ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΒΑΝΤΙΚΗ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Διπλωματική Εργασία

του

Δημήτρη Χατζόπουλου

Θεσσαλονίκη, Μάιος 2023

ΚΒΑΝΤΙΚΗ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Χατζόπουλος Δημήτριος
Πτυχίο Μαθηματικών,Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης,2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ
ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Ρεφανίδης Ιωάννης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

Ρεφανίδης Ιωάννης	Σακελλαρίου Ηλίας	Χρήστου Βαρσακέλης Δημήτριος
……	……..	……...

Χατζόπουλος Δημήτριος
……..

Περίληψη

Την τελευταία δεκαετία παρατηρείται μια τεράστια άνθιση του τομέα της Κλασικής Μηχανικής Μάθησης, λόγω της μεγάλης επέκτασης του διαδικτύου σε όλα τα μήκη και πλάτη του κόσμου και της συνδεσιμότητας περισσότερων συσκευών (smartphones, tablets πέρα από τα κλασικά λάπτοπ και σταθερούς υπολογιστές). Αυτό σημαίνει πως έχουμε πλέον να διαχειριστούμε μεγάλο όγκο δεδομένων (big data). Υπολογίζεται ότι μέχρι το τέλος του 2021 είχαν παραχθεί 74 zettabytes δεδομένων, με το zettabyte να αντιστοιχεί σε 1 τρισεκατομμύριο Gigabytes και αυτά τα μεγέθη αναμένεται να εκτοξευθούν στο σχετικά κοντινό μέλλον στην εποχή του Διαδικτύου των Πραγμάτων, με τις εκτιμήσεις για το 2023 να κάνουν λόγο για 130 zettabytes [1]. Τα πολλά διαθέσιμα δεδομένα είναι ζωτικής σημασίας για ένα μοντέλο μηχανικής μάθησης, μιας και εκπαιδεύεται καλύτερα όταν έχουμε περισσότερα δεδομένα και με μεγάλη διαφορετικότητα (diversity). Η Μηχανική Μάθηση λοιπόν είναι εκείνο το κομμάτι της Τεχνητής Νοημοσύνης που χρησιμοποιώντας εργαλεία προερχόμενα κυρίως από την Στατιστική, ανακαλύπτει μοτίβα (pattern recognition) χωρίς να έχει προγραμματιστεί επακριβώς, κι έχει φτάσει σε εξαιρετικά επίπεδα αναγνωρίζοντας εικόνες ή ήχους (image classification, voice/speech recognition) σχεδόν όπως και ο άνθρωπος. Από την άλλη το Quantum Computing πραγματοποιεί επεξεργασία πληροφορίας σε συσκευές που εκμεταλλεύονται νόμους της Κβαντικής Μηχανικής, όπως η υπέρθεση (superposition) και η διεμπλοκή (entaglement). Το ζήτημα που τίθεται και θα πραγματευτούμε στην παρούσα εργασία είναι το κατά πόσο η σύνθεση των 2 αυτών αντικειμένων (Quantum Machine Learning) μπορεί να κάνει τη διαφορά στην έρευνα της τεχνητής νοημοσύνης και της κβαντικής υπολογιστικής, με βάση πάντα τα όποια μέχρι στιγμής δεδομένα και αποτελέσματα στην έρευνα, και κατά πόσο τελικά θα μπορούσε η κβαντική μάθηση να μας φέρει πιο κοντά σε αυτό που ονομάζουν οι ερευνητές Γενική Νοημοσύνη (AGI).

Λέξεις-Κλειδιά:Κβαντική Υπολογιστική, Κβαντική Μηχανική Μάθηση, Κβαντικοί Αλγόριθμοι, Κβαντική Πληροφορία

Abstract

In the last decade, there has been a huge boom in the field of Classical Machine Learning, due to the great expansion of the internet to all the lengths and breadths of the world and the connectivity of more devices (smartphones, tablets in addition to classic laptops and desktop computers) which means that we now have a large amount of data to manage (big data). It is estimated that by the end of 2021, 74 zettabytes of data had been produced, with a zettabyte corresponding to 1 trillion Gigabytes, and these figures are expected to skyrocket in the relatively near future in the Internet-of-Things era. The many data available is vital for a machine learning model, as it trains better when we have more data and with diversity. So Machine Learning is that part of Artificial Intelligence that, using tools mainly derived from Statistics, finds patterns (pattern recognition) without being precisely programmed, and has reached exceptional levels recognizing images or sounds (image classification, voice/speech recognition) almost like humans. On the other hand, Quantum Computing performs information processing on devices that exploit laws of Quantum Mechanics, such as superposition, entanglement. The issue that arises and will be dealt with in this paper is the whether the composition of these 2 objects (Quantum Machine Learning) can makes a difference in artificial intelligence and quantum computing research and may bring us closer to what researchers call General Intelligence (AGI).

Keywords:Quantum Computing, Quantum Machine Learning, Quantum Algorithms, Quantum Information.

Πρόλογος-Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κύριο Ρεφανίδη Ιωάννη για την πολύτιμη και συνεχή καθοδήγηση και υποστήριξή του καθ’ όλη τη διάρκεια της διπλωματικής μου εργασίας. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για την καθολική στήριξή τους σε όλη τη διάρκεια των σπουδών μου, από το προπτυχιακό στο Μαθηματικό του ΑΠΘ μέχρι και τώρα στο Μεταπτυχιακό της Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας.

Περιεχόμενα σχημάτων|εικόνων

Περιεχόμενα αποσπασμάτων και ερμηνείας κώδικα

1 Εισαγωγή

Υπάρχουν αρκετοί λόγοι για να ασχοληθεί κανείς με τους κβαντικούς υπολογιστές, εδώ θα παραθέσουμε τους πιο βασικούς.

Από τη δεκαετία του 1960 μέχρι και σήμερα, η ισχύς των υπολογιστών έχει αυξηθεί εκθετικά, επιτρέποντας μας να έχουμε μικρότερους και καλύτερους υπολογιστές την ίδια στιγμή. Βάσει του νόμου του Moore [2], η υπολογιστική ισχύς διπλασιάζεται κάθε δύο χρόνια, ή ακριβέστερα: ”ο αριθμός των τρανζίστορ ενός πυκνού ολοκληρωμένου κυκλώματος διπλασιάζεται κάθε δύο χρόνια”, όπως ακριβώς διατυπώθηκε από τον ίδιο το Moore το 1965, με κάποιους να ρίχνουν το χρονικό διάστημα στους 18 μήνες. Όμως αυτή η διαδικασία είναι πλέον κοντά στο να φτάσει τα φυσικά της όρια. Διάφορα μέρη ενός υπολογιστή προσεγγίζουν πλέον το μέγεθος ενός ατόμου και αυτό αποτελεί πρόβλημα, διότι σε υποατομικό επίπεδο παύει να ισχύει πλέον το αιτιοκρατικό μοντέλο που έχουμε στους κλασικούς υπολογιστές, για κάθε πράξη που συντελείται στη CPU δεν μπορούμε να είμαστε σίγουροι για το ποιο θα είναι το αποτέλεσμα. Για παράδειγμα, λόγω της ιδιότητας του quantum tunelling δε μπορούμε να γνωρίζουμε πού ακριβώς βρίσκεται ένα ηλεκτρόνιο, πχ μπορεί να ξεπηδήσει κάποιο ηλεκτρόνιο από έναν αγωγό στον άλλο.

Ένα ακόμα στοιχείο είναι μια εργασία του διάσημου θεωρητικού φυσικού, βραβευμένου με βραβείο Nobel, Richard Feynman, ότι η φύση δεν λειτουργεί με τους κανόνες της κλασικής μηχανικής και ότι αν θέλουμε να προσομοιώσουμε το τι πραγματικά συμβαίνει στη φύση πρέπει να το κάνουμε κβαντομηχανικά, κάτι που πιθανώς ήταν η πρώτη σκέψη για τη δημιουργία ενός κβαντικού υπολογιστή [3].

Βεβαίως το μοντέλο του κβαντικού υπολογιστή δεν είναι το μοναδικό που προτείνεται ως λύση τώρα που ο νόμος του Moore φτάνει σε κάποιο τέλμα. Διάφορες μορφές διαφορετικού hardware έχουν προταθεί από τους ερευνητές, όπως είναι η νευρομορφική υπολογιστική (neuromorphic computing ή neuromorphic engineering), που στην ουσία πρόκειται για κυκλώματα που βασίζονται στον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου (βεβαίως μιλάμε για τσιπ που συνδέονται με μικροηλεκτρόδια και είναι της τάξεως μερικών εκατοντάδων “συνάψεων”, ενώ ο εγκέφαλος έχει τρισεκατομμύρια συνάψεις) που είναι και τομέας που αναμένεται να φέρει επανάσταση στον τομέα της Τεχνητής Νοημοσύνης, μιας και

προσφέρει ένα hardware ικανό να μιμείται τη βιολογική νοημοσύνη [4]. Επίσης υπάρχει το μοντέλο των DNA υπολογιστών (που αποθηκεύει πληροφορία σε πολύ μικρότερους χώρους απ' ό,τι ένας συμβατικός υπολογιστής, ακριβώς όπως το ανθρώπινο DNA), των 3D-υβριδικών υπολογιστών κλπ που δε θα μας απασχολήσουν στην παρούσα εργασία περισσότερο.

Αρχικά, θα πρέπει να παρουσιάσουμε μερικά στοιχεία από την Κβαντική Μηχανική (Quantum Mechanics) και τον Κβαντικό Υπολογισμό (Quantum Computation) που θα μας βοηθήσουν στη συνέχεια να κατανοήσουμε το γενικότερο πλαίσιο της Κβαντικής Μηχανικής Μάθησης.

2 Βασικές Έννοιες Κβαντομηχανικής

2.1 Qubit και Κβαντική Υπέρθεση

Στους κλασικούς Υπολογιστές έχουμε ως μονάδα μέτρησης της πληροφορίας το bit, που ορίζεται ως η ελάχιστη μορφή πληροφορίας που μπορούμε να έχουμε. "**Το 1 bit είναι η ποσότητα πληροφορίας που μπορούμε να πάρουμε όταν κάνουμε ερώτηση και η απάντηση είναι της μορφής ναι-όχι**." Ξέρουμε πως ένα κλασικό bit μπορεί να βρίσκεται σε 2 καταστάσεις, 0 ή 1. Αντίθετα, το qubit μπορεί να βρίσκεται είτε στις θέσεις 0 και 1 είτε στην υπέρθεση (superposition) αυτών, που σημαίνει πως κατά κάποιο τρόπο μπορεί να βρίσκεται και στις 2 καταστάσεις ταυτόχρονα, ή πιο σωστά σε ένα γραμμικό συνδυασμό αυτών των καταστάσεων. Κι επίσης, αν το qubit (ή γενικότερα το όποιο κβαντικό σύστημα) δεν επηρεαστεί από το εξωτερικό περιβάλλον, για παράδειγμα αν δεν το παρατηρήσουμε, θα παραμείνει στην κατάσταση υπέρθεσης [10,31].

Ένα ακόμα στοιχείο από την κβαντική φυσική που θα χρειαστούμε είναι η φύση των σωματιδίων: τα σωματίδια παρουσιάζουν ιδιότητες που έχουν τα κύματα, άρα ένα σωματίδιο μπορεί να κινείται σε 2 διαφορετικά μονοπάτια ταυτόχρονα. Αν έχω 2 κύματα που κινούνται σε 2 μονοπάτια, μπορεί αυτά τα 2 κύματα να αλληλεπιδράσουν με ένα φαινόμενο που λέγεται συμβολή (interference), δηλαδή είτε θα ενισχυθούν σε ένα ισχυρότερο κύμα είτε θα αλληλοκαταστραφούν. Κι αυτό είναι ένα φαινόμενο που προσπαθούμε να το κάνουμε να λειτουργήσει υπέρ μας στην προσπάθεια κατασκευής κβαντικών συστημάτων. Γενικότερα δεν είναι εύκολο να κατασκευάσουμε κβαντικούς αλγορίθμους, γι αυτό χρησιμοποιούμε αυτό ακριβώς το φαινόμενο της συμβολής που κατά κάποιο τρόπο οι πολλές καταστάσεις, οι πολλές

“λάθος” απαντήσεις αλληλοακυρώνονται και μένει μόνο μία κατάσταση, η “σωστή” απάντηση [31].

Όταν κάνουμε μέτρηση ή παρατήρηση (measurement) ενός qubit σε υπέρθεση, τότε συμβαίνει αυτό που ονομάζεται κατάρρευση (collapse) και παίρνουμε ως “απάντηση” μόνο μία από τις καταστάσεις του qubit. Γενικότερα, το qubit μπορεί να γραφτεί ως εξής:

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle=\alpha*\begin{bmatrix}1\\0\end{bmatrix}+\beta*\begin{bmatrix}0\\1\end{bmatrix}$$

όπου προφανώς τα $|0\rangle=\begin{bmatrix}1\\0\end{bmatrix}$και $|1\rangle=\begin{bmatrix}0\\1\end{bmatrix}$ είναι τα διανύσματα βάσης του διανυσματικού χώρου και $|\psi\rangle$ είναι το qubit, δηλαδή ένα κβαντικό σύστημα 2 καταστάσεων, γραμμένο με βάση το συμβολισμό Dirac. Τα α και β είναι μιγαδικοί αριθμοί για τους οποίους ισχύει η σχέση:
$|\alpha|^2+|\beta|^2=1$ και ονομάζονται αλλιώς και **πλάτη πιθανότητας** (probability amplitudes) [10,31,33].

Στην Κβαντική Μηχανική, όπως ήδη είδαμε, μία κβαντική κατάσταση (quantum state) $|\psi\rangle$ είναι ένα διάνυσμα σε χώρο Hilbert (μιγαδικός διανυσματικός χώρος με εσωτερικό γινόμενο ο οποίος είναι πλήρης). Για να τα πάρουμε από την αρχή, θα ορίσουμε μερικές έννοιες:

1) ket: είναι ένα κλασικό διάνυσμα σε έναν τυχαίο μιγαδικό διανυσματικό χώρο V (στην παρούσα εργασία θα θεωρούμε ως τέτοιο χώρο τον $\mathbf{C}^2$) και συμβολίζεται ως εξής:

$$|a\rangle=\begin{bmatrix}a_0\\a_1\end{bmatrix}$$

όπου τα α_0,α_1 είναι οι συντεταγμένες του διανύσματος $|a\rangle$ στη βάση $(|0\rangle,|1\rangle)$ του C^2, όπου $|0\rangle=(1,0)$ και $|1\rangle=(0,1)$.

2**) bra**: είναι στην ουσία μια γραμμική συνάρτηση $f:V\rightarrow C$ που απεικονίζει κάθε διάνυσμα του χώρου V σε κάποιο μιγαδικό αριθμό και γράφεται ως εξής:

$$\langle b|=|b\rangle^{+}=\begin{bmatrix}b_0\\b_1\end{bmatrix}^{+}=(b_o^{*},b_1^{*})$$

3) **bra-ket**: το εσωτερικό γινόμενο των $\langle b|$ και $|a\rangle$:

$\langle b|a\rangle=a^0b^{0*}+a^1b^{1*}=\langle a|b\rangle^*$ το οποίο είναι μιγαδικός αριθμός.

4) **ket-bra**:|a><b|= $\begin{bmatrix} a^0 b^0 & a^0 b^1 \\ a^1 b^0 & a^1 b^1 \end{bmatrix}$

το οποίο είναι το εξωτερικό γινόμενο των |α> και <b|.

Η σφαίρα του Bloch είναι ένα εργαλείο όπου μπορούμε να απεικονίσουμε γεωμετρικά ένα qubit στις 3 διαστάσεις. Δεν μπορούμε να αναπαραστήσουμε παραπάνω από 1 qubit, διότι για πχ 2 qubits πρέπει να έχουμε χώρο 4 διαστάσεων, που δεν έχουμε εποπτεία. Λέμε ότι όλες οι γνήσιες καταστάσεις, τα pure states απεικονίζονται πάνω στη σφαίρα του Bloch, η οποία έχει ακτίνα r=1.

Και αυτό διότι κάθε qubit μπορεί γενικώς να γραφεί όπως είδαμε παραπάνω ως υπέρθεση των βασικών καταστάσεων |0> και |1>:

$$|\psi>=\alpha|0>+\beta|1>$$

με τα α και β να είναι η πιθανότητα να βρεθεί το qubit στις θέσεις 0 ή 1. Και τέλος ισχύει και ο κανόνας του Born:

$$|\alpha|^2+|\beta|^2=1.$$

Πέρα από την παραπάνω μορφή, μπορούμε να πούμε ότι ένα qubit γράφεται και με τον παρακάτω τρόπο:

$$|\Psi>=e^{i\gamma}(\cos(\theta/2)|0>+e^{i\varphi}\sin(\theta/2)|1>)$$

όπου ο πρώτος παράγοντας του γινομένου($e^{i\gamma}$) μπορεί να παραληφθεί, οπότε γράφουμε

$$|\Psi>=\cos(\theta/2)|0>+e^{i\varphi}\sin(\theta/2)|1>$$

με τη γωνία φ να ανήκει στο [0,2π] και τη θ στο [0,π].

Ακόμη, πάνω στη Σφαίρα του Bloch μπορούν να αναπαρασταθούν και πύλες (quantum gates), που είναι περιστροφές πάνω στη σφαίρα, με κάποια γωνία ως προς κάποιον άξονα κάθε φορά.

Σημείωση: Διακρίνουμε τα qubits και γενικά τα κβαντικά συστήματα σε 2 βασικές κατηγορίες, τα καθαρά (pure) και τα μικτά (mixed). Αν δεν είναι καθαρό το σύστημα, που σημαίνει πως δεν είναι απομονωμένο, τότε μπορεί να υποστεί αποσυγκρότηση (decoherence), δηλαδή απώλεια της κατάστασης υπέρθεσης.

