000 8.649897 8.304075 7.020365 8.387237
[6,] 9.146619 9.311369 8.243661 8.740952 8.649897 0.000000 7.163792 7.986799 7.133106
[7,] 9.004143 9.165676 8.017266 8.631192 8.304075 7.163792 0.000000 7.792722 6.925786
[8,] 7.439420 7.050241 7.859345 7.601959 7.020365 7.986799 7.792722 0.000000 7.531931
[9,] 8.669275 8.817487 7.943459 8.364305 8.387237 7.133106 6.925786 7.531931 0.000000
[10,] 7.968585 8.207168 7.809279 7.952651 8.291717 8.214733 8.132045 7.295299 7.982405
[11,] 8.880014 9.029104 8.578299 8.623438 8.705959 8.133955 7.900032 8.178805 7.715476
[12,] 8.198179 8.302619 8.410294 7.969799 8.437594 7.557549 7.638920 7.376875 7.372690
[13,] 7.044991 6.737751 8.965701 7.452545 8.200968 8.900163 8.801571 6.835357 8.465443
[14,] 8.366360 8.530654 8.791214 8.140204 8.958089 8.468439 8.462624 8.023684 8.256515
[15,] 8.188810 8.172758 7.736920 8.175248 8.002151 7.587646 7.549321 6.756844 7.487109
[16,] 8.559649 8.778729 8.179311 8.257451 8.576431 8.128542 8.038280 7.574966 7.998339
[17,] 8.356621 8.265243 7.975978 8.131297 7.510931 7.953743 7.626673 6.808211 7.669446
[18,] 8.550563 8.567618 7.728848 8.438615 8.177400 8.002565 7.863572 6.975484 7.675047
[19,] 7.693075 7.707528 8.214079 7.723743 8.175405 8.053674 7.993990 6.883580 7.724794
[20,] 8.446747 8.630239 8.497642 8.275315 8.892997 7.786703 7.749455 8.001648 7.493636
[21,] 9.608461 9.448283 9.656410 9.128009 9.032739 7.748874 8.149049 8.414612 8.242287
[22,] 8.887841 9.182962 7.551871 8.373087 8.272934 7.788049 7.532701 7.710681 7.664243
[23,] 8.324262 8.006251 8.656409 8.404133 7.973724 9.196066 8.766059 6.823934 8.638869
[24,] 9.121994 9.296310 7.800842 8.675694 8.131311 7.240301 6.659722 7.829688 6.872773
[25,] 8.209169 8.393631 7.789467 7.986837 8.174580 7.071427 6.794219 7.281562 6.631874
[26,] 8.231837 8.526481 7.516873 8.256880 8.263492 8.200778 7.851924 7.369631 7.785384
[27,] 8.784782 8.956634 7.781233 8.371167 8.390193 8.352901 8.130136 7.561341 7.983235

```

	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]	[,17]	[,18]
[1,]	7.968585	8.880014	8.198179	7.044991	8.366360	8.188810	8.559649	8.356621	8.550563
[2,]	8.207168	8.209104	8.302619	6.737751	8.530654	8.172758	8.778729	8.265243	8.567618
[3,]	7.809279	8.578299	8.410294	8.965701	8.791214	7.736920	8.179311	7.975978	7.728848
[4,]	7.952651	8.623438	7.969799	7.452545	8.140204	8.175248	8.257451	8.131297	8.438615
[5,]	8.291717	8.705959	8.437594	8.200968	8.958089	8.002151	8.576431	7.510931	8.177400
[6,]	8.214733	8.133955	7.557549	8.900163	8.468439	7.587646	8.128542	7.953743	8.002565
[7,]	8.132045	7.900032	7.638920	8.801571	8.462624	7.549321	8.038280	7.626673	7.863572
[8,]	7.295299	8.178805	7.376875	6.835357	8.023684	6.756844	7.574966	6.808211	6.975484
[9,]	7.982405	7.715476	7.372690	8.465443	8.256515	7.487109	7.998339	7.669446	7.675047
[10,]	0.000000	8.405519	7.897065	7.729118	8.295101	7.714367	8.020687	7.645908	7.821914
[11,]	8.405519	0.000000	8.098136	8.668997	8.922096	8.358529	8.779194	8.078088	8.381691
[12,]	7.897065	8.098136	0.000000	7.810052	7.917012	7.613202	7.898550	7.916195	7.979378