Λέμε πως ένα pure state γράφεται απλώς ως ένα διάνυσμα ket:|ψ>. Ενώ ένα mixed state είναι σύνολο από πολλά διανύσματα ket,$|\psi_1>$,$|\psi_2>$,$|\psi_3>$.., όπου κάθε ένα από αυτά έχει μια μη

μηδενική πιθανότητα να συμβεί. Στην ουσία πρόκειται για μια συλλογή από καθαρές καταστάσεις που έχουν κάποια πιθανότητα να συμβούν. Πώς όμως αναπαριστούμε τα mixed states; Κι εδώ θα χρειαστούμε τη βοήθεια των πινάκων πυκνότητας (density matrices):

Ένα pure state γράφεται ως πίνακας με την ακόλουθη μορφή:

$$\rho=|\psi><\psi|$$

ξέρουμε πως από την κατάσταση υπέρθεσης

$$|\psi>=\alpha|0>+\beta|1>$$

οπότε με βάση αυτό,γράφουμε:

$$\rho=\begin{bmatrix} a \\ b \end{bmatrix} * \begin{bmatrix} \dot{a} & \dot{b} \end{bmatrix} = \begin{bmatrix} a\dot{a} & a\dot{b} \\ b\dot{a} & b\dot{b} \end{bmatrix}$$

κι επειδή είναι pure state πρέπει να ισχύει βάσει του Κανόνα του Born όπως είδαμε πιο πάνω, ότι:

$$|\alpha|^2+|\beta|^2=1$$

όπου εδώ,αυτά τα 2 στοιχεία είναι πάνω στην κύρια διαγώνιο και όπως ξέρουμε από τη γραμμική άλγεβρα, αυτό το άθροισμα ονομάζεται ίχνος (trace) του πίνακα.

Στις μικτές καταστάσεις, ο πίνακας γράφεται ως το άθροισμα των επιμέρους πιθανοτήτων κάθε κατάστασης επί την κατάσταση αυτή:

$$\rho=\sum Pk*|\psi><\psi>$$

2.2 Κβαντική Διεμπλοκή (Entanglement):

Ένα πάρα πολύ σημαντικό κομμάτι της κβαντικής θεωρίας είναι αυτό της κβαντικής διεμπλοκής. Είναι ένα σημείο της κβαντομηχανικής που ακόμη και σπουδαίοι επιστήμονες όπως πχ ο Einstein αρνούνταν να δεχτούν ότι ισχύει, λόγω του ότι είναι τόσο αντίθετο προς την κοινή λογική. Η διεμπλοκή είναι η ιδιότητα που επιτρέπει σε 2 σωματίδια να αλληλεπιδρούν άμεσα, ακόμα κι αν απέχουν τεράστια απόσταση το ένα από το άλλο. Ακόμη κι αν το ένα σωματίδιο είναι στη μία άκρη του σύμπαντος και το άλλο στην άλλη, όταν είναι διεμπλεκόμενα (entagled), μπορούν να “επικοινωνούν” με ταχύτητα **μεγαλύτερη από την ταχύτητα του φωτός** [5]. Τα σωματίδια δεν επικοινωνούν ακριβώς, αλλά υπάρχει μια

στατιστική συσχέτιση μεταξύ των αποτελεσμάτων όταν κάνουμε μέτρηση σε κάθε σωματίδιο, η οποία συσχέτιση είναι δύσκολο να ερμηνευθεί με κάποια θεωρία της κλασικής φυσικής. Για παράδειγμα λέμε πως αν δύο σωματίδια έχουν διεμπλακεί, τότε, όταν θα μετρηθεί η ταχύτητα του ενός μπορούμε άμεσα να είμαστε σίγουροι για το ποια είναι η ταχύτητα του άλλου, όσο κι αν έχουν απομακρυνθεί, πριν μετρηθεί το δεύτερο σωματίδιο. Ο λόγος που λέμε πως επικοινωνούν αστραπιαία είναι ότι δεν αποθηκεύουν πληροφορία τοπικά. Και αποκτούν κάποια καλώς ορισμένη κατάσταση αμέσως μόλις μετρηθούν. Εξαιτίας αυτής της ιδιότητας των σωματιδίων, τα σωματίδια δεν μπορούν να χρησιμοποιηθούν για να μεταφερθεί πληροφορία γρηγορότερα από την ταχύτητα του φωτός, μιας και θα πρέπει να μετρηθεί η κατάσταση τους πρώτα και μετά να εξαχθεί κάποιο αποτέλεσμα.

Πιστεύεται ότι η διεμπλοκή θα μας βοηθήσει να πετύχουμε αυτό που λέγεται κβαντική τηλεμεταφορά (quantum teleportation) [31,32]. Ο όρος φαντάζει τελείως εξωπραγματικός, παρ' όλα αυτά αυτό που εννοούμε είναι η μεταφορά μιας κβαντικής κατάστασης από ένα σημείο σε ένα άλλο χωρίς απώλειες. Εννοούμε την μεταφορά της πληροφορίας και όχι του αντικειμένου. Το αρχικό αντικείμενο καταστρέφεται και στο νέο σημείο δημιουργείται ένα νέο αντικείμενο, σύμφωνα πάντα με την πληροφορία που "τηλεμεταφέρθηκε". Επίσης η διεμπλοκή θα είναι από τα θεμελιώδη στοιχεία του κβαντικού διαδικτύου, της ασφαλούς επικοινωνίας μέσω του διαδικτύου (κβαντική κρυπτογράφηση) και πιθανώς θα έχει και άλλες εφαρμογές στο μέλλον.

2.3 Τα 4 Βασικά Αξιώματα Της Κβαντικής Μηχανικής

1ο αξίωμα: Για κάθε κλειστό κβαντικό σύστημα, πρέπει να υπάρχει τρόπος να περιγράψουμε κάθε σωματίδιο που βρίσκεται μέσα του. Για κάθε απομονωμένο φυσικό σύστημα, υπάρχει ένας χώρος Hilbert ο οποίος σχετίζεται άμεσα με το κλειστό σύστημα και ονομάζεται "χώρος καταστάσεων". Το σύστημα, για κάποια δεδομένη χρονική στιγμή, περιγράφεται πλήρως από το διάνυσμα κατάστασης, που είναι ένα μοναδιαίο διάνυσμα του χώρου Hilbert.

2ο αξίωμα: Η χρονική εξέλιξη ενός κλειστού κβαντικού συστήματος περιγράφεται από έναν ορθομοναδιαίο μετασχηματισμό. Κάτι που σημαίνει ότι αν έχω την κατάσταση $|\psi_1\rangle$ τη χρονική στιγμή t_1 και την κατάσταση $|\psi_2\rangle$ τη χρονική στιγμή t_2 , τότε αυτές οι 2 καταστάσεις συνδέονται μεταξύ τους με έναν ορθομοναδιαίο πίνακα U με την παρακάτω σχέση:

$$|\psi_2\rangle = U|\psi_1\rangle.$$

Μπορώ δηλαδή να περιγράψω τη συμπεριφορά ενός συστήματος χρησιμοποιώντας ορθομοναδιαίους πίνακες.

3ο αξίωμα: Οι κβαντικές μετρήσεις περιγράφονται από μια συλλογή $\{K_m\}$ τελεστών μέτρησης. Αυτοί οι τελεστές δρουν πάνω στο χώρο καταστάσεων του συστήματος που πρόκειται να μετρηθεί. Το m υποδηλώνει το πλήθος των αποτελεσμάτων που μπορεί να έχουμε.

4ο αξίωμα: Ο χώρος καταστάσεων ενός σύνθετου φυσικού συστήματος μπορεί να αναλυθεί σε ένα τανυστικό γινόμενο των επιμέρους χώρων Hilbert που απαρτίζουν το σύνθετο σύστημα.

2.4 Μαθηματικά της Κβαντικής Μηχανικής

2.4.1 Στοιχεία Γραμμικής Άλγεβρας

Εδώ να ορίσουμε μερικά στοιχεία από τη γραμμική άλγεβρα και συγκεκριμένα από τη θεωρία πινάκων που θα μας βοηθήσουν και στην συνέχεια όταν θα μελετήσουμε τις κβαντικές πύλες και τα κβαντικά κυκλώματα.

Θα χρειαστούμε τα παρακάτω είδη πινάκων:

1) **τετραγωνικός πίνακας**: ένας πίνακας λέγεται τετραγωνικός όταν το πλήθος των γραμμών του ισούται με το πλήθος των στηλών (της μορφής N*N). Στη δική μας περίπτωση, οι πίνακες είναι της μορφής 2^N*2^N, πάλι τετραγωνικοί, απλώς υποσύνολο του συνόλου των N*N πινάκων.

2) **Μοναδιαίος πίνακας**: όλα τα στοιχεία της κύριας διαγωνίου είναι ίσα με τη μονάδα και ΌΛΑ τα υπόλοιπα ίσα με το 0.

3) **Μηδενικός πίνακας**: όλα τα στοιχεία=0.

4) **Συμμετρικός πίνακας**: $A=A^T$, όπου με A^T συμβολίζεται ο ανάστροφος του A, αυτός που έχει ως στήλες τις γραμμές του A και ως γραμμές τις στήλες του (θα ακολουθήσει επεξηγηματικό παράδειγμα).

5)**Ερμητιανός πίνακας**: είναι ο πίνακας που είναι ίσος με τον ανάστροφο-συζυγή του. Συμβολίζεται με $A^+ =A$, όπου $A^+=(A^T)^*=(A^*)^T$.

6)**Κανονικός πίνακας**: είναι αυτός που αντιμετατίθεται με τον ερμητιανό του, δηλαδή:

$A^+A=AA^+$.

7)**Ορθομοναδιαίος πίνακας**: αυτός που η παραπάνω σχέση ισούται με το μοναδιαίο:

$A^+A=AA^+ =I$. Ένας πίνακας όπως θα δούμε και παρακάτω, μπορεί να είναι κβαντική πύλη μόνο αν είναι ορθομοναδιαίος. Οι ορθομοναδιαίοι και οι ερμητιανοί είναι υποσύνολο των κανονικών πινάκων.

Παράδειγμα:

Έστω ο

$$A=\begin{bmatrix} 1+2i & 2-5i \\ 0 & 7-3i \end{bmatrix}$$

τότε θα έχω:

$$A^T=\begin{bmatrix} 1+2i & 0 \\ 2-5i & 7-3i \end{bmatrix}$$

ο ανάστροφος.

$$A^*=\begin{bmatrix} 1-2i & 2+5i \\ 0 & 7+3i \end{bmatrix}$$

ο συζυγής, και

$$A^+=(A^*)^T=(A^T)^*=\begin{bmatrix} 1-2i & 0 \\ 2+5i & 7+3i \end{bmatrix}$$

ο ερμητιανός.

2.4.2 Τανυστές

Ένα άλλο κομμάτι από τα μαθηματικά που θα το χρειαστούμε στη συνέχεια είναι η έννοια του τανυστή (tensor). Οι τανυστές είναι γεωμετρικά αντικείμενα που θα μπορούσαμε να τα δούμε ως τη γενίκευση των διανυσμάτων σε μεγαλύτερες διαστάσεις. Ένας τανυστής μηδενικής τάξης είναι ένα βαθμωτό μέγεθος (scalar), δηλαδή ένας πραγματικός αριθμός. Ο τανυστής πρώτης τάξης είναι ένα διάνυσμα (vector), δεύτερης τάξης είναι ο πίνακας (array) και από τρίτη τάξη και πάνω είναι ο τανυστής της αντίστοιχης διάστασης. Στην κλασική μηχανική

μάθηση χρειαζόμαστε τους τανυστές όταν αναπαριστούμε δεδομένα, διότι αυτή η αναπαράσταση πρέπει να γίνει αριθμητικά. Άρα όσον αφορά την Επιστήμη των Υπολογιστών ο τανυστής είναι ένας χώρος στη μνήμη που μπορεί να αποθηκεύσει δεδομένα στις N διαστάσεις. Στην περίπτωση των καταχωρητών, έχουμε το εξής:

Όταν θέλουμε να δώσουμε μια μαθηματική αναπαράσταση για ολόκληρο τον κβαντικό καταχωρητή, γράφουμε:

$$|q\rangle=|q_{n-1}\ldots\ldots q_1q_0\rangle=|q_{n-1}\rangle\ldots..|q_0\rangle=$$
$$=|q_{n-1}\rangle\otimes....\otimes|q_0\rangle.$$

το οποίο είναι το τανυστικό γινόμενο όλων των καταστάσεων που είναι αποθηκευμένες στον καταχωρητή.

Το τανυστικό γινόμενο για 2 καταστάσεις γράφεται ως εξής:

$$\begin{bmatrix} a \\ b \end{bmatrix}\otimes\begin{bmatrix} c \\ d \end{bmatrix}=\begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

για τις 2 τυχαίες καταστάσεις $|\psi1\rangle=\begin{bmatrix} a \\ b \end{bmatrix}$ και $|\psi2\rangle=\begin{bmatrix} c \\ d \end{bmatrix}$.

2.5 Είδη κβαντικών υπολογιστών

Εδώ αξίζει να αναφέρουμε ότι υπάρχουν 5 διαφορετικά είδη κβαντικών υπολογιστικών συστημάτων, με πιο βασικό το κυκλωματικό μοντέλο (circuit model) που είναι και το πιο πρακτικό, μιας και όλες οι μεγάλες εταιρείες και τα πανεπιστήμια που έχουν κβαντικούς υπολογιστές, έχουν αυτό ακριβώς το μοντέλο. Τα άλλα μοντέλα κβαντικών υπολογιστών είναι το αδιαβατικό, η λειτουργία του οποίου βασίζεται στο Αδιαβατικό Θεώρημα της Κβαντικής Μηχανικής που δε θα αναλύσουμε στην παρούσα εργασία, το measurement-based cluster state, οι ολονομικοί (holonomic) και οι τοπολογικοί (topological) κβαντικοί υπολογιστές [9,10,31]. Όλοι είναι ισοδύναμοι μεταξύ τους, τουλάχιστον σε θεωρητικό επίπεδο, δηλαδή μπορούμε να λύσουμε ένα πρόβλημα σε κυκλωματικό κβαντικό υπολογιστή και μετά να το λύσουμε και με τοπολογικό κβαντικό υπολογιστή, ίσως αλλάζοντας κάπως τον αλγόριθμο που χρησιμοποιήσαμε, προσαρμόζοντας τον κάθε φορά στις ανάγκες του νέου μοντέλου. Σε πρακτικό επίπεδο όμως, όλα τα υπόλοιπα μοντέλα πλην του κυκλωματικού είναι σε τελείως πειραματικό στάδιο ακόμη, με λιγότερα από 10 qubits. Επομένως η έρευνα αυτή τη στιγμή

βρίσκεται σε τελείως θεωρητικό επίπεδο όσον αφορά τα άλλα μοντέλα και έχει κυρίως να κάνει με το πώς θα κατασκευάσουμε κβαντικούς αλγορίθμους που θα λειτουργούν σε αυτά τα μοντέλα πέραν του κυκλωματικού. Το γιατί δεν μπορούμε να έχουμε υπολογιστές με παραπάνω από 10 qubits είναι ένα μεγάλο ερώτημα και έχουμε κάποιες πιθανές απαντήσεις:

1ον ένα τεράστιο ζήτημα για τους κβαντικούς υπολογιστές είναι ότι είναι ευάλωτοι σε θόρυβο, μπορεί δηλαδή να μας δίνει λάθος απαντήσεις λόγω του ότι υπάρχει παρέμβαση από τον έξω χώρο (γι' αυτό και οι κβαντικοί υπολογιστές που διατίθενται σήμερα βρίσκονται σε μεγάλους χώρους όσο ένα δωμάτιο, που λειτουργεί ουσιαστικά ως ψυγείο με πάρα πολύ χαμηλές θερμοκρασίες) ή μπορεί ακόμα να μην μπορεί ο υπολογιστής να λειτουργήσει απρόσκοπτα για ικανοποιητικό χρονικό διάστημα λόγω του φαινομένου της αποσυγκρότησης (decoherence). Ας αναφέρουμε ένα γενικό παράδειγμα για την αποσυγκρότηση: έστω ένα νόμισμα που περιστρέφεται στον αέρα και άρα δεν έχουμε ακόμα εικόνα για το ποια θα είναι η τελική ένδειξη. Το νόμισμα αυτό δε θα περιστρέφεται για πάντα, οπότε αν θέλουμε να επηρεάσουμε το αποτέλεσμα εφαρμόζοντας μια πύλη, πρέπει να το "προλάβουμε" όσο ακόμα είναι στον αέρα και περιστρέφεται, ή με πιο μαθηματικό τρόπο γραφής, να είναι στην κατάσταση υπέρθεσης:

$$1/2*|0>+1/2*|1>.$$

Επομένως, σε πάρα πολύ μικρό χρονικό διάστημα, κάνουμε όσες πράξεις προλαβαίνουμε (εφαρμόζουμε πύλες) και όχι όσες θα χρειαζόμασταν και αυτό εν τέλει μας δίνει διαφορετικό αποτέλεσμα σε σχέση με αυτό που θα περιμέναμε.