[13,]	7.729118	8.668997	7.810052	0.000000	8.046887	7.971562	8.324537	7.883868	8.423932
[14,]	8.295101	8.922096	7.917012	8.046887	0.000000	8.074838	7.797756	8.526012	8.454308
[15,]	7.714367	8.358529	7.613202	7.971562	8.074838	0.000000	7.695706	7.612780	7.432377
[16,]	8.020687	8.779194	7.898550	8.324537	7.797756	7.695706	0.000000	8.133556	7.950331
[17,]	7.645908	8.078088	7.916195	7.883868	8.526012	7.612780	8.133556	0.000000	7.640166
[18,]	7.821914	8.381691	7.979378	8.423932	8.454308	7.432377	7.950331	7.640166	0.000000
[19,]	7.630726	8.231917	7.514099	7.247853	7.932459	7.547344	7.821725	7.719909	7.842285
[20,]	8.273147	8.114795	7.627589	8.183461	8.192112	7.951899	8.201127	8.302169	8.314627
[21,]	9.038803	9.158533	8.104510	8.848041	8.610013	8.401191	8.492665	8.583026	8.973951
[22,]	7.800120	8.471463	7.992911	8.833379	7.961307	7.651962	7.341791	7.701250	7.615715
[23,]	8.263894	8.953444	8.571394	7.901111	8.930505	8.106569	8.670835	7.858395	8.285606
[24,]	8.142940	7.766448	7.767969	8.959926	8.610347	7.674651	8.050469	7.414702	7.741026
[25,]	7.699997	7.557368	7.106744	7.958009	7.927125	7.224135	7.722977	7.438586	7.523315
[26,]	7.694161	8.319637	8.081945	8.342237	8.490026	7.553803	8.047995	7.841525	7.663062
[27,]	7.957137	8.649951	8.223856	8.665409	8.366376	7.998212	7.862975	7.822685	7.767273

	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]	[,25]	[,26]	[,27]
[1,]	7.693075	8.446747	9.608461	8.887841	8.324262	9.121994	8.209169	8.231837	8.784782
[2,]	7.707528	8.630239	9.448283	9.182962	8.006251	9.296310	8.393631	8.526481	8.956634
[3,]	8.214079	8.497642	9.656410	7.551871	8.656409	7.800842	7.789467	7.516873	7.781233
[4,]	7.723743	8.275315	9.128009	8.373087	8.404133	8.675694	7.986837	8.256880	8.371167
[5,]	8.175405	8.892997	9.032739	8.272934	7.973724	8.131311	8.174580	8.263492	8.390193
[6,]	8.053674	7.786703	7.748874	7.788049	9.196066	7.240301	7.071427	8.200778	8.352901
[7,]	7.993990	7.749455	8.149049	7.532701	8.766059	6.659722	6.794219	7.851924	8.130136
[8,]	6.883580	8.001648	8.414612	7.710681	6.823934	7.829688	7.281562	7.369631	7.613411
[9,]	7.724794	7.493636	8.242287	7.664243	8.638869	6.872773	6.631874	7.785384	7.983235
[10,]	7.630726	8.273147	9.038803	7.800120	8.263894	8.142940	7.699997	7.694161	7.957137
[11,]	8.231917	8.114795	9.158533	8.471463	8.953444	7.766448	7.557368	8.319637	8.649951
[12,]	7.514099	7.627589	8.104510	7.992911	8.571394	7.767969	7.106744	8.081945	8.223856
[13,]	7.247853	8.183461	8.848041	8.833379	7.901111	8.959926	7.958009	8.342237	8.665409
[14,]	7.932459	8.192112	8.610013	7.961307	8.930505	8.610347	7.927125	8.490026	8.366376
[15,]	7.547344	7.951899	8.401191	7.651962	8.106569	7.674651	7.224135	7.553803	7.998212
[16,]	7.821725	8.201127	8.492665	7.341791	8.670835	8.050469	7.722977	8.047995	7.862975
[17,]	7.719909	8.302169	8.583026	7.701250	7.858395	7.414702	7.438586	7.841525	7.822685
[18,]	7.842285	8.314627	8.973951	7.615715	8.285606	7.741026	7.532315	7.663062	7.767273
[19,]	0.000000	7.761120	8.488208	7.928117	8.072084	8.074198	7.417282	7.835592	8.025525
[20,]	7.761120	0.000000	8.551221	8.148018	9.015630	7.753199	7.177739	8.202774	8.507388
[21,]	8.488208	8.551221	0.000000	8.472117	9.737333	8.344701	8.116762	9.464792	9.128749
[22,]	7.928117	8.148018	8.472117	0.000000	8.598465	7.309764	7.529789	7.663414	7.218161
[23,]	8.072084	9.015630	9.737333	8.598465	0.000000	8.743703	8.388488	8.199563	8.390223
[24,]	8.074198	7.753199	8.344701	7.309764	8.743703	0.000000	6.708062	7.764538	7.860575
[25,]	7.417282	7.177739	8.116762	7.529789	8.388488	6.708062	0.000000	7.529950	7.937703
[26,]	7.835592	8.202774	9.464792	7.663414	8.199563	7.764538	7.529950	0.000000	7.903781
[27,]	8.025525	8.507388	9.128749	7.218161	8.390223	7.860575	7.937703	7.903781	0.000000

	[,28]	[,29]	[,30]	[,31]	[,32]	[,33]	[,34]	[,35]	[,36]
[1,]	8.982562	9.655131	8.809665	9.080145	8.417081	8.454135	8.417514	8.475039	7.999339
[2,]	8.699489	9.221376	8.590417	9.336134	8.681584	8.782564	8.476934	8.431236	7.750698
[3,]	8.879614	9.538878	8.321495	7.702517	7.736533	7.527039	8.447320	8.300143	8.458611
[4,]	8.740646	9.359549	8.877359	8.785766	8.072953	8.137735	8.204864	8.168350	8.098212
[5,]	7.902798	8.416626	8.142764	8.521529	8.303878	8.241729	8.271057	8.154732	7.826925
[6,]	8.363046	8.840024	8.934750	7.782591	7.786289	7.352345	8.237006	7.399673	8.895105
[7,]	8.082404	8.753310	8.624283	7.169039	7.585954	6.903346	8.142999	7.382350	8.669485
[8,]	7.333027	7.909034	7.474761	8.008141	7.362091	7.421840	7.112307	7.045491	6.561553
[9,]	8.170369	8.732734	8.735659	7.087751	7.624634	6.702196	8.022084	7.428754	8.486385
[10,]	8.416689	9.093211	8.306416	8.264125	7.695195	7.750815	8.101574	8.039419	7.823812
[11,]	8.479286	9.116544	9.177690	7.953704	8.420371	7.526293	8.658912	8.396967	8.672242
[12,]	8.423582	8.899121	8.872573	8.075341	7.657190	7.447865	7.868143	7.350098	8.345042
[13,]	8.358859	9.981302	8.442831	9.178202	8.188356	8.471059	8.197287	8.027406	7.467413
[14,]	9.148704	9.713131	9.237037	8.735910	7.691706	8.117228	7.961460	7.634066	8.847199
[15,]	8.300292	8.815864	8.114427	7.871515	7.545481	7.420754	7.687108	7.163221	8.134636
[16,]	8.785646	9.456822	8.651111	8.256017	7.118821	7.730763	7.726850	7.274630	8.602885
[17,]	7.376155	8.088354	8.034330	7.825606	7.795176	7.430336	7.931002	7.762200	7.365994
[18,]	8.254458	8.924032	8.162772	7.663556	7.676822	7.393876	7.731259	7.674512	8.125716
[19,]	8.175788	8.860974	8.382733	8.246731	7.507434	7.644700	7.767818	7.472908	7.736636
[20,]	8.700084	9.376684	9.096287	7.987384	7.794445	7.409093	8.336032	7.680182	8.726288
[21,]	8.577019	9.018424	9.527808	9.266756	8.233677	8.692461	8.532319	7.447166	9.336504
[22,]	8.381683	9.120788	8.193283	7.601310	6.764045	7.293190	7.696654	7.319127	8.399055
[23,]	8.476971	8.966250	8.481025	8.759051	8.510266	8.386483	8.366157	8.462159	7.790603
[24,]	7.881073	8.613537	8.364995	6.659774	7.449757	6.611314	8.199294	7.523145	8.528185
[25,]	8.078047	8.727834	8.478252	6.938081	7.360671	6.474211	7.841875	7.153142	8.282902
[26,]	8.639180	9.332413	8.142294	7.661716	7.656872	7.379870	8.140158	8.030161	8.166679
[27,]	8.488602	9.082029	8.388231	7.900153	7.415505	7.644832	7.987073	7.908883	8.068608