Ένα άλλο θέμα που έχουμε είναι ότι το ένα qubit μπορεί να επηρεάσει το άλλο. Γενικά θέλουμε το σύστημα του κβαντικού καταχωρητή να είναι ένα ***κλειστό σύστημα,*** απομονωμένο από τον υπόλοιπο κόσμο, διότι οποιαδήποτε αμελητέα αλληλεπίδραση, θα αλλάξει το αποτέλεσμα μας. Δεν θα πρέπει να γίνεται καμία κατά λάθος μέτρηση πάνω στο σύστημα μας. Όταν παρατηρούμε, στην ουσία αυτό που κάνουμε είναι για παράδειγμα να στείλουμε κάποιο φωτόνιο πάνω στο σύστημα, που είναι μορφή ενέργειας, οπότε επηρεάζουμε την κατάσταση, το θέμα είναι να επηρεάζουμε μόνο εμείς και με τον τρόπο που θέλουμε κάθε φορά. Υπάρχει κι ένα άλλο θεμελιώδες πρόβλημα: στην Κβαντική Θεωρία υπάρχει επίσης η έννοια του κβαντικού κενού: υπάρχει μη μηδενική πιθανότητα, σε κατάσταση απολύτου κενού να υπάρχουν κάποια σωματίδια, να παραχθούν δηλαδή με κάποιο τρόπο, κάτι που αντιβαίνει

πλήρως την κοινή λογική [7]. Έτσι λοιπόν, μπορεί ακόμα και σε συνθήκες απόλυτης απομόνωσης του κβαντικού καταχωρητή μου να έχω αλλοίωση της κατάστασής του.

2.6 Κβαντικές πύλες

Περνάμε σε ένα πολύ σημαντικό κομμάτι της εργασίας, μιας και οι κβαντικές πύλες όπως και τα κβαντικά κυκλώματα που θα δούμε στη συνέχεια, είναι τα πιο βασικά εργαλεία πάνω στα οποία θα κάνουμε την κβαντική μάθηση. Η μηχανική μάθηση σε κβαντικό περιβάλλον χρησιμοποιεί κβαντικά κυκλώματα (parameterized quantum circuits ή αλλιώς variational models) με κάποια παραμετροποίηση $\Theta=(\theta_0,\ldots,\theta_n)$, με τις παραμέτρους να παίζουν το ρόλο των βαρών (weights) στα κλασικά νευρωνικά δίκτυα.

Στην περίπτωση των κλασικών υπολογιστών έχουμε τις λογικές πύλες (logic gates) και τους αγωγούς. Οι αγωγοί μεταφέρουν την πληροφορία από τη μία πύλη στην άλλη με χρήση της ηλεκτρικής ενέργειας. Οι πύλες (που μπορεί να είναι η OR, η AND, η NOT ή και πιο σύνθετες) παίρνουν αυτή την πληροφορία και την επεξεργάζονται για να δώσουν άλλο αποτέλεσμα τελικά, με βάση πάντα τον πίνακα αληθείας κάθε πύλης.

Στην περίπτωση όμως των κβαντικών υπολογιστών δεν έχουμε φυσικά συστήματα. Οι κβαντικές πύλες δηλαδή δεν είναι συστήματα που διαρρέονται από ηλεκτρική ενέργεια, αλλά στην πραγματικότητα είναι κάποιες δράσεις πάνω στα απομονωμένα qubits ή σε κάποιο σύνολο από qubits που ονομάζεται κβαντικός καταχωρητής. Οι δράσεις αυτές είναι κάποιοι τελεστές που εφαρμόζονται πάνω στα qubits και που απεικονίζονται με πίνακες συγκεκριμένων διαστάσεων [10,31,33].

Μια ακόμα διαφορά μεταξύ των 2 συστημάτων είναι ότι στην κβαντική περίπτωση, η πληροφορία δεν διέρχεται μέσα από τις κβαντικές πύλες όπως γίνεται στην κλασική περίπτωση αλλά είναι αποθηκευμένη εκεί.

Όπως έχουμε αναφέρει και προηγουμένως, τα qubits απεικονίζονται ως διανύσματα σε χώρο Hilbert. Οι κβαντικές πύλες λοιπόν είναι τελεστές του χώρου Hilbert, που επιδρούν πάνω στα qubits αλλάζοντας την κατάστασή τους. Αν θεωρήσουμε τα qubits ως διανύσματα στη σφαίρα του Bloch, τότε οι κβαντικές πύλες περιστρέφουν το διάνυσμα κατάστασης που απεικονίζει το qubit κατά μια γωνία, χωρίς όμως να αλλάζει το μήκος του που είναι σταθερά ίσο με 1.

Για να μπορεί ένας τελεστής σε χώρο Hilbert να είναι κβαντική πύλη, θα πρέπει να πληρούνται 2 βασικές προϋποθέσεις:

1) να διατηρεί το μήκος των διανυσμάτων κατάστασης

2) να τηρεί τη χρονική συμμετρία των κβαντικών συστημάτων

Οι τελεστές αυτοί που έχουν αυτές τις ιδιότητες ονομάζονται ορθομοναδιαίοι ή ερμητιανοί (hermitian operators) και περιγράφονται από ορθομοναδιαίους πίνακες όπως τους ορίσαμε παραπάνω.

Ως προς τη χρονική συμμετρία, αξίζει να αναφέρουμε το εξής: η συμπεριφορά ενός κβαντικού συστήματος δεν αλλάζει με τη φορά του χρόνου. Δηλαδή, αν έχω μια πύλη K και αλλάξει την κατάσταση ενός κβαντικού καταχωρητή από $|q_1>$ σε $|q_2>$, τότε εφαρμόζοντας πάλι την πύλη στο q_2 θα πρέπει να πάρω q_1.

$$K|q_1>=|q_2>$$

$$K|q_2>=|q_1>$$

Οι πύλες αυτές ονομάζονται αναστρέψιμες και είναι ιδιότητα όλων των κβαντικών πυλών. Όπως είπαμε, μια κβαντική πύλη αναπαρίσταται από έναν πίνακα ο οποίος πρέπει να έχει κάποιες ιδιότητες:

1) να είναι τετραγωνικός (της μορφής 2^n*2^n)

2) να έχει αντίστροφο

3) να μην αλλάζει το μήκος του διανύσματος πάνω στο οποίο επιδρά, παρά μόνο τη γωνία του.

Αυτοί είναι οι ορθομοναδιαίοι πίνακες όπως τους είχαμε αναλύσει προηγουμένως [31].

2.6.1 Κβαντικές πύλες που δρουν σε ένα qubit

Αυτές οι πύλες περιστρέφουν το διάνυσμα κατάστασης ενός qubit πάνω στη Σφαίρα Bloch, χωρίς να επηρεάζουν το μήκος του διανύσματος, απλώς μεταβάλλουν τις γωνίες φ και θ.

Υπάρχουν άπειρες τέτοιες περιστροφές που μπορούν να γίνουν, επομένως και άπειρες πύλες που μπορούν να δράσουν πάνω σε 1 qubit. Εδώ θα δούμε τις πιο βασικές.

Η πύλη NOT

Η πύλη NOT συμπεριφέρεται ακριβώς όπως και στην κλασική περίπτωση, δηλαδή μετατρέπει το $|0>$ σε $|1>$ και αντίστροφα.

ο πίνακας είναι ο παρακάτω: $NOT=\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$

και πίνακα αληθείας:

|q0>	|q1>
|0>	|1>
|1>	|0>

Εξίσου σημαντικές πύλες που θα αναλύσουμε είναι οι εξής :

1) η κβαντική πύλη αδρανείας I

2) η κβαντική πύλη μετατόπισης φάσης Φ

3) η κβαντική πύλη Hadamard H

Ως προς την πρώτη, είναι η πύλη που εκφράζεται από το μοναδιαίο πίνακα:

$$I=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

και είναι η πύλη που αφήνει ανέπαφη την κατάσταση στην οποία επιδρά.

Ο πίνακας αληθείας είναι ο παρακάτω:

$\|q_1\rangle$	$\|q_2\rangle$
|0>	|0>
|1>	|1>

ο πίνακας I προκύπτει και ως των πράξη βασικών καταστάσεων

$$I=|0><0|+|1><1|$$

Πύλη μετατόπισης φάσης:

είναι η πύλη που εκφράζεται από τον πίνακα:

$$\Phi=\begin{bmatrix}1 & 0\\0 & e^{i\varphi}\end{bmatrix}$$

με πίνακα δράσης τον:

|q0>	|q1>
|0>	|0>
|1>	$e^{i\varphi}$|1>
a|0>+b|1>	a|0>+$e^{i\varphi}$b|1>

Η πύλη Hadamard

η οποία έχει ως πίνακα:

$$H= \frac{1}{\sqrt{2}} * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

με πίνακα δράσης τον:

|q1>	|q2>
|0>	(|0>+|1>)/$\sqrt{2}$
|1>	(|0>-|1>)/$\sqrt{2}$

Η πύλη Hadamard είναι από τις πιο βασικές πύλες που θα μας απασχολήσουν και στη συνέχεια [10]. Μπορεί να μεταφέρει τα qubits από τις βασικές τους καταστάσεις |0> και |1> σε κατάσταση υπέρθεσης και το αντίστροφο, να μεταφέρει δηλαδή τα qubits από την υπέρθεση στις βασικές καταστάσεις. Είναι από τις πιο βασικές πράξεις που θα συναντήσουμε αργότερα στη μάθηση με variational models.

2.6.2 Πύλες Pauli

Από τις πιο βασικές ομάδες πυλών είναι οι πύλες Pauli. Οποιαδήποτε κβαντική πύλη μπορεί να γραφεί ως γραμμικός συνδυασμός των πυλών Pauli και της πύλης αδρανείας. Μπορούμε δηλαδή να εκτελέσουμε κάθε πιθανή περιστροφή πάνω στη σφαίρα Bloch. Λειτουργούν ομοίως με τις λογικές πύλες όπως οι AND, OR, NOT σε κλασικό υπολογιστή.
Συμβολίζονται με τα γράμματα X, Y, Z και οι αντίστοιχοι πίνακες είναι οι εξής:

$$X=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$$Y=\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$$

$$Z=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

με πίνακα δράσης τον:

	|q1>	|q0>
X	a|0>+b|1>	b|0>+a|1>
Y	a|0>+b|1>	-ib|0>+ia|1>
Z	a|0>+b|1>	a|0>-b|1>

2.6.3 Πύλες που δρουν σε 2 qubits

Εδώ θα αναφέρουμε πύλες που χρειάζονται 2 qubits για να δράσουν, η κβαντική πύλη ελεχγόμενου ΟΧΙ και η πύλη ελεγχόμενης μετατόπισης φάσης.

Η πύλη ελεγχόμενου ΟΧΙ (αναφέρεται με τον όρο CNOT) είναι μια κβαντική πύλη που δρα σε 2 qubits, το ένα ονομάζεται qubit ελέγχου και συμβολίζεται με c και το άλλο qubit στόχος και συμβολίζεται με t. Ορίζουμε τις καταστάσεις πριν από τη δράση της πύλης ως $|c_i>$ και $|t_i>$ και μετά τη δράση ως $|c_0>$ και $|t_0>$. Όταν το qubit ελέγχου είναι στην κατάσταση |1>, τότε αλλάζει η κατάσταση του qubit στόχου, ενώ αν το αρχικό qubit είναι στο |0>, το qubit στόχος δεν επηρεάζεται. Η κατάσταση του qubit ελέγχου είναι πάντα η ίδια, δηλαδή το $|c_0>$ είναι στην ουσία το ίδιο με το $|c_1>$. Ο πίνακας της CNOT είναι ο παρακάτω:

$$CNOT=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

ο πίνακας δράσης είναι ο εξής:

$\|c_i t_i\rangle$	$\|c_0 t_0\rangle$
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

Φαίνεται και από εδώ πως αν το πρώτο qubit (ελέγχου) είναι 0, δεν αλλάζει τίποτα, ενώ αν είναι 1, τότε επηρεάζεται το 2ο qubit (στόχος) [10,31].

Πύλη ελεγχόμενης μετατόπισης φάσης

Ομοίως με προηγουμένως, κι εδώ έχουμε 1 qubit ελέγχου c και 1 qubit στόχο t. Ακριβώς όπως πριν συμβολίζουμε με $|c_i\rangle$ και $|t_i\rangle$ τις καταστάσεις πριν από τη δράση της πύλης και $|c_0\rangle$, $|t_0\rangle$ τις καταστάσεις μετά τη δράση. Η πύλη CΦ όπως θα τη συμβολίζουμε εδώ, πολλαπλασιάζει την κατάσταση του qubit στόχου με τον παράγοντα $e^{i\varphi}$ μόνο όταν η κατάσταση του qubit ελέγχου και του qubit στόχου είναι και οι $|1\rangle$. Στις άλλες περιπτώσεις, δεν μεταβάλλεται απολύτως τίποτα. Ο πίνακας CΦ είναι ο κάτωθι:

$$\text{CΦ}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}$$

με πίνακα αληθείας τον:

$\|c_i t_i\rangle$	$\|c_0 t_0\rangle$
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|10\rangle$
$\|11\rangle$	$e^{i\varphi}\|11\rangle$

2.6.4 Πύλες που δρουν σε 3 qubits

Εδώ θα δούμε την τελευταία κατηγορία απο πύλες που μας ενδιαφέρει, η δράση πάνω σε 2 qubits. Συγκεκριμένα θα δούμε την πύλη διπλά ελεγχόμενου ΟΧΙ και την πύλη Fredkin.

Πύλη διπλά ελεγχόμενου ΟΧΙ

Αυτή η πύλη (ο διεθνής όρος είναι CCNOT, δηλαδή controlled-controlled NOT) χρειάζεται 3 qubits. Τα 1 qubits ονομάζονται qubits ελέγχου και συμβολίζονται με c_1 και c_2 και το τρίτο ονομάζεται qubit στόχος και συμβολίζεται με t. Η κατάσταση πριν από τη δράση των 2 qubits είναι $|c_{1i}>$,$|c_{2i}>$ και $|t_i>$ και μετά τη δράση γίνονται $|c_{10}>$,$|c_{20}>$ και $|t_0>$. Η πύλη CCNOT αλλάζει την κατάσταση του qubit στόχου, όταν και τα 2 qubits ελέγχου είναι στην κατάσταση |1>. Σε οποιαδήποτε άλλη περίπτωση δεν επηρεάζει τίποτα. Ο πίνακας που περιγράφει την CCNOT είναι ο: $CCNOT=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

και ο πίνακας δράσης :

$\|c_i t_{2i} t_{1i}>$	$\|c_0 t_{20} t_{10}>$
|000>	|000>
|001>	|001>
|010>	|010>
|011>	|011>
|100>	|100>
|101>	|101>
|110>	|110>
|111>	|111>

Κβαντική Πύλη Fredkin

Η κβαντική πύλη Fredkin δρα σε 3 qubits. Το πρώτο ονομάζεται qubit ελέγχου (με συμβολισμό c) και τα 2 επόμενα qubits στόχοι (t_1 και t_2 αντίστοιχα). Οι αρχικές καταστάσεις τους είναι $|c_i>$,$|\ t_{1i}>$, $|t_{2i}>$ και μετά τη δράση της πύλης γίνονται $|c_0>$, $|t_{10}>$, $|t_{20}>$. Η πύλη Fredkin F εναλλάσσει τις καταστάσεις των qubits στόχων όταν το qubit ελέγχου βρίσκεται στην κατάσταση |1>. Όταν το qubit ελέγχου βρίσκεται στην κατάσταση |0>, οι καταστάσεις των qubits δεν επηρεάζονται. Και όπως ίσχυε σε ανάλογη περίπτωση προηγουμένως, η κατάσταση του qubit ελέγχου δεν αλλάζει καθόλου.

Η πύλη περιγράφεται από τον πίνακα F:

$$F=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

και ο πίνακας δράσεων της πύλης είναι ο παρακάτω:

$\|c_i t_{2i} t_{1i}>$	$\|c_0 t_{20} t_{10}>$
|000>	|000>
|001>	|001>
|010>	|010>
|011>	|011>
|100>	|100>
|101>	|110>
|110>	|101>
|111>	|111>

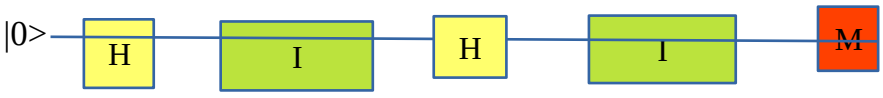
2.7 Κβαντικά Κυκλώματα και κυκλωματικό μοντέλο

Εδώ θα αναλύσουμε τα κβαντικά κυκλώματα και κατ' επέκταση το κβαντικό υπολογιστικό μοντέλο που βασίζεται σε αυτά. Θα λέγαμε ότι ο όρος "κυκλωματικό μοντέλο" ταυτίζεται στην ουσία με τον όρο "κβαντικός υπολογιστής" μιας και είναι το μόνο πραγματικό μοντέλο που έχουμε στη διάθεση μας.

Όπως αναφέραμε, οι κλασικοί υπολογιστές αποτελούνται από αγωγούς και λογικές πύλες, η πληροφορία μεταφέρεται από τη μία πύλη στην άλλη μέσω ηλεκτρικής τάσης και σε κάθε πύλη έχουμε επεξεργασία της πληροφορίας αυτής.

Στους κβαντικούς υπολογιστές από την άλλη πλευρά, η πληροφορία είναι αποθηκευμένη σε qubits ή σε κβαντικούς καταχωρητές και παραμένει εκεί. Οι κβαντικές πύλες, σε αντίθεση με τις λογικές, δεν είναι φυσικά συστήματα αλλά αντιπροσωπεύουν δράσεις που ασκούνται πάνω στα qubits ή τους κβαντικούς καταχωρητές.