```
[ ,37]
[1,] 8.592197
[2,] 8.656697
[3,] 8.154991
[4,] 8.464271
[5,] 8.327291
[6,] 8.261303
[7,] 8.202972
[8,] 7.464645
[9,] 8.192092
[10,] 7.751557
[11,] 8.835898
[12,] 8.249220
[13,] 8.316002
[14,] 8.416162
[15,] 7.856304
[16,] 8.309606
[17,] 7.647083
[18,] 7.790019
[19,] 8.131168
[20,] 8.888902
[21,] 8.777971
[22,] 7.774750
[23,] 8.616182
[24,] 8.313255
[25,] 8.113279
[26,] 8.159784
[27,] 8.097282
[ reached getOption("max.print") -- omitted 10 rows ]
```

```
$average.between
[1] 8.159127
```

```
$average.within
[1] 7.258307
```

```
$n.between
[1] 265598
```

```
$n.within
[1] 7832
```

```
$max.diameter
[1] 15.43074
```

```
$min.separation
[1] 0.2932619
```

```
$within.cluster.ss
[1] 21624.84
```

```

$clus.avg.silwidths
      1      2      3      4      5      6      7
-0.18600683 -0.19105681 -0.07470571 -0.20180429 -0.17162322 -0.02958507 -0.09421220
      8      9     10     11     12     13     14
 0.23011431 -0.11621389 -0.16371248 -0.17555533 -0.16998618 -0.05712513 -0.14024236
     15     16     17     18     19     20     21
-0.09537155 -0.18269423 -0.09418160 -0.12097353 -0.22059536 -0.16767772 -0.04950989
     22     23     24     25     26     27     28
-0.02204207 -0.20871981 -0.02698808 -0.11494825 -0.14587012 -0.13966252 -0.10387266
     29     30     31     32     33     34     35
-0.16737929 -0.16331390 -0.04136133 -0.11590480 -0.04355648 -0.17015580 -0.07004084
     36     37
-0.15631500  0.06547687

$avg.silwidth
[1] -0.11629

$g2
NULL

$g3
NULL

$pearsongamma
[1] 0.05848364

$dunn
[1] 0.01900504

$dunn2
[1] 0.747704

$entropy
[1] 3.554273

$wb.ratio
[1] 0.8895936

$ch
[1] 4.747226

$cwidegap
 [1] 8.006144 7.812127 7.043537 7.908605 6.913267 7.476343 6.136562 6.010698 6.308852
[10] 6.101178 9.874780 5.913957 6.568892 6.838734 6.021215 8.835472 7.249955 8.324075
[19] 6.789731 6.757830 6.868184 6.593165 6.998036 6.929925 6.606172 6.185192 7.493594
[28] 6.783535 7.564265 8.371282 8.593945 6.513373 6.476367 7.369976 6.082538 6.514183
[37] 6.070794

$widestgap
[1] 9.87478

$sindex
[1] 0.5152458

$corrected.rand
NULL

$vi
NULL

```




BIBLIOGRAPHY:

- [1] Acín, A., Chen, J. L., Gisin, N., Kaszlikowski, D., Kwek, L. C., Oh, C. H., & Żukowski, M. (2004). Coincidence Bell Inequality for Three Three-Dimensional Systems. *Physical Review Letters*, 92(25). <https://doi.org/10.1103/PhysRevLett.92.250404>
- [2] Acín, A., Durt, T., Gisin, N., & Latorre, J. I. (2002). Quantum nonlocality in two three-level systems. *Physical Review A*, 65(5). <https://doi.org/10.1103/PhysRevA.65.052325>
- [3] Adcock, J., Allen, E., Day, M., Frick, S., Hinchliff, J., Johnson, M., ... Stanisc, S. (2015). Advances in quantum machine learning. *ArXiv:1512.02900 [Quant-Ph]*. Retrieved from <http://arxiv.org/abs/1512.02900>
- [4] Aerts, D., Aerts, S., Broekaert, J., & Gabora, L. (2000). The Violation of Bell Inequalities in the Macroworld. *ArXiv:Quant-Ph/0007044*. Retrieved from <http://arxiv.org/abs/quant-ph/0007044>
- [5] Alsina, D., Cervera, A., Goyeneche, D., Latorre, J. I., & Życzkowski, K. (2016). Operational approach to Bell inequalities: Application to qutrits. *Physical Review A*, 94(3). <https://doi.org/10.1103/PhysRevA.94.032102>
- [6] Alsina, D., & Latorre, J. I. (2016). Experimental test of Mermin inequalities on a five-qubit quantum computer. *Physical Review A*, 94(1). <https://doi.org/10.1103/PhysRevA.94.012314>
- [7] Anonymous. (2015). Bell's theorem. Retrieved from Quantiki website: <https://www.quantiki.org/wiki/bells-theorem>
- [8] Ardehali, M. (1992). Bell inequalities with a magnitude of violation that grows exponentially with the number of particles. *Physical Review A*, 46(9), 5375–5378. <https://doi.org/10.1103/PhysRevA.46.5375>
- [9] Arunachalam, S., & de Wolf, R. (2017). A Survey of Quantum Learning Theory. *ArXiv:1701.06806 [Quant-Ph]*. Retrieved from <http://arxiv.org/abs/1701.06806>
- [10] Baaquie, B. E. (2013). *The theoretical foundations of quantum mechanics*. New York, NY: Springer.
- [11] Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika*, 1(3), 195–200. <https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195>
- [12] Bell, J. S. (1981). Bertlmann's Socks and The Nature Of Reality. *Le Journal de Physique Colloques*, 42(C2), C2-41-C2-62. <https://doi.org/10.1051/jphyscol:1981202>
- [13] Bell, J. S. (1987). *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge. New York: Cambridge University Press.

- [14] Brunner, N., Cavalcanti, D., Pironio, S., Scarani, V., & Wehner, S. (2014). Bell nonlocality. *Reviews of Modern Physics*, 86(2), 419–478. <https://doi.org/10.1103/RevModPhys.86.419>
- [15] Buhrman, H., & Massar, S. (2005). Causality and Tsirelson's bounds. *Physical Review A*, 72(5), 052103. <https://doi.org/10.1103/PhysRevA.72.052103>
- [16] Cabello, A. (2008). Experimentally Testable State-Independent Quantum Contextuality. *Physical Review Letters*, 101(21). <https://doi.org/10.1103/PhysRevLett.101.210401>
- [17] Chen, J.-L., Wu, C., Kwek, L. C., Oh, C. H., & Ge, M.-L. (2006). Violating Bell inequalities maximally for two d -dimensional systems. *Physical Review A*, 74(3). <https://doi.org/10.1103/PhysRevA.74.032106>
- [18] Chen, K., Albeverio, S., & Fei, S.-M. (2006). *Tight Bell Inequalities for Many Qubits*. <http://webdoc.sub.gwdg.de/ebook/serien/e/sfb611/272.pdf>
- [19] Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., & Wossnig, L. (2018). Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 474(2209), 20170551. <https://doi.org/10.1098/rspa.2017.0551>
- [20] Clauser, J. F., Horne, M. A., Shimony, A., & Holt, R. A. (1969). Proposed Experiment to Test Local Hidden-Variable Theories. *Physical Review Letters*, 23(15), 880–884. <https://doi.org/10.1103/PhysRevLett.23.880>
- [21] Collins, D., Gisin, N. (2004). A relevant two qubit Bell inequality inequivalent to the CHSH inequality. *Journal of Physics A: Mathematical and General*, 37(5), 1775–1787. <https://doi.org/10.1088/0305-4470/37/5/021>
- [22] Collins, D., Gisin, N., Linden, N., Massar, S., & Popescu, S. (2002). Bell Inequalities for Arbitrarily High-Dimensional Systems. *Physical Review Letters*, 88(4), 040404. <https://doi.org/10.1103/PhysRevLett.88.040404>
- [23] Collins, D., Gisin, N., Popescu, S., Roberts, D., & Scarani, V. (2002). Bell-Type Inequalities to Detect True n -Body Nonseparability. *Physical Review Letters*, 88(17), 170405. <https://doi.org/10.1103/PhysRevLett.88.170405>