Οι κβαντικοί υπολογισμοί είναι δράσεις τελεστών που έχουν ως αποτέλεσμα την περιστροφή διανυσμάτων στο χώρο Hilbert. Σύμφωνα λοιπόν με το κυκλωματικό μοντέλο, κάθε κβαντικός υπολογισμός, όσο δύσκολος κι αν είναι, μπορεί να αναπαρασταθεί με ένα κβαντικό κύκλωμα. Τα κβαντικά κυκλώματα αποτελούνται από qubits, κβαντικούς καταχωρητές και κβαντικές πύλες. Στα κβαντικά κυκλώματα δεν υπάρχει ροή πληροφορίας από πύλη σε πύλη, αλλά διαδοχικές δράσεις κβαντικών πυλών πάνω στους καταχωρητές που είναι αποθηκευμένη η πληροφορία. Από μαθηματικής σκοπιάς, αυτό που συμβαίνει είναι ότι γίνονται πράξεις μεταξύ τανυστών, όπως τις γνωρίζουμε από το πλαίσιο της Γραμμικής Άλγεβρας, όπως γινόμενο μεταξύ πινάκων ή πράξεις μεταξύ πινάκων και διανυσμάτων, αν θεωρήσουμε ότι κάποια διανύσματα γράφονται στη μορφή πινάκων όπως το διάνυσμα στήλη.



Σχήμα 1:απλό κβαντικό κύκλωμα

Στο κύκλωμα παραπάνω έχουμε έναν κβαντικό υπολογισμό. Στην ουσία, η ίδια η δομή του κυκλώματος μας δείχνει τη χρονική πορεία με την οποία γίνονται οι πράξεις πάνω στην αρχική κατάσταση |0>. Εδώ αρχικά έχουμε μια πύλη Hadamard να επιδρά στην αρχική κατάσταση |0>

φέρνοντας το σε κατάσταση υπέρθεσης. Έπειτα έχουμε κατά σειρά μια πύλη αδρανείας I η οποία αλλάζει έστω απειροελάχιστα την κατάστασή μας, μια πύλη Hadamard ξανά και μια πύλη αδρανείας. Στο τέλος έχουμε τη μέτρηση/παρατήρηση της κατάστασης του qubit.

Η μέτρηση είναι μια διαδικασία ΜΗ αναστρέψιμη. Έχουμε την απώλεια της υπέρθεσης, από τις πολλές πιθανές απαντήσεις παίρνουμε μόνο μία. Επίσης, δεν μπορούμε να έχουμε ούτε αντιγραφή πληροφορίας, ούτε επαναλήψεις (for loops) λόγω του No-cloning Theorem [8].

Ο κβαντικός υπολογισμός μπορεί να θεωρηθεί ως ένας ορθομοναδιαίος μετασχηματισμός που αλλάζει την αρχική κατάσταση ενός κβαντικού συστήματος σε μια τελική κατάσταση.

2.7.1 Πλεονεκτήματα και μειονεκτήματα του κυκλωματικού μοντέλου

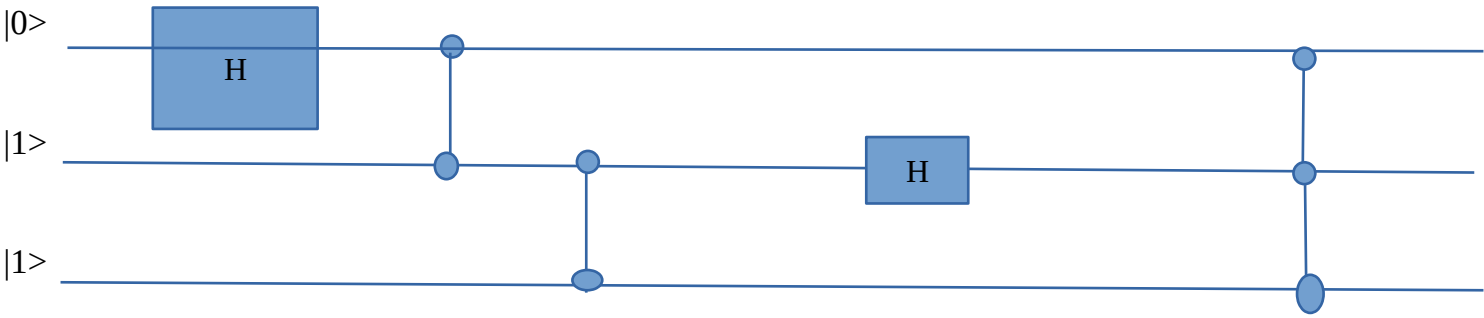
Το βασικό πλεονέκτημα του κβαντικού κυκλωματικού μοντέλου είναι ότι "μοιάζει" αρκετά με το κλασικό κυκλωματικό μοντέλο, άρα ήδη έχουμε μια αρκετά καλή βάση για το πως μπορεί να λειτουργεί. Είναι μια ειδική περίπτωση αυτού του μοντέλου, αφού έχουμε κι εδώ πύλες, κύκλωμα και πράξεις. Υπάρχουν αρκετές γλώσσες κβαντικού προγραμματισμού όπως η Q# [47], η Quipper [48] και η projectQ [49] και βιβλιοθήκες όπως η Qiskit και η Pennylane, που υποστηρίζονται από το κυκλωματικό μοντέλο, είναι βασισμένες σε αρκετές περιπτώσεις σε γλώσσες όπως η python που έχουν πάρα πολλές δυνατότητες και μεγάλη κοινότητα προγραμματιστών.

Ως προς τα μειονεκτήματα, έχουμε ότι 1) για να μπορέσει να προσομοιωθεί ακριβώς μια κβαντική πύλη, χρειάζεται μεγάλος αριθμός υπολογιστικών βημάτων, ίσως και άπειρος, κάτι που είναι πολύ δύσκολο-έως και αδύνατο-να συμβεί. 2)Επίσης, το πλήθος των κβαντικών πυλών που χρειάζονται για να εκτελεστεί ένας κβαντικός αλγόριθμος αυξάνεται εκθετικά με τον αριθμό των qubits που θα χρειαστούν. Ξέρουμε πως ένα τυχαίο qubit μπορεί να βρίσκεται στην υπέρθεση 2 καταστάσεων το πολύ και μια κβαντική πύλη μπορεί να επιδρά σε 2 qubits το πολύ για κάθε χρονική στιγμή. Ας θεωρήσουμε ως παράδειγμα τον αλγόριθμο του Shor [50] .Για να τρέξει σε έναν υποθετικό κβαντικό υπολογιστή των 200 qubits θα χρειαζόταν 2^{200} πύλες, αριθμός πραγματικά τεράστιος. Και τέλος 3) το μοντέλο, πέραν των δυσκολιών που περιγράψαμε, είναι πολύ επιρρεπές σε σφάλματα, επηρεάζεται (στις συσκευές που έχουμε μέχρι στιγμής διαθέσιμες) πολύ από το εξωτερικό περιβάλλον.[9,10,31]

2.7.2 Αναλυτικός Κβαντικός Υπολογισμός Σε Κβαντικό Κύκλωμα

Τώρα θα εξετάσουμε ένα τυπικό παράδειγμα υπολογισμού σε κβαντικό κύκλωμα [10]. Θα δούμε τη βασική δομή ενός κυκλώματος, τις διαφορές με το κλασικό κύκλωμα και περισσότερο διότι αυτό θα μας χρειαστεί αργότερα όταν θα εξετάζουμε την περίπτωση της μάθησης σε κβαντικό επίπεδο με τα variational models που όπως αναφέραμε πολλές φορές, είναι στην ουσία κβαντικά κυκλώματα στα οποία πραγματοποιούνται πράξεις με κάποια παραμετροποίηση.

Έστω το παρακάτω κβαντικό κύκλωμα:



Σχήμα 2: αναλυτικός υπολογισμός κβαντικού κυκλώματος

Θα κάνουμε τον υπολογισμό βήμα προς βήμα, για να αντιληφθούμε επακριβώς τί πράξεις πραγματοποιούνται σε κάθε χρονική στιγμή και πως επηρεάζεται η πληροφορία που περιέχει ο καταχωρητής.

Αρχικά λοιπόν, ο άνω κβαντικός καταχωρητής του κυκλώματος αποτελείται από 3 qubits και η τρέχουσα κατάσταση του είναι η |011>. Ο πίνακας που αντιστοιχεί στην κατάσταση αυτή προκύπτει από το τανυστικό γινόμενο των επιμέρους πινάκων που αντιστοιχούν στις καταστάσεις των 3 qubits:

θυμίζουμε πως $|0\rangle=\begin{bmatrix}1\\0\end{bmatrix}$ και το $|1\rangle=\begin{bmatrix}0\\1\end{bmatrix}$

επομένως θα έχω το εξής τανυστικό γινόμενο:

$$P_1=|011>=\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}0\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix}$$

Αυτό σημαίνει πως αν μετρήσουμε την κατάσταση του κβαντικού καταχωρητή στο 1ο βήμα, θα βρούμε την κατάσταση |011> με πιθανότητα 100%. Από τη στιγμή που έχουμε 3 qubits και το καθένα από αυτά μπορεί να βρίσκεται σε 2 καταστάσεις, τότε μπορούμε να έχουμε συνολικά 2^3=8 καταστάσεις, οι οποίες είναι οι παρακάτω:

|000>=0

|001>=1

|010>=2

|011>=3

|100>=4

|101>=5

|110>=6

|111>=7

Στο 2ο βήμα έχουμε δράση της πύλης Hadamard πάνω στο 1ο qubit και στα άλλα 2 δεν επιδρά καμία πύλη. Όταν δεν επιδρά καμία πύλη πάνω σε κάποιο qubit, θα θεωρούμε πως υπάρχει δράση της πύλης αδρανείας I που αναπαρίσταται από το μοναδιαίο πίνακα.

Η συνολική δράση των πυλών Hadamard και αδρανείας δίνεται όπως πριν με το τανυστικό τους γινόμενο.

$$H\otimes I\otimes I=1/\sqrt{2}*\left(\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\otimes\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\otimes\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)=$$

$$1/\sqrt{2}*\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Και τώρα αυτό το αποτέλεσμα πρέπει να το πολλαπλασιάσουμε με το αρχικό διάνυσμα, τον

$$\text{πίνακα στήλη}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

που ήταν η προηγούμενη κατάσταση, επομένως θα έχουμε:

$$1/\sqrt{2}*\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad *\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} =$$

$$\text{Άρα τελικά έχω: } P_2 = 1/\sqrt{2}*\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Άρα μεταβήκαμε από την κατάσταση |011> στην υπέρθεση των καταστάσεων |011> και |111>. Πιο σωστά, η πιθανότητα να καταλήξουμε αρχικά στην |011> ήταν ίση με 1 ή 100% ενώ τώρα η πιθανότητα να είμαστε στις |011> και |111> είναι ισοπίθανη με ποσοστό 50%. Θυμίζουμε πως η πιθανότητα δίνεται από το τετράγωνο του μέτρου του πλάτους πιθανότητας.

Στο επόμενο 3ο βήμα εφαρμόζεται η πύλη CNOT που όπως είχαμε αναφέρει και πιο πριν χρειάζεται 2 qubits. Εδώ βλέπουμε πως επιδρά στα 2 πρώτα, οπότε για το 3ο θα θεωρήσουμε πάλι πως επιδρά η πύλη αδρανείας. Έχουμε λοιπόν:

$$P_3=(CNOT \otimes I)*P_2=(\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\otimes\begin{bmatrix}1&0\\0&1\end{bmatrix})*1/\sqrt{2}*\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\1\end{bmatrix}=$$

$$1/\sqrt{2}*\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\end{bmatrix}*\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\1\end{bmatrix}=1/\sqrt{2}*\begin{bmatrix}0\\0\\0\\1\\0\\1\\0\\0\end{bmatrix}$$

Μετά το τέλος του 3ου βήματος θα βρισκόμαστε με πιθανότητα 50% είτε στην κατάσταση |011> είτε στην |101>.

Στο 4ο βήμα έχουμε πάλι CNOT απλώς τώρα έχουμε ως qubit ελέγχου το 2ο και ως στόχο το 3ο και πάλι στο qubit που δεν επιδρά κάποια πύλη, θεωρούμε την πύλη αδρανείας. Ομοίως λοιπόν με προηγουμένως έχουμε:

$$P_4=(I \otimes CNOT)*P_3=$$

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\otimes\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}*1/\sqrt{2}*\begin{bmatrix}0\\0\\0\\0\\1\\0\\1\\0\\0\end{bmatrix}=$$

$$1/\sqrt{2}*\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}*\begin{bmatrix}0\\0\\0\\1\\0\\1\\0\\0\end{bmatrix}=1/\sqrt{2}*\begin{bmatrix}0\\0\\1\\0\\0\\1\\0\\0\end{bmatrix}$$

Μετά το πέρας και αυτού του βήματος, θα βρεθούμε με ίση πιθανότητα 50% στις καταστάσεις |010> και |101>.

Στο 5ο βήμα εφαρμόζεται η πύλη Hadamard πάνω στο 2ο qubit, ενώ στα άλλα 2 θα έχουμε την πύλη αδρανείας.

$$P_5=(I\otimes H\otimes I)*P_4=$$

$$\left(\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\otimes 1/\sqrt{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\otimes\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)*1/\sqrt{2}*\begin{bmatrix}0\\0\\1\\0\\0\\1\\0\\0\end{bmatrix}=$$

$$1/2*\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 1/2* \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Μετά το πέρας και αυτού το βήματος έχουμε να μοιράζεται η πιθανότητα να βρούμε τον κβαντικό καταχωρητή σε 4 καταστάσεις πλέον, στις |000>, |010>, |101> και |111>. Το μείον στην κατάσταση |010> υποδηλώνει την ύπαρξη κάποιας φάσης, κατά τ' άλλα δεν μας επηρεάζει σε κάτι, η πιθανότητα είναι 25% για κάθε κατάσταση.

Το επόμενο είναι το 6ο και τελευταίο βήμα όπου έχουμε την πύλη διπλά ελεγχόμενου όχι ή αλλιώς την CCNOT. Εδώ δε θα χρειαστεί να κάνουμε κάποιο τανυστικό γινόμενο, οπότε θα πολλαπλασιάσουμε κατευθείαν τον πίνακα της πύλης με τον πίνακα στήλη που προέκυψε από το προηγούμενο βήμα.

$$P_6 = CCNOT * P_5 =$$

$$1/2*\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Μετά και το πέρας του τελευταίου βήματος, ο κβαντικός καταχωρητής έχει από 25% πιθανότητα να βρίσκεται στις καταστάσεις |000>, |010>, |101> και |110>.

Είδαμε αναλυτικά λοιπόν ένα παράδειγμα κβαντικού υπολογισμού με το κυκλωματικό μοντέλο, δηλαδή έναν κβαντικό υπολογιστή που λειτουργεί με κβαντικά κυκλώματα και αλλάζει την κατάσταση του κβαντικού καταχωρητή και των qubits με τη χρήση κβαντικών πυλών. Είδαμε στην πράξη πως η πληροφορία όπως έχουμε αναφέρει στην κβαντική περίπτωση δε μεταφέρεται, δε διαγράφεται και γενικά δε λειτουργεί όπως στα κλασικά κυκλώματα και τέλος η πύλη αδρανείας, που θα περίμενε κανείς να μην κάνει απολύτως τίποτα, στην ουσία κάνει κάτι.

3 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΚΒΑΝΤΙΚΗ ΜΑΘΗΣΗ

Εδώ είναι το πιο βασικό κομμάτι της παρούσας εργασίας, θα ασχοληθούμε με την Κβαντική Μηχανική Μάθηση, τον συνδυασμό των 2 ίσως πιο πολλά υποσχόμενων αντικειμένων στο χώρο της Επιστήμης των Υπολογιστών. Υπάρχουν πολλοί τομείς στους οποίους η κβαντική μηχανική μάθηση αναμένεται (ή τουλάχιστον ευελπιστούμε) να δώσει τεράστια ώθηση, όπως πχ τα logistics, τα οικονομικά, η επιστήμη των Υλικών, η ιατρική και τέλος η ίδια η φυσική στο κομμάτι που ασχολείται με την κατανόηση του σύμπαντος. Τι εννοούμε όμως με τον όρο Κβαντική μηχανική μάθηση;

Είδη Κβαντικής Μηχανικής Μάθησης		Υπολογιστικό σύστημα	
		Κλασικός υπολογιστής	Κβαντικός υπολογιστής
Τύπος Δεδομένων	Κλασικά Δεδομένα	CC	CQ
	Κβαντικά δεδομένα	QC	QQ

Σχήμα 3: γενικό πλαίσιο κβαντικής μηχανικής μάθησης

Εδώ βλέπουμε ότι το πεδίο της κβαντικής μηχανικής μάθησης χωρίζεται γενικώς σε 4 υποπεδία, αναλόγως με το αν χρησιμοποιούμε κλασικά ή κβαντικά δεδομένα (δηλαδή αν τα δεδομένα είναι κλασική πληροφορία ή αποτελέσματα κβαντικού πειράματος) και το πως γίνεται η επεξεργασία των δεδομένων, αν χρησιμοποιείται κλασικός ή κβαντικός υπολογιστής.
Στο πρώτο block έχουμε το κλασικό μοντέλο μηχανικής μάθησης. Έχουμε κλασικά δεδομένα (πχ εικόνες, ηχητική πληροφορία, κείμενο κλπ) και τα επεξεργαζόμαστε με κλασικό υπολογιστή. Μπορούμε όμως σε κλασικό υπολογιστή πάλι, να έχουμε κβαντικά δεδομένα, που όπως αναφέρθηκε είναι αποτελέσματα πειράματος κβαντικής φυσικής και προφανώς μπορούμε να έχουμε κβαντικά δεδομένα σε κβαντικό υπολογιστή, κάτι που διαισθητικά φαίνεται ως το πιο “σωστό” πρότυπο κβαντικής μηχανικής μάθησης. Στην παρούσα εργασία θα ασχοληθούμε, στο κομμάτι της θεωρίας τουλάχιστον, με το τελευταίο κομμάτι του πίνακα, τα κλασικά δεδομένα σε κβαντικό υπολογιστή. Στον κώδικα παρακάτω θα ασχοληθούμε και με ένα υβριδικό μοντέλο κβαντικής-κλασικής μάθησης, όπου αξιοποιούνται στοιχεία από την κβαντική μηχανική αλλά η βελτιστοποίηση γίνεται κλασικά.
Όπως αναφέραμε και στην αρχή της εργασίας, το μόνο μοντέλο κβαντικού υπολογιστή που έχουμε στη διάθεση μας αυτή την εποχή είναι ο κυκλωματικός κβαντικός υπολογιστής, επομένως όταν μιλάμε για κβαντικό προγραμματισμό εννοούμε κυκλώματα που εκτελούν

κάποιους υπολογισμούς με συγκεκριμένη χρονική σειρά και κβαντική μάθηση είναι τα ίδια κυκλώματα αλλά έχοντας αυτή τη φορά κάποια παραμετροποίηση, οπότε στόχος είναι όπως και στην κλασική μάθηση να βρούμε το σύνολο των βέλτιστων παραμέτρων. Ακόμη, να αναφέρουμε πως είμαστε στην εποχή των NISQ συσκευών, που είναι ευάλωτοι σε σφάλματα, είναι σε ισχύ στα 50 qubits ή μέχρι μερικά εκατοντάδες qubits και τέλος δεν μπορούμε να εκμεταλλευτούμε πλήρως όλα τα θετικά της κβαντικής μηχανικής. Σε κάποιες περιπτώσεις μάλιστα, οι προσομοιωτές δίνουν καλύτερα αποτελέσματα από τον πραγματικό κβαντικό υπολογιστή. Στο μέλλον αναμένεται να έχουμε συσκευές fault-tolerant, που θα μπορούν δηλαδή να συνεχίσουν να δουλεύουν ακόμα και όταν συμβαίνουν σφάλματα χωρίς να καταρρέει όλο το σύστημα.[32,34,36]

Τα κβαντικά μοντέλα χωρίζονται σε 2 μεγάλες κατηγορίες: τα αιτιοκρατικά (deterministic quantum models) και τα στοχαστικά (variational quantum models). Στο ντετερμινιστικό μοντέλο γνωρίζουμε από πριν το αποτέλεσμα με βεβαιότητα, ξέρουμε πχ ότι θα δράσουμε με μια πύλη X πάνω στο qubit |q1> και θα πάρουμε το αποτέλεσμα |q2>. Τέτοιο παράδειγμα είναι ο αλγόριθμος Deutsch-Jozsa [11] και ο αλγόριθμος του Deutsch που ήταν η έμπνευση για τον αλγόριθμο Deutsch-Jozsa (ο δεύτερος είναι η γενίκευση του πρώτου).

Έστω συνάρτηση $f:\{0,1\}\rightarrow\{0,1\}$. Η συνάρτηση αυτή μπορεί να έχει τιμές μόνο το 0 και το 1 όπως είναι εμφανές. Αν οι 2 τιμές είναι ίσες, οι περιπτώσεις δηλαδή $f(0)=0=f(1)$ ή $f(0)=1=f(1)$, τότε λέμε πως η συνάρτηση είναι σταθερή, ενώ όταν $f(0)=1$ αλλά $f(1)=0$ ή $f(0)=0$ αλλά $f(1)=1$, τότε η συνάρτηση δεν είναι σταθερή. Ο αλγόριθμος του Deutsch μας δίνει τη δυνατότητα να βρίσκουμε πότε η συνάρτηση είναι σταθερή ή όχι,γνωρίζοντας μόνο τη μία από τις 2 τιμές! Ο αλγόριθμος των Deutsch και Jozsa γενικεύει το αποτέλεσμα σε συνάρτηση $f:\{0,1\}^n\rightarrow\{0,1\}$ με περισσότερες εισόδους αλλά πάλι 2 τιμές.

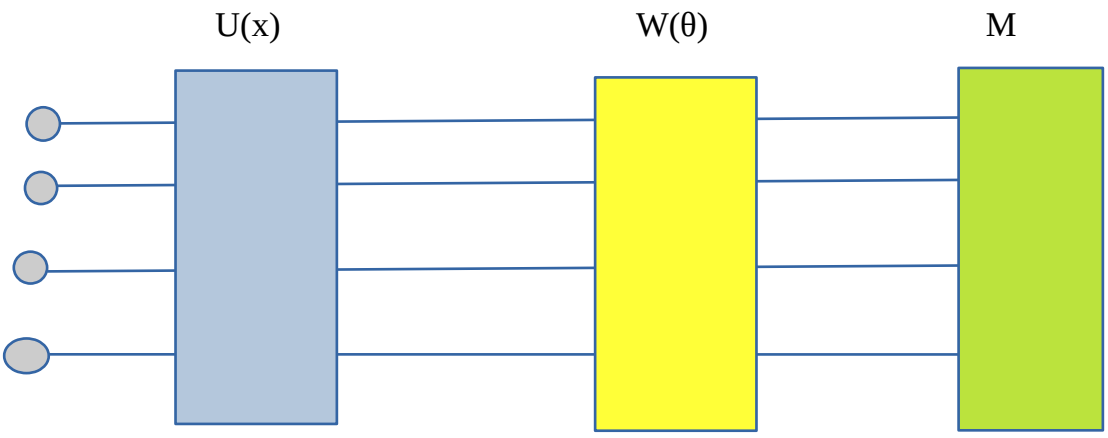
Η άλλη κατηγορία αλγορίθμων είναι οι στοχαστικοί (variational) αλγόριθμοι. Στα στοχαστικά κβαντικά μοντέλα, τα οποία κερδίζουν έδαφος στο χώρο της κβαντικής υπολογιστικής, της κβαντικής μηχανικής μάθησης και της κβαντικής χημείας τα τελευταία χρόνια, το αποτέλεσμα παύει να είναι προβλέψιμο εκ των προτέρων, αλλά θυμίζει περισσότερο κατανομή πιθανότητας κάποιων πιθανών λύσεων. Σε αντιστοιχία με προηγουμένως, θα λέγαμε πως αν έχουμε ένα qubit |q1> και δράσουμε με μια πύλη $P(\theta)$, όπου θ τυχαία παράμετρος από ένα σύνολο παραμέτρων $\Theta=(\theta_1,\ldots..\theta_v)$ τότε το αποτέλεσμα δε θα είναι απλώς μια άλλη κβαντική

κατάσταση |q2> αλλά ένα σύνολο από καταστάσεις $|q_1>,\ldots..,|q_n>$ όπου κάθε μια κατάσταση έχει πιθανότητα pi να είναι αυτή η λύση.

Τέτοιοι αλγόριθμοι είναι ο Variational Quantum Eigensolver (VQE) ο Variational Quantum Classifier, τα κβαντικά ανάλογα των μηχανών διανυσμάτων υποστήριξης (QSVM) και τα κβαντικά νευρωνικά δίκτυα (QNNs).[36]

3.1 Variational Model

Εδώ θα περιγράψουμε σύντομα το γενικό πρότυπο ενός αλγορίθμου κβαντικής μηχανικής μάθησης που ονομάζεται στοχαστικό μοντέλο [32,36]. Ένα variational model είναι “απλώς” ένα κβαντικό κύκλωμα όπως αυτά που αναλύσαμε πιο πάνω, που περιέχει κάποιες παραμέτρους τις οποίες θέλουμε να βελτιστοποιήσουμε και άρα να εκπαιδεύσουμε το μοντέλο με βάση αυτές. Είναι το ανάλογο των βαρών (weights) στα κλασικά νευρωνικά δίκτυα. Και σχηματικά, έστω το παρακάτω κύκλωμα:



Σχήμα 4: variational model

Έστω λοιπόν ότι έχουμε μερικά qubits και τα αρχικοποιούμε στην κατάσταση |0>. Το πρώτο βήμα είναι να μετατρέψουμε τα κλασικά δεδομένα σε κβαντικές καταστάσεις σε έναν χώρο Hilbert και από εκεί να κάνουμε στην ουσία γραμμική άλγεβρα. Το πρώτο κομμάτι αυτό ονομάζεται κωδικοποίηση κλασικών δεδομένων σε κβαντικές καταστάσεις ή data encoding. Από εδώ και πέρα, τα ηνία τα αναλαμβάνει η κβαντική υπολογιστική. Έπειτα έχουμε το επόμενο βήμα που είναι να κάνουμε κάποιες πράξεις πάνω σε αυτά τα qubits για να αλλάξουμε

την κατάσταση τους, με κάποιες κβαντικές πύλες ή γενικώς έναν ορθομοναδιαίο τελεστή. Οι πράξεις αυτές στην ουσία θα έχουν ως αποτέλεσμα το να περιστραφούν τα διανύσματα των κβαντικών καταστάσεων στο χώρο Hilbert, τα διανύσματα που είναι οι απεικονίσεις των δεδομένων μας. Οι κβαντικές πύλες που κάνουν αυτές τις πράξεις εξαρτώνται από κάποιες παραμέτρους θ^i από ένα σύνολο παραμέτρων $\Theta=(\theta^1,\ldots\ldots,\theta^n)$. Οι περιστροφές (rotation) που θα γίνουν στο χώρο Hilbert είναι κατά γωνία θ^i. Επομένως μετά από πράξεις έχουμε φτάσει σε μια κατάσταση όπου τα δεδομένα έχουν μετασχηματιστεί από μια συνάρτηση f και εξαρτώνται από κάποιες παραμέτρους: $|f(x,\theta)>$. Το επόμενο στάδιο είναι διαφορετικό σε σύγκριση με το κλασικό μοντέλο, είναι το στάδιο της μέτρησης/παρατήρησης (measurement). Στη κβαντική θεωρία όπως αναφέραμε, μπορεί ένα qubit να βρίσκεται σε παραπάνω από μία καταστάσεις, οπότε όταν παρατηρούμε τα δεδομένα, βλέπουμε την κατάσταση που είχε μεγαλύτερη πιθανότητα να εμφανιστεί και ΟΛΕΣ οι άλλες καταστάσεις καταρρέουν καθώς καταρρέει η υπέρθεση (superposition). Το τελικό στάδιο λοιπόν είναι η κατάσταση y' που παρατηρήθηκε προηγουμένως, που πάντα θα διαφέρει έστω λίγο από την επιθυμητή κατάσταση y, και τα υπόλοιπα είναι ήδη γνωστά από την κλασική περίπτωση: υπολογίζεται αυτή η διαφορά $y-y'$, αλλάζουν οι παράμετροι θ^i και ξαναγίνεται ο υπολογισμός από την αρχή με τις νέες παραμέτρους, παρατηρούμε το αποτέλεσμα ξανά κλπ μέχρις ότου το σφάλμα να πέσει κάτω από κάποιο αποδεκτό όριο που έχουμε θέσει. Αυτό είναι γενικά το πλαίσιο που θα μπορούσε με κάποια τροποποίηση να ονομαστεί και Κβαντικό Νευρωνικό Δίκτυο, ίσως με παραπάνω στάδια πράξεων με κβαντικές πύλες, πχ αντί για W(θ) να έχουμε W(θ1),W(θ2),…. αν και πολλοί ερευνητές του χώρου διαφωνούν με τον όρο, κάνοντας λόγο για “παραμετροποιημένα κβαντικά κυκλώματα”. Δεν έχει βρεθεί κάποιο μοντέλο που θα μπορούσε να οριστεί ως Κβαντικός Νευρώνας, έχουμε μόνο γραμμικές πράξεις, δεν μπορούμε να έχουμε την ευελιξία των συναρτήσεων ενεργοποίησης όπως πχ η ReLU, δεν υπάρχει δυνατότητα χρήσης αλγορίθμου όπως ο backpropagation κλπ. Μπορούμε ίσως να ορίσουμε την έννοια των γραμμικών QNNs. Οπότε ο όρος Κβαντικά Νευρωνικά δίκτυα σε γενικές γραμμές προς το παρόν αναφέρεται σε κάποια μορφή των Variational Models και στην ουσία χρησιμοποιείται για λόγους Marketing.

3.2 Μετατροπή Δεδομένων σε Κβαντικές Καταστάσεις (Data Encoding)

Ας τα δούμε πιο αναλυτικά από την αρχή. Το πρώτο στάδιο της Κβαντικής Μάθησης που θα εξετάσουμε είναι το πως μετασχηματίζονται τα δεδομένα από την κλασική τους μορφή σε κβαντικές καταστάσεις σε χώρο Hilbert όπως είδαμε ήδη, η κωδικοποίηση των δεδομένων (data encoding) [34,35]. Θεωρείται το πιο βασικό κομμάτι της μάθησης, διότι άπαξ κι έχουμε διαθέσιμα τα δεδομένα μας ως κβαντικές καταστάσεις, μπορούμε να εκμεταλλευτούμε όλα τα στοιχεία από την κβαντική φυσική για τα οποία και γίνεται όλη αυτή η διαδικασία. Επίσης, όταν βρισκόμαστε σε χώρο Hilbert, όλες οι πράξεις από εκεί και μετά είναι γραμμικές και τέλος, όλη αυτή η κωδικοποίηση μπορεί να αντιστοιχηθεί με το κομμάτι των χώρων χαρακτηριστικών (feature maps) που ξέρουμε από την κλασική μάθηση και συγκεκριμένα για μεθόδους πυρήνα (kernel methods): εκεί απεικονίζουμε τα δεδομένα σε χώρο μεγαλύτερης διάστασης για να μπορούμε να κάνουμε αποτελεσματικά κατηγοριοποίηση (classification). Εδώ, τα δεδομένα βρίσκονται ως κβαντικές καταστάσεις στο χώρο Hilbert που είναι χώρος μεγάλης διάστασης, μεγαλύτερης από τον αρχικό χώρο δεδομένων (data space).

Ως προς τη σύγκριση με την κλασική περίπτωση, θα μπορούσαμε να πούμε πως στην Κλασική Μάθηση, η διαδικασία σχηματικά είναι η ακόλουθη:

Εισαγωγή δεδομένων στο μοντέλο → Επεξεργασία των δεδομένων με κάποιον αλγόριθμο → Εξαγωγή Αποτελέσματος.

Στην κβαντική μάθηση,η ίδια διαδικασία θα μπορούσε να περιγραφεί ως εξής:

Χώρος Δεδομένων → Μετατροπή Δεδομένων σε Κβαντικές Καταστάσεις (data encoding) → Επεξεργασία Δεδομένων → Παρατήρηση (measurement) → Εξαγωγή Αποτελέσματος.

Για την μετατροπή τώρα των δεδομένων σε κβαντικές καταστάσεις, υπάρχουν διάφορες τεχνικές, θα δούμε τις πιο δημοφιλείς εδώ, αν και σε γενικές γραμμές παραμένει ανοιχτό ερώτημα επιστημονικά το ποια είναι η βέλτιστη τεχνική.

Basis Encoding

Το πρώτο λοιπόν που θα δούμε είναι η κωδικοποίηση ως προς τις βασικές καταστάσεις (basis encoding) [12]. Όταν έχουμε κβαντικούς υπολογισμούς ή γενικότερα κάποιο κβαντικό κύκλωμα, είδαμε πως έχουμε κάποιες βασικές καταστάσεις (basis states), όπως πχ οι

καταστάσεις |0> και |1> που είδαμε αρχικά στην παρούσα εργασία. Άρα κωδικοποιούμε την πληροφορία με βάση αυτές τις καταστάσεις. Επομένως και τα δεδομένα μας θα πρέπει να είναι σε δυαδική μορφή. Έστω λοιπόν σύνολο D=(x^1,…...x^n), όπου το κάθε $x^k \in D$ είναι στη μορφή (b_1^k,…...b_N^k) με $b_i^k \in 0,1$για i=1,…..N.

Οπότε μπορούμε να κωδικοποιήσουμε το σύνολο D ως κβαντική κατάσταση:

$$|D>=\frac{1}{\sqrt{M}}\sum_{m=1}^{M}|x^m>,$$

Δημιουργείται στην ουσία μια υπέρθεση από βασικές καταστάσεις που αντιστοιχούν επακριβώς στα δυαδικά μας δεδομένα.

Ας δούμε όμως ένα παράδειγμα για να γίνει πιο κατανοητό.

Έστω λοιπόν 2 σημεία δεδομένων x^1 και x^2 ως εξής:

$$x^1=(0,1)^T$$

$$x^2=(1,1)^T$$

Μετατρέπουμε τις 2 αυτές καταστάσεις σε κβαντική μορφή:

$$x^1=|01>$$

$$x^2=|11>$$

Αυτές είναι οι βασικές καταστάσεις (basis states) που αντιστοιχούν στα δεδομένα.