- [24] Coppersmith, D., & Winograd, S. (1990). Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3), 251–280. [https://doi.org/10.1016/S0747-7171\(08\)80013-2](https://doi.org/10.1016/S0747-7171(08)80013-2)

- [25] Das, A., Datta, C., & Agrawal, P. (2017). New Bell inequalities for three-qubit pure states. *Physics Letters A*, 381(47), 3928–3933. <https://doi.org/10.1016/j.physleta.2017.10.023>
- [26] Das, S., Aravinda, S., Srikanth, R., & Home, D. (2013). Unification of Bell, Leggett-Garg and Kochen-Specker inequalities: Hybrid spatio-temporal inequalities. *EPL (Europhysics Letters)*, 104(6), 60006. <https://doi.org/10.1209/0295-5075/104/60006>
- [27] Dawid Kopczyk. (2018). *Quantum Circuit Learning. Contribute to dawidkopczyk/qcl development by creating an account on GitHub*. Retrieved from <https://github.com/dawidkopczyk/qcl> (Original work published 2018)
- [28] Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz Machine. *Neural Computation*, 7(5), 889–904. <https://doi.org/10.1162/neco.1995.7.5.889>
- [29] Deng, D.-L., Zhou, Z.-S., & Chen, J.-L. (2009). Relevant multi-setting tight Bell inequalities for qubits and qutrits. *Annals of Physics*, 324(9), 1996–2003. <https://doi.org/10.1016/j.aop.2009.05.002>
- [30] d’Espagnat, B. (1979). The Quantum Theory and Reality. *Scientific American*, 241(5), 158–181. <https://doi.org/10.1038/scientificamerican1179-158>
- [31] Diósi, L. (2011). *A short course in quantum information theory: An approach from theoretical physics* (2nd ed). Heidelberg: Springer.
- [32] Durham, Ian (2006). A derivation of Bell’s inequalities in Wigner form from the fundamental assumption of statistical mechanics. *ArXiv:Quant-Ph/0612015*. <http://arxiv.org/abs/quant-ph/0612015>
- [33] Greenberger, D. M., Horne, M. A., Shimony, A., & Zeilinger, A. (1990). Bell’s theorem without inequalities. *American Journal of Physics*, 58(12), 1131–1143. <https://doi.org/10.1119/1.16243>
- [34] Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations* (3rd ed). Baltimore: Johns Hopkins University Press.
- [35] Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15), 150502. <https://doi.org/10.1103/PhysRevLett.103.150502>
- [36] Hey, T., Tansley, S., & Tolle, K. (2009). *The fourth paradigm: Data-intensive scientific discovery*. Redmond, Washington: Microsoft Research.
- [37] High-Level Steering Committee (2017). Quantum Technologies Flagship Final Report. Retrieved from: http://ec.europa.eu/newsroom/document.cfm?doc_id=46979
- [38] Hinton, G., Dayan, P., Frey, B., & Neal, R. (1995). The “wake-sleep” algorithm for unsupervised neural networks. *Science*, 268(5214), 1158–1161. <https://doi.org/10.1126/science.7761831>

- [39] Horn, D., & Axel, I. (2003). Novel clustering algorithm for microarray expression data in a truncated SVD space. *Bioinformatics*, 19(15), 2004–2004. <https://doi.org/10.1093/bioinformatics/btg441>
- [40] Horn, D., & Axel, I. (2003). Novel clustering algorithm for microarray expression data in a truncated SVD space. *Bioinformatics*, 19(9), 1110–1115. <https://doi.org/10.1093/bioinformatics/btg053>
- [41] Horn, David, & Gottlieb, A. (2001). Algorithm for Data Clustering in Pattern Recognition Problems Based on Quantum Mechanics. *Physical Review Letters*, 88(1).
<https://doi.org/10.1103/PhysRevLett.88.018702>
- [42] Horn, D., Gottlieb, A., & Axel I. (2003). *Quantum Clustering Page*. Retrieved from
<http://horn.tau.ac.il/QC.htm>
- [43] Jain M., & Chaturvedi, S. K. (2014). Quantum Computing Based Technique for Cancer Disease Detection System. *Journal of Computer Science & Systems Biology*, 7(3).
<https://doi.org/10.4172/jcsb.1000143>
- [44] Ji, S.-W., Lee, J., Lim, J., Nagata, K., & Lee, H.-W. (2008). Multisetting Bell inequality for qudits. *Physical Review A*, 78(5). <https://doi.org/10.1103/PhysRevA.78.052103>
- [45] Jordan, S. (2018). *Algebraic and Number Theoretic Algorithms*. Retrieved from
<http://quantumalgorithmzoo.org/>
- [46] Jungnitsch, B. (2012). *Criteria for genuine multiparticle quantum correlations Methods of entanglement detection and verification*. Publisher: Südwestdeutscher Verlag für Hochschulschriften.
- [47] Kabacoff, R. I. (2017). *Quick-R: Cluster Analysis*. Retrieved from
<https://www.statmethods.net/advstats/cluster.html>