Κατασκευάζουμε μια υπέρθεση των 2 αυτών καταστάσεων σε κύκλωμα που έχει 2 qubits.

$$\frac{1}{\sqrt{4}}*(|00>+|01>+|10>+|11>).$$

Εδώ έχουμε στην ουσία μια πιθανότητα όλων των βασικών καταστάσεων με 2 qubits. Οι καταστάσεις είναι ισοπίθανες. Με άλλα λόγια, αν έχουμε κύκλωμα με 2 qubits, όλες οι πιθανές καταστάσεις που μπορούμε να πάρουμε είναι οι |00> , |01> , |10> και |11>, με πιθανότητα 25% η κάθε μία. Το διάνυσμα των amplitudes (πλάτους πιθανότητας) θα ήταν το

$$(\frac{1}{\sqrt{4}},\frac{1}{\sqrt{4}},\frac{1}{\sqrt{4}},\frac{1}{\sqrt{4}})^T.$$

Εμείς όμως χρειαζόμαστε κάτι αντιπροσωπευτικό των δεδομένων μας, οπότε δε χρειαζόμαστε και τις 4 καταστάσεις, μιας κι έχουμε μόνο τις |01| και |11>. Άρα το διάνυσμα των πιθανοτήτων γίνεται πλέον:

$$(0, \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})^T$$

Τελικά θα έχουμε πως τα δεδομένα κωδικοποιούνται στη μορφή:

$$|D>=\frac{1}{\sqrt{2}}*|01>+ \frac{1}{\sqrt{2}}*|11>.$$

Στο παράδειγμα που είδαμε, θεωρήσαμε κύκλωμα με 2 qubits, οι συνολικές καταστάσεις του οποίου είναι 2^2=4. Εάν είμαστε σε μεγαλύτερο κύκλωμα με περισσότερες καταστάσεις και μεγαλύτερη διάσταση, επιζητούμε μεγαλύτερη ακρίβεια, επομένως το διάνυσμα πλάτους πιθανότητας γίνεται αρκετά έως πολύ αραιό (sparse), έχει δηλαδή πολλές θέσεις με μηδενικά και ελάχιστες με αριθμό διάφορο του μηδενός που υποδηλώνει βασική κατάσταση. Όμως, η τεχνική αυτή είναι από τις πιο γρήγορες ως προς το μετασχηματισμό δεδομένων, μας δίνει μεγάλη ελευθερία για το πως γίνονται οι υπολογισμοί και παραμένει ο πιο εύκολος τρόπος μετασχηματισμού.

Amplitude Encoding

Μια άλλη τεχνική που έχουμε στη διάθεση μας είναι η κωδικοποίηση με βάση το διάνυσμα του πλάτους πιθανότητας, ή αλλιώς το **amplitude encoding** [13]. Ακριβώς αντίστοιχα με προηγουμένως που θέλαμε να μετατρέψουμε τα κλασικά δεδομένα σε κβαντικές καταστάσεις ως προς τις βασικές καταστάσεις, εδώ η μετατροπή βασίζεται στο συσχετισμό των κλασικών δεδομένων με τις πιθανότητες της κάθε κβαντικής κατάστασης. Ας δούμε ένα παράδειγμα κι εδώ, για να το κατανοήσουμε καλύτερα. Έστω 2 σημεία δεδομένων όπως πριν (data points):

$$x^1=(0.888, -1.25)^T$$

$$x^2=(-0.23, 0.992)^T$$

Εδώ δε χρειάζεται να είναι τα δεδομένα σε δυαδική μορφή. Αρχικά, γράφουμε τις συντεταγμένες των 2 διανυσμάτων ως συντεταγμένες ενός διανύσματος:

$$\frac{1}{\sqrt{4}}(0.888, -1.25, -0.23, 0.992)^T$$

και μετά συσχετίζουμε αυτές τις συντεταγμένες με κάποιες βασικές καταστάσεις:

$$(|00>, |01>, |10>, |00>)$$

δηλαδή η πρώτη συντεταγμένη 0.888 σχετίζεται με την κατάσταση |00> κ.ο.κ..

Επομένως το σύνολο των δεδομένων θα κωδικοποιείται ως προς τις πιθανότητες να εμφανιστεί κάθε μία από τις βασικές καταστάσεις:

$$|D>=\frac{1}{\sqrt{2^n}}\sum_{i=1}^{2^n}|x_i i>$$

Αυτή η τεχνική μπορεί να μετατρέψει σε κβαντικές καταστάσεις 2^n στοιχεία, με n να είναι το πλήθος των qubits που χρειαζόμαστε. Εδώ πχ έχουμε 4 καταστάσεις και χρειάζονται 2 qubits. Σε μεγαλύτερες διαστάσεις αυτή η τεχνική είναι πολύ αποδοτική, μιας και μπορεί να εκμεταλλευτεί τον εκθετικά μεγαλύτερο χώρο Hilbert (σε σύγκριση πάντα με τον αρχικό χώρο δεδομένων). Για παράδειγμα, αν έχουμε 512 στοιχεία προς μετατροπή σε κβαντική κατάσταση:

$$x=(x^1,\ldots\ldots\ldots,x^{512})$$

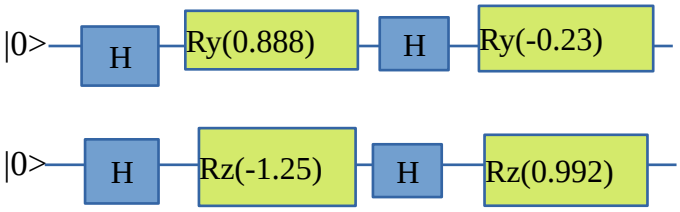
τότε θα χρειαστούμε n=9 qubits μόλις, μιας και $2^9=512$.

Angle Encoding

Μια τρίτη εκδοχή μετατροπής δεδομένων σε κβαντικές καταστάσεις είναι η κωδικοποίηση της κλασικής πληροφορίας σε γωνίες περιστροφής (rotation) [14]. Πιο συγκεκριμένα, έχουμε έστω κάποια qubits, που βρίσκονται σε χώρο Hilbert ως διανύσματα. Η γωνία περιστροφής των διανυσμάτων αυτών είναι η τιμή ακριβώς των δεδομένων. Ας δούμε πάλι ένα παράδειγμα:
Έστω πάλι 2 data points:

x1=(0.888,-1.25)

x2=(-0.23,0.992)



Σχήμα 5: angle encoding

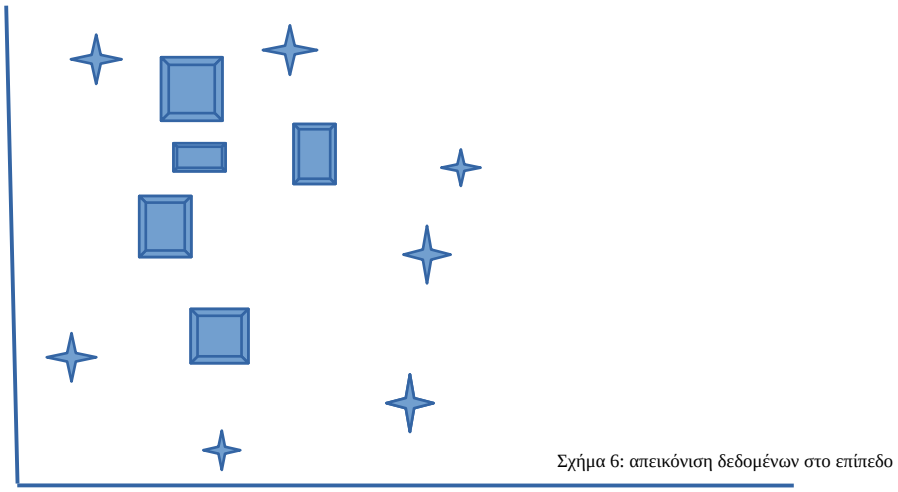
Έχουμε λοιπόν πως αρχικά εφαρμόζουμε πύλες Hadamard και στα 2 qubits για να τα φέρουμε σε κατάσταση υπέρθεσης. Το επόμενο βήμα είναι να περιστρέψουμε το κάθε ένα από αυτά τα 2 qubit κατά μια γωνία με βάση τις συντεταγμένες των διανυσμάτων, για παράδειγμα εδώ περιστρέφουμε το πρώτο qubit κατά μια γωνία ως προς τον άξονα των y:Ry(0.888) ενώ το δεύτερο qubit κατά γωνία ως προς τον z:Rz(-1.25). Άρα ως εδώ έχουμε κωδικοποιήσει το

πρώτο διάνυσμα εφαρμόζοντας περιστροφές ίσες με τις τιμές των συντεταγμένων του πρώτου διανύσματος. Ομοίως προχωράμε, εφαρμόζοντας ξανά πύλη Hadamard και στα 2 qubits και μετά rotations με βάση τώρα τις συντεταγμένες του δεύτερου διανύσματος ως προς κάποιον άξονα κάθε φορά.

Υπάρχουν και άλλες τεχνικές όπως το higher order embedding, το variational/trained embedding κ.α.[15,16] που λειτουργούν με παρόμοιο τρόπο. Σε κάθε περίπτωση 1ον οι διαφορές έχουν να κάνουν με το ποια τεχνική είναι πιο αποδοτική και με τη μικρότερη δυνατή πολυπλοκότητα, 2ον όπως αναφέρθηκε στην αρχή της ενότητας, το ποια είναι η “τέλεια” τεχνική παραμένει ανοιχτό πρόβλημα έως τη στιγμή που γράφεται η παρούσα εργασία και τέλος είναι ίσως το πιο βασικό στοιχείο στο γενικό πλαίσιο της μάθησης με κβαντικούς υπολογιστές: από αυτό το σημείο και μετά ασχολούμαστε με κβαντικούς αλγορίθμους, πράξεις με κβαντικές πύλες πάνω σε qubits κλπ οπότε είναι πολύ σημαντικό να έχουμε μετατρέψει “σωστά” τα δεδομένα σε κβαντικές καταστάσεις, για να έχουμε την περισσότερη δυνατή πληροφορία και για να γίνει όσο πιο βέλτιστα η διαδικασία της μάθησης.

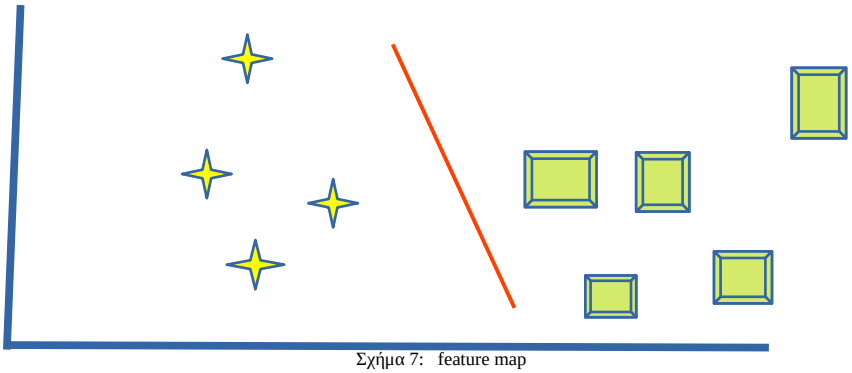
3.3 Quantum Feature Maps

Ας κάνουμε τώρα μια μικρή εισαγωγή τώρα στις κβαντικές απεικονίσεις χαρακτηριστικών [17,18,35]. Όπως είδαμε και στην κλασική περίπτωση, τα feature maps είναι εκείνες οι συναρτήσεις που απεικονίζουν τα δεδομένα μας σε χώρο μεγαλύτερης διάστασης (όπως ακριβώς είδαμε στις μηχανές διανυσμάτων υποστήριξης, ή αλλιώς SVM) όταν δε γίνεται να διαχωριστούν γραμμικώς στον αρχικό χώρο δεδομένων. Όταν για παράδειγμα τα δεδομένα είναι όπως στο παρακάτω σχήμα:



Σχήμα 6: απεικόνιση δεδομένων στο επίπεδο

και όπως είναι εμφανές δεν μπορούμε να τα διαχωρίσουμε γραμμικώς στο επίπεδο,τότε τα απεικονίζουμε σε χώρο μεγαλύτερης διάστασης,διότι γενικά ο στόχος μας είναι να τα διαχωρίζουμε με τον πλέον απλό και σύντομο τρόπο. Χάρη στους feature maps οδηγούμαστε στο παρακάτω σχήμα:



Σχήμα 7: feature map

Να το δούμε και πιο μαθηματικώς ορισμένο: έστω ο χώρος των αρχικών δεδομένων X. Η απεικόνιση των χαρακτηριστικών ή feature map στην κλασική μηχανική μάθηση είναι μια συνάρτηση $\varphi: X \rightarrow F$, με F να ορίζεται ο χώρος των χαρακτηριστικών (feature space). Στην κβαντική μηχανική μάθηση, η μόνη διαφορά είναι ότι ο feature space είναι χώρος Hilbert και τα στοιχεία που περιέχει επομένως είναι διανύσματα που έχουν την πληροφορία των χαρακτηριστικών των δεδομένων μας κωδικοποιημένη (feature vectors). Ο μετασχηματισμός των δεδομένων από x σε $|\varphi(x)\rangle$ γίνεται μέσω ορθομοναδιαίου πίνακα $U_{\varphi}(x)$, ή πιο γενικά από ένα variational κύκλωμα με πολλούς τέτοιους πίνακες, με το κύκλωμα αυτό να εξαρτάται από κάποιες παραμέτρους που συνδέονται με τα δεδομένα εισόδου.

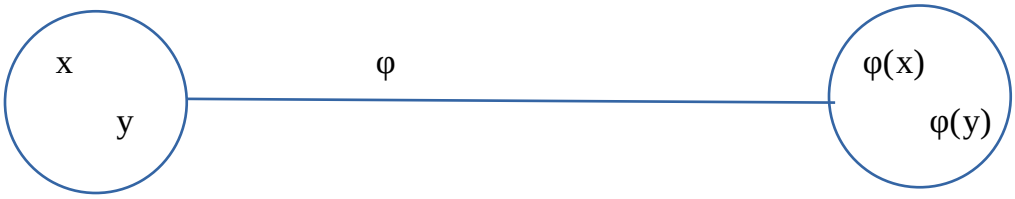
Με βάση τα παραπάνω, καλό θα ήταν να κάνουμε μια αναφορά κάπως πιο διεξοδικά και στις μεθόδους πυρήνα (kernel methods) που χρησιμοποιείται και στον αλγόριθμο μηχανών διανυσμάτων υποστήριξης (Support vector machine). Η συνάρτηση του πυρήνα (kernel function) που είναι θεμελιώδης έννοια για τις μεθόδους πυρήνα, συνδέεται με τη συνάρτηση των χαρακτηριστικών (feature map) και μας δείχνει ομοιότητα των στοιχείων στον feature space.

Με πιο απλά λόγια, ας δούμε σε βήματα αναλυτικά, όλα αυτά που έχουμε πει μέχρι τώρα: έχουμε αρχικώς τα δεδομένα σε έναν τυχαίο χώρο δεδομένων. Ας θεωρήσουμε για χάριν της απλότητας ότι αυτός ο χώρος είναι το επίπεδο και ότι τα δεδομένα είναι τυχαία διάσπαρτα σημεία στο επίπεδο. Επειδή λόγω της πιθανώς αδόμητης φύσης των δεδομένων αυτών να μην

μπορούμε να κάνουμε κάτι βασικό όπως κατηγοριοποίηση με εύκολο και γρήγορο τρόπο (πιο σωστά: δεν είναι γραμμικώς διαχωρίσιμα τα δεδομένα, δεν μπορώ να τα διαχωρίσω σε πχ 2 βασικές κλάσεις με μια ευθεία γραμμή) απεικονίζουμε τα δεδομένα από τον τυχαίο αυτό χώρο δεδομένων, σε χώρο μεγαλύτερης διάστασης, λχ τον R^3, τον γνωστό μας τρισδιάστατο ευκλείδειο πραγματικό χώρο. Αυτή η διαδικασία γίνεται χάρη στη συνάρτηση των χαρακτηριστικών (feature map) που παίρνει τα σημεία του R^2 και τα απεικονίζει σε σημεία του R^3. Τα δεδομένα πλέον είναι τα φ(x) στον feature space και πλέον είναι πιθανό να είναι διαχωρίσιμα με κάποιο επίπεδο. Θέλουμε λοιπόν μια συνάρτηση η οποία να μετράει την ομοιότητα που έχουν τα δεδομένα στον feature space, να υπολογίζει με κάποιο τρόπο την απόσταση και αυτή η απόσταση να μας βοηθάει στο να κατηγοριοποιήσουμε πιο εύκολα, πχ μικρότερη απόσταση μεταξύ 2 σημείων να σημαίνει πως αυτά τα 2 σημεία ανήκουν στην ίδια κλάση.

Ορίζουμε λοιπόν μια συνάρτηση πυρήνα k (kernel function) ως εξής:

χώρος δεδομένων χώρος χαρακτηριστικών



Σχήμα 8:kernel function

Αν έχω 2 σημεία στον αρχικό χώρο,έστω x,y η συνάρτηση του πυρήνα ορίζεται ως εξής:

$$k:X*X \rightarrow C \text{ με } k(x,y)=<\varphi(x),\varphi(y)>$$

η συνάρτηση αυτή συνδέει την απόσταση των x και y με το εσωτερικό γινόμενο των εικόνων τους στο χώρο των χαρακτηριστικών. Ισχύει δηλαδή ότι ο πυρήνας είναι εσωτερικό γινόμενο διανυσμάτων του μεγαλύτερου σε διάσταση χώρου κι επίσης μπορούμε να πούμε μαθηματικώς ότι κάθε φορά που έχω ένα εσωτερικό γινόμενο εγγυημένα θα έχουμε και κάποιον πυρήνα.

Ας έρθουμε στο σημαντικό κομμάτι της βελτιστοποίησης στο χώρο της κβαντικής μηχανικής μάθησης. Όπως αναφέρθηκε ήδη, τα βήματα είναι η μετατροπή των κλασικών δεδομένων σε κβαντικές καταστάσεις (όπως είναι πασιφανές, αυτό το βήμα δε χρειάζεται καν όταν έχω κβαντικά δεδομένα εξ’ αρχής, με κβαντικά δεδομένα θα δουλέψουμε παρακάτω γι αυτό και η

αναφορά) η διαδικασία πράξεων με τις πύλες που έχουν κάποια παράμετρο θ (που είναι η γωνία περιστροφής στο χώρο Hilbert) και η παρατήρηση του αποτελέσματος, που ως γνωστόν έχει ως αντίκτυπο την κατάρρευση της υπέρθεσης. Από εδώ και πέρα, θα μας ήταν πολύ χρήσιμο να έχουμε κάτι αντίστοιχο με της μεθόδους gradient-descent που έχουμε στην κλασική περίπτωση. Θα ήταν αρκετά βολικό να μπορούμε να κάνουμε βελτιστοποίηση των παραμέτρων θ υπολογίζοντας κάποιες παραγώγους. Όμως εδώ τί μπορεί να παραγωγιστεί;

4 ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΚΒΑΝΤΙΚΩΝ ΜΟΝΤΕΛΩΝ (variational models)

Η εκπαίδευση ενός παραμετροποιημένου κβαντικού κυκλώματος είναι η εύρεση των βέλτιστων παραμέτρων θi ώστε να ελαχιστοποιείται μια συνάρτηση κόστους C(θ). Θα δούμε 2 βασικούς τρόπους εκπαίδευσης, ο ένας είναι δανεισμένος από την κλασική Βαθιά Μάθηση (Deep Learning) και ο άλλος είναι πιο πρόσφατος τρόπος που έγινε για τις ανάγκες της κβαντικής μάθησης.

Αρχικά λοιπόν θα δούμε την περίπτωση της **Αυτόματης Παραγώγισης** (Automatic Differentiation) [36,37]. Η μέθοδος αυτή είναι κατάλληλη για αιτιοκρατικά μοντέλα, όμως μπορεί να επεκταθεί και σε πιθανοκρατικά μοντέλα εξίσου. Είναι ένα προγραμματιστικό παράδειγμα που μας παρέχει αυτομάτως όλες τις μερικές παραγώγους $\partial_\mu f_\theta$ μιας διαφορίσιμης συνάρτησης f_θ και θα μπορούσαμε να θεωρήσουμε πως είναι μια εφαρμογή του γνωστού μας από το Διαφορικό Λογισμό, Κανόνα της Αλυσίδας. Για παράδειγμα, όταν εκπαιδεύουμε νευρωνικά δίκτυα, το πρόγραμμα ήδη γνωρίζει πώς να κάνει τον αλγόριθμο backpropagation.

Έστω μια συνάρτηση κόστους (=συνάρτηση που υπολογίζει τις απώλειες, το πόσο διαφορετικό είναι το αποτέλεσμα της παρατήρησης από το επιθυμητό) C(θ) που εξαρτάται από το παραμετροποιημένο κύκλωμα (variational model) f, το οποίο με τη σειρά του εξαρτάται από τις παραμέτρους Θ. Η μερική παράγωγος $\partial_\mu C$της συνάρτησης κόστους λοιπόν ως προς μ, με $\mu \in \Theta$μπορεί να υπολογιστεί κατά τα γνωστά από τα μαθηματικά ως εξής:

$$\partial_\mu C(\theta) = \frac{\partial C}{\partial f_\theta} * \frac{\partial f_\theta}{\partial \mu}$$

Το λογισμικό ήδη γνωρίζει πως να υπολογίσει τις 2 αυτές μερικές παραγώγους. Στο κομμάτι της εκπαίδευσης των παραμετροποιημένων κυκλωμάτων, η μερική παράγωγος $\frac{\partial C}{\partial f_\theta}$είναι ένας

κλασικός υπολογισμός, που θα μπορούσε να πραγματοποιηθεί με κλασικές βιβλιοθήκες περί διαφορισιμότητας. Όμως, η μερική παράγωγος f_θ είναι αποτέλεσμα ενός κβαντικού υπολογισμού, πρέπει να βρούμε τρόπο να υπολογίσουμε τη μερική παράγωγο $\frac{\partial f_\theta}{\partial \mu}$.

Η έννοια της παραγώγου

Ας κάνουμε μια μικρή αναφορά στην έννοια της παραγώγου, στο κομμάτι που θα μας χρειαστεί, μιας και είναι αρκετά επεκταμένο αντικείμενο στο χώρο των μαθηματικών και δε θα χρειαστούμε όλες τις λεπτομέρειες. Μπορούμε σε γενικές γραμμές να θεωρήσουμε ότι υπάρχουν 4 είδη πραγματικών συναρτήσεων:

1) Οι συναρτήσεις f:$R \rightarrow R$, όπου η f έχει ως όρισμα πραγματικούς αριθμούς και δίνει πραγματικές τιμές. Η παράγωγος μιας τέτοιας συνάρτησης είναι πάλι μια συνάρτηση f΄ όπως και η αρχική f.

2) Μπορεί να έχουμε μια συνάρτηση f:$R \rightarrow R^N$όπου η f εδώ είναι διάνυσμα, ή αλλιώς παίρνει πραγματικούς αριθμούς και δίνει αποτέλεσμα διάνυσμα συναρτήσεων, άρα είναι της μορφής

$$f(x)=(f_1(x),\ldots\ldots\ldots f_n(x))$$

και άρα η παράγωγος μιας τέτοιας συνάρτησης είναι πάλι διάνυσμα

$$f'(x)=(f'_1(x),\ldots\ldots\ldots f'_n(x))$$

εδώ ακόμα διαχειριζόμαστε μία μεταβλητή.

3) Υπάρχουν και οι f:$R^N \rightarrow R$, συναρτήσεις πολλών μεταβλητών που έχουν ως σύνολο τιμών τους πραγματικούς αριθμούς

$$f(x_1,\ldots\ldots\ldots x_n)=f(\overline{x})$$

και παράγωγο ένα διάνυσμα που ονομάζεται ανάδελτα της f και είναι της μορφής:

$$\nabla f_\theta=(\partial_{\theta 1} f_\theta,\ldots\ldots\ldots,\partial_{\theta_\kappa} f)$$

Η παράγωγος της f ως προς κάθε παράμετρο $\theta=(\theta_1,\ldots..,\theta_\kappa)$.

4) Τέλος έχουμε και τις συναρτήσεις της μορφής f:$R^N \rightarrow R^D$,συναρτήσεις-διανύσματα με πολλές μεταβλητές:

$$\overline{f}(\overline{x})=\begin{matrix} f_1(x_1) & \ldots & f_1(x_n) \\ \ldots & \ldots & \ldots \\ f_n(x_1) & \ldots & f_n(x_n) \end{matrix}$$

όπου η παράγωγος είναι ένας πίνακας,που ονομάζεται Ιακωβιανή (Jacobian):

$$J(f_\theta) = \begin{matrix} \partial_{\theta_1} f_{\theta_1} & \dots & \partial_{\theta_1} f_{\theta_D} \\ \dots & \dots & \dots \\ \partial_{\theta_k} f_{\theta_1} & \dots & \partial_{\theta_k} f_{\theta_D} \end{matrix}$$

Στις 2 τελευταίες περιπτώσεις έχουμε την έννοια της μερικής παραγώγου, όπου μπορούμε να υπολογίσουμε παράγωγο ως προς μία από όλες τις μεταβλητές.

Πώς όμως μπορούμε να υπολογίσουμε την μερική παράγωγο στη δικιά μας περίπτωση; Δηλαδή αυτή του κβαντικού μοντέλου μας; Θα μπορούσαμε να πούμε πως κάνουμε μια αριθμητική προσέγγιση, οπότε έχουμε πως η παράγωγος ενός παραμετροποιημένου κυκλώματος f_θ ως προς μια παράμετρο $\mu \in \theta$ με τη γνωστή μέθοδο των Πεπερασμένων Διαφορών:

$$\frac{\partial f_\theta}{\partial \mu} \approx \frac{f_\theta - f_{\theta+\Delta\theta}}{|\Delta\theta|}.$$

Κάτι που θυμίζει τον ορισμό της παραγώγου μιας μεταβλητής που υπολογίζουμε τη διαφορά μεταξύ 2 πολύ κοντινών σημείων πάνω στην καμπύλη της f. Το Δθ είναι απειροστά διαφορετικό από το θ και το $\Delta\theta$ είναι το πόσο έχει μεταβληθεί το μ, δηλαδή είναι μια ποσότητα μ+Δμ, όπου πάλι το Δμ είναι απειροστά μικρό. Όμως αυτή η τεχνική έχει το μειονέκτημα ότι προσεγγίζει απλώς την τιμή της μερικής παραγώγου του μοντέλου, είναι μια εκτίμηση της τιμής που όμως περιέχει σφάλμα. Επομένως είναι δύσκολη μέθοδος όταν πρέπει να υπολογίσουμε με μεγάλη ακρίβεια το ελάχιστο της συνάρτησης κόστους ή όταν ο χώρος στον οποίο αναζητούμε έχει πολλά σαγματικά σημεία (saddle points) ή τέλος όταν ο αλγόριθμος κάνει παρατηρήσεις στο τέλος της εκπαίδευσης οι οποίες έχουν μεγάλη διακύμανση.

Οπότε το να μπορούμε να υπολογίζουμε ακριβώς την παράγωγο είναι πολύ σημαντικό και χρηστικό, αν και δεν είναι εύκολη υπόθεση πολλές φορές.

4.1 Βελτιστοποίηση σε κβαντικά μοντέλα και προβλήματα

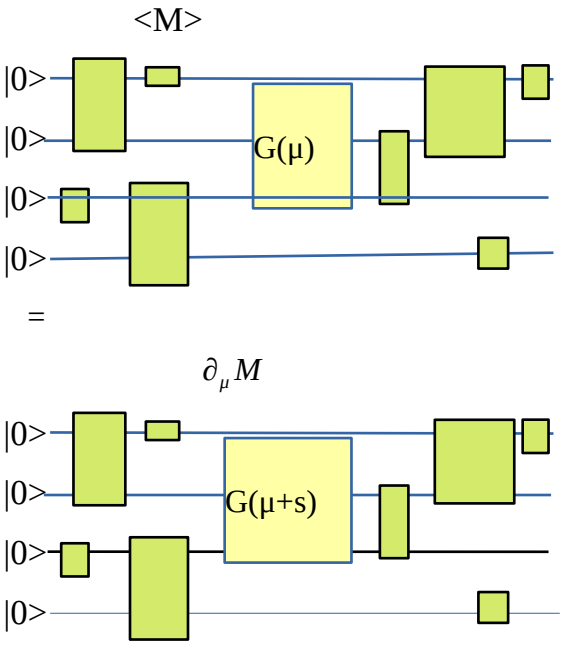
Κανόνες Μετατόπισης Της Παραμέτρου (Parameter-Shift Rules)

Οδηγούμαστε λοιπόν σε νέες μεθόδους όπου ο υπολογισμός της μερικής παραγώγου (και κατά συνέπεια η διαδικασία της εκπαίδευσης) γίνεται με τις παραμέτρους να έχουν αλλάξει, από μ σε μ+s [36].

Ορισμός: έστω $f_\mu = \langle M \rangle_\mu$ το αναμενόμενο αποτέλεσμα της εκπαίδευσης που εξαρτάται από την παράμετρο μ. Ο κανόνας μετατόπισης της παραμέτρου είναι ένας υπολογισμός της μερικής παραγώγου ως προς μ με τον ακόλουθο τρόπο:

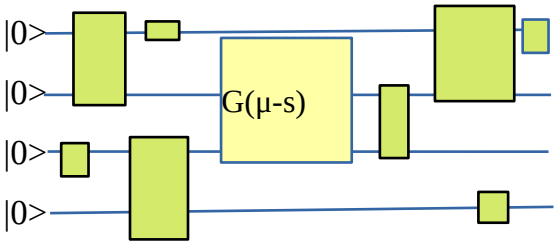
$$\partial_\mu f_\mu = \sum a_i f_{\mu + s_i}$$

όπου οι ποσότητες $\{a_i\}$ και $\{s_i\}$ είναι πραγματικοί αριθμοί. Οι 2 βασικές διαφορές με τη μέθοδο των πεπερασμένων διαφορών είναι ότι εδώ δεν έχουμε προσέγγιση αλλά ακριβή τιμή της παραγώγου, κι επίσης η ποσότητα si δεν είναι απαραίτητα απειροστά μικρή. Στην πραγματικότητα βέβαια, οι κβαντικοί υπολογιστές που έχουμε στη διάθεση μας σήμερα μπορούν απλώς να κάνουν μια εκτίμηση, μια προσέγγιση της αναμενόμενης τιμής, οπότε στην ουσία έχουμε ότι η μέθοδος της μετατόπισης παραμέτρου μας δίνει μια εκτίμηση της αναλυτικής παραγώγου, ενώ η μέθοδος πεπερασμένων διαφορών μας δίνει εκτίμηση μιας προσεγγιστικής παραγώγου.



-

$\partial_\mu M$



Σχήμα 9: parameter shift rule

Εν τέλει για να μετρήσουμε τη μερική παράγωγο του μοντέλου, αρκεί να κάνουμε 2 ξεχωριστούς υπολογισμούς στη σειρά, έναν με την παράμετρο αυξημένη κατά μια ποσότητα s, που όπως αναφέραμε μπορεί να είναι όσο μεγάλη θέλουμε, μιας και η γεωμετρία του χώρου Hilbert μας επιτρέπει να επιλέξουμε το s όσο μεγάλο χρειάζεται και έπειτα κάνουμε έναν ακόμα υπολογισμό αυτή τη φορά με την παράμετρο μειωμένη κατά την ίδια ποσότητα s, όπως και προηγουμένως και αυτή η διαφορά στους 2 υπολογισμούς μας δίνει ακριβή εικόνα για την παράγωγο και όχι προσέγγιση όπως οι πεπερασμένες διαφορές.

Barren Plateaus

Η διαδικασία που περιγράψαμε παραπάνω μας δίνει όπως είδαμε να βελτιστοποιούμε τις παραμέτρους σε ένα παραμετροποιημένο κβαντικό κύκλωμα, χρησιμοποιώντας εργαλεία προερχόμενα ως έμπνευση από την Κλασική Μηχανική Μάθηση, όμως κανείς δεν μας εγγυάται ότι ο χώρος των παραμέτρων μπορεί να μην είναι κατάλληλος για εκπαίδευση. Τα barren plateaus είναι τμήματα του χώρου των παραμέτρων θ που η παράγωγος είναι 0, ή πιο σωστά είναι με μεγάλη πιθανότητα ίση με 0 [19]. Κάτι που σημαίνει ότι η βελτιστοποίηση και κατά συνέπεια η εκπαίδευση είναι πολύ αργή και κοστίζει σε πόρους, διότι θέλουμε αποτελέσματα με μεγάλη ακρίβεια και χωρίς σφάλματα, κάτι που είναι έτσι κι αλλιώς δύσκολο με τις NISQ συσκευές που έχουμε προς το παρόν στη διάθεση μας, δηλαδή τα barren plateaus είναι ακόμα ένα πρόβλημα που προστίθεται στην ήδη δύσκολη βελτιστοποίηση των κβαντικών μοντέλων.

Τα barren plateaus παρατηρούνται ως φαινόμενο σε διαφορετικές περιπτώσεις μάθησης, επομένως δεν μπορούμε να είμαστε σίγουροι από που προέρχεται κάθε φορά. Μπορεί να είναι

θέμα της αρχιτεκτονικής του κβαντικού μοντέλου, επίσης μπορεί να είναι θέμα του μεγάλου χώρου Hilbert (ο οποίος είναι εκ φύσεως εκθετικά μεγάλος διανυσματικός χώρος). Ας θεωρήσουμε για παράδειγμα ένα διάνυσμα $\bar{v}$ σε έναν τέτοιο χώρο, που επιδρούμε πάνω του με κάποια περιστροφή κατά γωνία θ και μετά κάνουμε διάφορες περιστροφές κατά θ ως προς όλες τις διαθέσιμες διαστάσεις του χώρου. Αυτό έχει ως αποτέλεσμα η επίδραση που έχει η παράμετρος θ στη διαδικασία της παρατήρησης να είναι απειροελάχιστη και ουσιαστικά η συνάρτηση κόστους να είναι οριακά ελεύθερη παραμέτρων. Για να το εκφράσουμε και πιο παραστατικά, αν πρέπει να περπατήσουμε σε έναν τεράστιο χώρο και πρέπει να διανύσουμε πχ μια απόσταση d, το κάθε ένα μικρό βήμα προς οποιαδήποτε κατεύθυνση δε μας δίνει ιδιαίτερα καμία πληροφορία για το που θα βρισκόμαστε στο τέλος.

Υπάρχουν μερικές προτάσεις για το πώς μπορούμε να αποφεύγουμε τέτοια προβλήματα όπως τα barren plateaus, αλλά όπως όλη σχεδόν η βιβλιογραφία της Κβαντικής Μηχανικής Μάθησης, σε γενικές γραμμές παραμένει ένα ανοιχτό ερώτημα.

Ένας τρόπος είναι να κατασκευάσουμε μια δομή μέσα στο κύκλωμα με τον εξής τρόπο: αντί να έχουμε όλες τις παραμέτρους τυχαία ορισμένες, σταθεροποιούμε μερικές από αυτές και αφήνουμε τις υπόλοιπες να παίρνουν τιμές τυχαία, και αυτό σύμφωνα με το [20] αποφεύγει τη δημιουργία των barren plateaus με κάποιο τρόπο. Ακόμα, στο [21] μας δίνουν μία γεύση για το ποια είναι η συχνότητα εμφάνισης barren plateaus στο μοντέλο μας και στη διαδικασία βελτιστοποίησης, κάτι που σύμφωνα με την έρευνα εξαρτάται σε μεγάλο βαθμό από τη συνάρτηση κόστους που χρησιμοποιούμε κάθε φορά. Αλλιώς, ο τρόπος με τον οποίο κάνουμε την παρατήρηση, επηρεάζει σημαντικά το πως και αν θα εμφανιστεί barren plateau.