- [48] Kak, S. (2013). Probability Constraints on the Classical/Quantum Divide. *NeuroQuantology*, 11(4).
<https://doi.org/10.14704/nq.2013.11.4.691>
- [49] Kafatos, M. (1989). *Bell's Theorem, Quantum Theory and Conceptions of the Universe*. Springer.
<https://doi.org/10.1007/978-94-017-0849-4>
- [50] Kerenidis, I., & Luongo, A. (2018). Quantum classification of the MNIST data set via Slow Feature Analysis. *ArXiv:1805.08837 [Quant-Ph]*. Retrieved from <http://arxiv.org/abs/1805.08837>
- [51] Khrennikov, A. (2003). Wigner's inequality for conditional probabilities and nonexistence of a realistic model for the two dimensional Hilbert space. *ArXiv:Quant-Ph/0308078*. <http://arxiv.org/abs/quant-ph/0308078>

- [52] Khrennikov, A. (2016). *Probability and randomness: Quantum versus classical*. Covent Garden, London: Imperial College Press.
- [53] Kochen, S., & Specker, E. (1967). The Problem of Hidden Variables in Quantum Mechanics. *Indiana University Mathematics Journal*, 17(1), 59–87. <https://doi.org/10.1512/iumj.1968.17.17004>
- [54] Lavín, J. L., & Anguita, J. (2018). Quantum DNA Sequencing: A Peek Into a Dystopic Future? *BioEssays*, 40(3), 1700248. <https://doi.org/10.1002/bies.201700248>
- [55] Leggett, A. J., & Garg, A. (1985). Quantum mechanics versus macroscopic realism: Is the flux there when nobody looks? *Physical Review Letters*, 54(9), 857–860. <https://doi.org/10.1103/PhysRevLett.54.857>
- [56] Liu, Y., & Zhang, S. (2017). Fast quantum algorithms for least squares regression and statistic leverage scores. *Theoretical Computer Science*, 657, 38–47. <https://doi.org/10.1016/j.tcs.2016.05.044>
- [57] Lloyd, S., Mohseni, M., & Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, 10(9), 631–633. <https://doi.org/10.1038/nphys3029>
- [58] Maccone, L. (2013). A simple proof of Bell's inequality. *American Journal of Physics*, 81(11), 854–859. <https://doi.org/10.1119/1.4823600>
- [59] Mackey, G. W. (2004). *Mathematical foundations of quantum mechanics*. Mineola, N.Y: Dover Publications.
- [60] Massen, S. (2019). *Bell inequalities*. Retrieved from <http://users.auth.gr/massen/KMIII/avisotntes-Bell-1.pdf>
- [61] Meila, M., & Shi, J. (2001). Learning Segmentation by Random Walks. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13* (pp. 873–879). Retrieved from <http://papers.nips.cc/paper/1830-learning-segmentation-by-random-walks.pdf>
- [62] Mermin, N. D. (1981). Bringing home the atomic world: Quantum mysteries for anybody. *American Journal of Physics*, 49(10), 940–943. <https://doi.org/10.1119/1.12594>
- [63] Mermin, N. D. (1990). Extreme quantum entanglement in a superposition of macroscopically distinct states. *Physical Review Letters*, 65(15), 1838–1840. <https://doi.org/10.1103/PhysRevLett.65.1838>
- [64] Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information* (10th anniversary ed). Cambridge ; New York: Cambridge University Press.

- [65] Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J., & Biswas, R. (2018). Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3), 030502. <https://doi.org/10.1088/2058-9565/aab859>
- [66] Peres, A. (1986). Existence of “free will” as a problem of physics. *Foundations of Physics*, 16(6), 573–584. <https://doi.org/10.1007/BF01886522>
- [67] Polozova, E., & Strauch, F. W. (2016). Higher-dimensional Bell inequalities with noisy qudits. *Physical Review A*, 93(3). <https://doi.org/10.1103/PhysRevA.93.032130>
- [68] Prakash, A. (2014). *Quantum Algorithms for Linear Algebra and Machine Learning*. eScholarship. Retrieved from <http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2014-211.pdf>
- [69] Preskill, J. (2018). *Physics 219 Course Information*. Retrieved from: <http://theory.caltech.edu/~preskill/ph229/>
- [70] Qian, X.-F., Little, B., Howell, J. C., & Eberly, J. H. (2015). Shifting the quantum-classical boundary: theory and experiment for statistically classical optical fields. *Optica*, 2(7), 611. <https://doi.org/10.1364/OPTICA.2.000611>
- [71] Rau, J. (2009). On quantum vs. classical probability. *Annals of Physics*, 324(12), 2622–2637. <https://doi.org/10.1016/j.aop.2009.09.013>
- [72] Rigetti (2016). *Quantum algorithms built using pyQuil. Contribute to rigetti/grove development by creating an account on GitHub*. Retrieved from <https://github.com/rigetti/grove>
- [73] Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge: Cambridge University Press.
- [74] Roy, S. M., & Singh, V. (1979). Experimental tests of quantum mechanics versus local hidden variable theories. *Journal of Physics A: Mathematical and General*, 11(8), L167–L171. <https://doi.org/10.1088/0305-4470/11/8/001>
- [75] Saunders, S. (2006). On the explanation for quantum statistics. *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, 37(1), 192–211. <https://doi.org/10.1016/j.shpsb.2005.11.002>
- [76] Schuld, M., Sinayskiy, I., & Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2). <https://doi.org/10.1103/PhysRevA.94.022342>
- [77] Scott, T., Therani, M., & Wang, X. (2017). Data Clustering with Quantum Mechanics. *Mathematics*, 5(1), 5. <https://doi.org/10.3390/math5010005>

[78] Sekar, K. K. (2017). *Awesome Quantum Machine Learning*.

<https://github.com/krishnakumarsekar/awesome-quantum-machine-learning>

[79] Servedio, R. A., & Gortler, S. J. (2004). Equivalences and Separations Between Quantum and Classical Learnability. *SIAM Journal on Computing*, 33(5), 1067–1092.

<https://doi.org/10.1137/S0097539704412910>

[80] Shun, P. H. (2015). *Testing the Quantum-Classical Boundary and Dimensionality of Quantum Systems* (Thesis Ph.D.). Retrieved from

<https://www.semanticscholar.org/paper/Testing-the-Quantum-Classical-Boundary-and-of-Shun/c50e6b721766c71d6eb244b12baa76da70727362>

[81] Sliorde. (2019). *some implementations of the unsupervised algorithm of quantum clustering* :

sliorde/quantum-clustering. Retrieved from <https://github.com/sliorde/quantum-clustering>

[82] Stapp, H. P. (1971). S -Matrix Interpretation of Quantum Theory. *Physical Review D*, 3(6), 1303–1320.

<https://doi.org/10.1103/PhysRevD.3.1303>

[83] Svetlichny, G. (1987). Distinguishing three-body from two-body nonseparability by a Bell-type inequality. *Physical Review D*, 35(10), 3066–3069. <https://doi.org/10.1103/PhysRevD.35.3066>

[84] Taylor, R.D. (2017). *A powerful, non-linear clustering method implemented in R*:

rdtaylor/quantumClustering . Retrieved from <https://github.com/rdtaylor/quantumClustering>

[85] Tsirelson, B. S. (1993). Some results and problems on quantum Bell-type inequalities. *Hadronic Journal Supplement* 8, 329-345. Retrieved from

<https://m.tau.ac.il/~tsirel/download/hadron.pdf>

[86] Weinstein, M., Meirer, F., Hume, A., Sciau, P., Shaked, G., Hofstetter, R., ... Horn, D. (2013).

Analyzing Big Data with Dynamic Quantum Clustering. *ArXiv:1310.2700 [Physics]*. Retrieved from

<http://arxiv.org/abs/1310.2700>

[87] Weinstein, M. (2010). Strange Bedfellows: Quantum Mechanics and Data Mining. *Nuclear Physics B - Proceedings Supplements*, 199(1), 74–84. <https://doi.org/10.1016/j.nuclphysbps.2010.02.009>

[88] Weinstein, M., & Horn, D. (2009). Dynamic quantum clustering: a method for visual exploration of structures in data. *Physical Review E*, 80(6), 066117. <https://doi.org/10.1103/PhysRevE.80.066117>

[89] Werner, R. F., & Wolf, M. M. (2001). All-multipartite Bell-correlation inequalities for two dichotomic observables per site. *Physical Review A*, 64(3), 032112. <https://doi.org/10.1103/PhysRevA.64.032112>

[90] Wheeler, J. A., & Zurek, W. H. (1983). Quantum theory and measurement. Princeton, N.J: Princeton University Press.

[91] Wiebe, N., Braun, D., & Lloyd, S. (2012). Quantum Algorithm for Data Fitting. *Physical Review Letters*, 109(5). <https://doi.org/10.1103/PhysRevLett.109.050505>

[92] Wigner, E. P. (1970). On Hidden Variables and Quantum Mechanical Probabilities. *American Journal of Physics*, 38(8), 1005–1009. <https://doi.org/10.1119/1.1976526>

[93] Wilde, M. M. (2013). *Quantum Information Theory*. Cambridge University Press.

<https://doi.org/10.1017/CBO9781139525343>

[94] Xie, Z., & Sato, I. (2017). A Quantum-Inspired Ensemble Method and Quantum-Inspired Forest Regressors. *ArXiv:1711.08117 [Cs]*. Retrieved from <http://arxiv.org/abs/1711.08117>

[95] Zohren, S., & Gill, R. D. (2008). Maximal Violation of the Collins-Gisin-Linden-Massar-Popescu Inequality for Infinite Dimensional States. *Physical Review Letters*, 100(12), 120406. <https://doi.org/10.1103/PhysRevLett.100.120406>

[96] Żukowski, M., & Brukner, Č. (2002). Bell's Theorem for General N -Qubit States. *Physical Review Letters*, 88(21), 210401. <https://doi.org/10.1103/PhysRevLett.88.210401>