Μία ακόμη τεχνική για να αποφεύγουμε τα barren plateaus είναι αυτή που αναφέρεται στο [22] και προσπαθεί να αναλύσει το χώρο στον οποίο βρίσκεται η συνάρτηση κόστους (lost landscape) και το πως μοιάζει στην ουσία η γραφική της παράσταση. Όπως ξέρουμε, ο σκοπός μας είναι να ελαχιστοποιήσουμε τη συνάρτηση κόστους (στη βιβλιογραφία αναφέρεται και ως συνάρτηση απώλειας, εξ'ου και ο όρος lost landscape), δηλαδή να βρούμε το ολικό ελάχιστο μιας συνάρτησης που σε ορισμένες περιπτώσεις μπορεί να είναι και μεγάλης διάστασης, με πολλές μεταβλητές και άρα να μην έχουμε εποπτεία γεωμετρικά. Στο [22] προσπαθούν να εξετάσουν το πως μοιάζει αυτός ο χώρος, αυτή η επιφάνεια πιθανώς που είναι η σχηματική απεικόνιση της συνάρτησης κόστους, οπότε χρησιμοποιούν τον Εσσιανό πίνακα, που είναι η

2η παράγωγος μιας συνάρτησης πολλών μεταβλητών. Έστω ότι η συνάρτηση C(θ) έχει ως παράγωγο:

$$\frac{\partial C}{\partial \theta}=\left(\frac{(\partial C)}{(\partial \theta_1)},\frac{(\partial C)}{(\partial \theta_1)},.....\right)$$

τότε η 2η παράγωγος της C είναι ο εσσιανός πίνακας που έχει την ακόλουθη μορφή:

$$\mathbf{H}=\begin{matrix}\frac{(\partial^2 C)}{(\partial \theta_1^2)} & \frac{(\partial^2 C)}{(\partial \theta_1 \theta_2)} & ... \\ \frac{(\partial^2 C)}{(\partial \theta_2 \theta_1)} & ... & ... \\ ... & ... & ...\end{matrix}$$

ένας πίνακας με όλες τις μερικές παραγώγους 2ης τάξης ως προς κάθε παράμετρο. Επομένως στο [22] ισχυρίζονται πως αντί να κάνουμε τεχνικές που βασίζονται στη βελτιστοποίηση με χρήση της 1ης παραγώγου (gradient-based techniques), ας δοκιμάσουμε να χρησιμοποιήσουμε τη 2η παράγωγο για να εξερευνήσουμε την επιφάνεια της συνάρτησης κόστους. Κι έτσι θα βρούμε το ελάχιστο και θα αποφύγουμε τυχόν barren plateaus. Διαισθητικά, και με βάση τη θεωρία των επιφανειών, η 2η παράγωγος μας δίνει ένα μέτρο για την καμπυλότητα του χώρου στον οποίο εξερευνούμε, οπότε μας δείχνει κατά πόσο απέχει αυτή η επιφάνεια από το να είναι επίπεδο (καμπυλότητα=0) και μας δείχνει πόσα σημεία είναι με κάποια καμπυλότητα, πόσα “εξογκώματα” έχει για να αναζητήσουμε εκεί πιθανό ολικό ακρότατο.

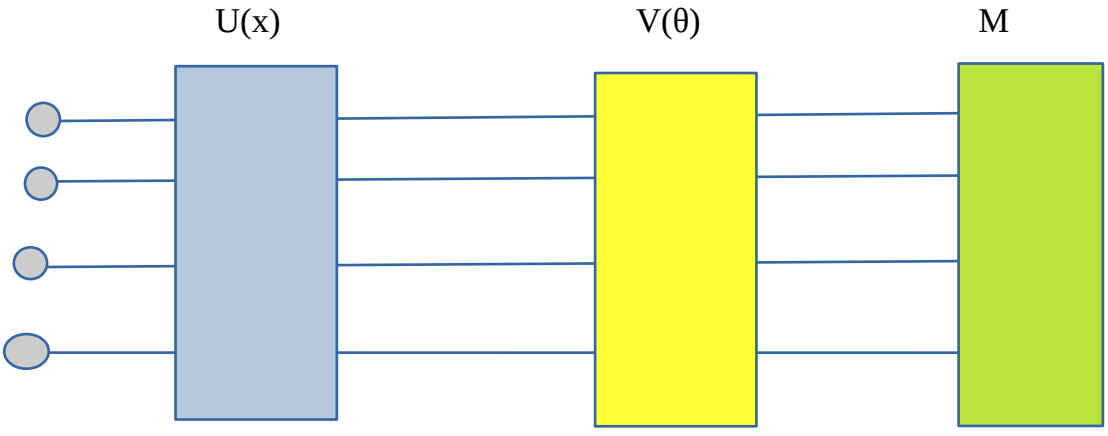
5 ΚΒΑΝΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Εδώ θα αναφέρουμε επιγραμματικά μερικούς από τους πιο βασικούς αλγορίθμους που έχουν αναπτυχθεί για να μας βοηθήσουν να κάνουμε κβαντική μηχανική μάθηση, να εκπαιδεύσουμε δηλαδή κβαντικά κυκλώματα. Κάποιοι από αυτούς έχουν δημιουργηθεί από την αρχή ως κομμάτι της έρευνας πάνω στους κβαντικούς υπολογιστές και βασίζονται στα variational models όπως για παράδειγμα οι Variational Quantum Classifier(VQC) [41], ο Variational Quantum Eigensolver (VQE) [42], ο Quantum Approximation Optimization Algorithm (QAOA) [43], ο Quadratic Unconstrained Binary Optimization (QUBO) [44] κ.α. ενώ άλλοι είναι η μεταφορά των κλασικών αλγορίθμων μάθησης και η προσαρμογή τους σε κβαντικό

περιβάλλον όπως π.χ. οι Quantum Support Vector Machine (QSVM) [45], Quantum Principal Component Analysis (QPCA) [46], κ.α..

Variational Quantum Classifier

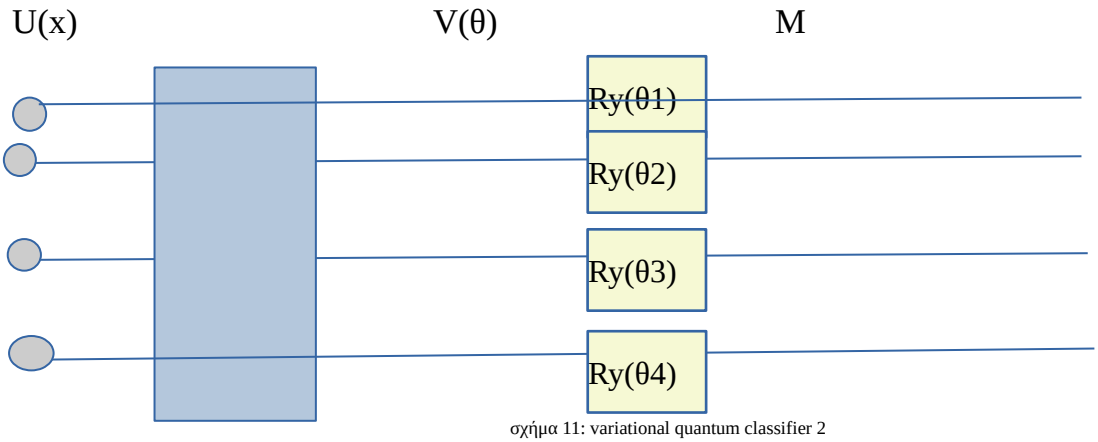
Θα δούμε για λόγους οικονομίας, αναλυτικά, μόνο τον παραπάνω αλγόριθμο [36,41]. Ξεκινάμε λοιπόν με το να ορίσουμε τον VQC. Ο αλγόριθμος αυτός, όπως σχεδόν υποδηλώνει το όνομα, κατηγοριοποιεί δεδομένα που βρίσκονται ήδη σε χώρο Hilbert, είναι δηλαδή κβαντικές καταστάσεις είτε έχουν απεικονιστεί εκεί χάρη σε κάποιον feature map και ο αλγόριθμος ουσιαστικά εκπαιδεύει ένα μοντέλο χρησιμοποιώντας ένα κβαντικό κύκλωμα με παραμέτρους (γενικότερα, οι όροι variational circuit, parameterized quantum circuit ακόμα και ο όρος ansatz χρησιμοποιούνται ισοδύναμα στη βιβλιογραφία και εννοούμε σε κάθε περίπτωση το κβαντικό κύκλωμα που έχει παραμέτρους και άρα μπορεί να εκπαιδευτεί) και ταξινομεί τα αποτελέσματα σε 2 ή περισσότερες κλάσεις. Κάτι που διαισθητικά μοιάζει με ειδική περίπτωση του αλγορίθμου των μηχανών διανυσμάτων υποστήριξης, όπου εκεί σε μερικά προβλήματα χρειάζεται να χρησιμοποιηθεί feature map για να μεταβούμε πχ από το επίπεδο στο χώρο και μετά διαχωρίζουμε τα δεδομένα με επίπεδο. Σχηματικά θα λέγαμε πως δεν έχει καμία ιδιαίτερη διαφορά με το μοντέλο που είδαμε παραπάνω όταν περιγράφαμε γενικώς τα variational models:



Σχήμα 10: variational quantum classifier 1

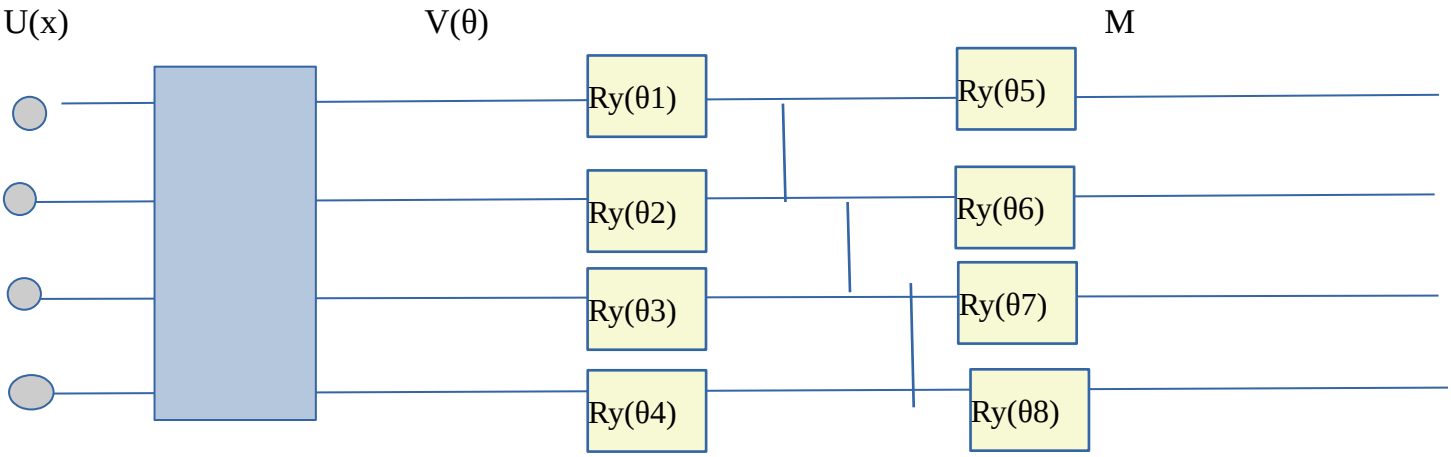
Έχουμε αρχικά έναν feature map που μετατρέπει τα δεδομένα σε διανύσματα του χώρου Hilbert, μετά κάποιες πράξεις που είναι στην ουσία γινόμενα με πύλες που αυτές περιέχουν την παράμετρο θ και τέλος υπάρχει η παρατήρηση των δεδομένων. Μετά από όλες αυτές τις διεργασίες χρησιμοποιούμε μια κλασική συνάρτηση η οποία μας δίνει την κλάση στην οποία ανήκουν τα δεδομένα, μας δίνει με άλλα λόγια μια ετικέτα την οποία θα πρέπει να έχει π.χ. η εικόνα μας που είναι το δεδομένο εισόδου και η ετικέτα αυτή μπορεί να είναι το τί περιέχει η εικόνα αυτή, ομοίως με την κλασική περίπτωση.

Έχει αρκετό ενδιαφέρον να δούμε πως λειτουργεί το 2ο σκέλος, αυτό των πράξεων με τις πύλες παραμέτρου θ. Είπαμε γενικώς ότι οι πράξεις αυτές είναι περιστροφές των διανυσμάτων των κβαντικών καταστάσεων στο χώρο Hilbert. Ας δούμε λίγο πιο αναλυτικά τι συμβαίνει εδώ:



σχήμα 11: variational quantum classifier 2

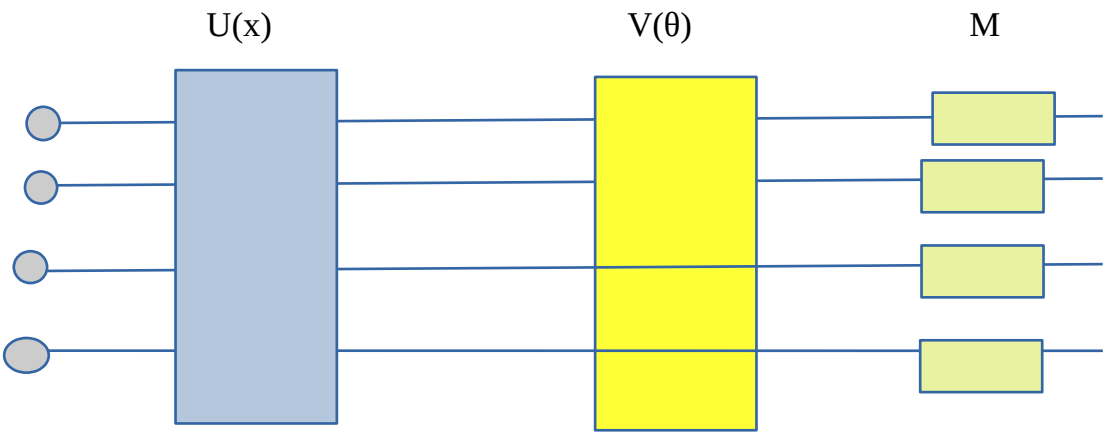
Έστω πάλι το μοντέλο που είδαμε προηγουμένως, ελαφρώς διαφοροποιημένο. Οι πράξεις λοιπόν στην περίπτωση αυτού του αλγορίθμου είναι κάποιες περιστροφές (rotations) σε όλα τα qubits κατά κάποια γωνία, διαφορετική για κάθε qubit, η οποία γωνία δεν είναι γνωστή εξ' αρχής και αυτό ακριβώς ψάχνει το μοντέλο στην ουσία, να βρει την κατάλληλη γωνία περιστροφής που εξαρτάται από την παράμετρο θ_i. Το επόμενο βήμα είναι να εφαρμοστούν πύλες που προκαλούν διεμπλοκή μεταξύ 2 qubit κάθε φορά,όπως φαίνεται παρακάτω:



Σχήμα 12: variational quantum classifier 3

Η διεμπλοκή (entaglement) θυμίζουμε είναι η πράξη εκείνη που επιδρά σε 2 qubits και από την Κβαντική Μηχανική γνωρίζουμε πως αυτά τα 2 qubits ή γενικότερα 2 σωματίδια που έχουν διεμπλακεί συμπεριφέρονται ως ένα, ή πιο σωστά περιγράφονται από την ίδια κυματοσυνάρτηση. Έπειτα εφαρμόζουμε και πάλι κάποιες περιστροφές με διαφορετικές γωνίες από το προηγούμενο στάδιο. Εάν θέλουμε να βάλουμε κι άλλες παραμέτρους στο κύκλωμα, τότε απλώς εφαρμόζουμε το κομμάτι του σχήματος που περιέχει την διεμπλοκή και τις περιστροφές μερικές φορές ακόμα, αναλόγως με το πόσες παραμέτρους θέλουμε. Δηλαδή πρώτα διεμπλέκουμε τα qubits και μετά εφαρμόζουμε περιστροφές με παράμετρο κατά άγνωστη γωνία όσες φορές χρειαστεί. Όσο βέβαια αυξάνονται οι παράμετροι και κατά συνέπεια το βάθος, θα έχουμε από τη μία μεριά πιο μεγάλη εκφραστικότητα, το μοντέλο θα γίνει πιο περιγραφικό, από την άλλη μεριά όμως, όπως είναι φυσικό θα αυξηθεί η πολυπλοκότητα. Αυτή η διαδικασία μοιάζει με τα νευρωνικά δίκτυα από την κλασική περίπτωση, διότι μπορούμε να αντιστοιχίσουμε τα κρυφά επίπεδα (hidden layers) των βαθιών νευρωνικών δικτύων (deep neural networks) με το κομμάτι των πράξεων που περιέχουν τις παραμέτρους και που αναφέραμε ότι επαναλαμβάνεται. Όμως εδώ όπως προείπαμε δεν έχουμε μη γραμμικές πράξεις πχ με συναρτήσεις ενεργοποίησης, στο κομμάτι των κβαντικών πυλών κάνουμε πράξεις με πίνακες και διανύσματα πολλές φορές, οπότε θα λέγαμε πως αυτό το μοντέλο θα μπορούσε να θεωρηθεί ως μια εκδοχή ενός γραμμικού νευρωνικού δικτύου. Αυτή η μη γραμμικότητα των νευρωνικών δικτύων είναι που τα έχει καταστήσει τόσο ισχυρό μοντέλο, διότι υπάρχει και το θεώρημα καθολικής προσέγγισης [51] που εγγυάται ότι κάθε πρόβλημα

της πραγματικής ζωής που μπορεί να εκφραστεί ως συνάρτηση, μπορεί να προσεγγιστεί όσο καλά θέλουμε από ένα νευρωνικό δίκτυο, κάτι που στην κβαντική περίπτωση δε συμβαίνει.
Το τελευταίο σκέλος του αλγορίθμου είναι το κομμάτι της παρατήρησης των αποτελεσμάτων.



Σχήμα 13: variational quantum classifier 4

Όταν κάνουμε παρατήρηση σε κάθε qubit σε αυτόν τον αλγόριθμο, μπορούμε να κάνουμε το εξής: έστω ότι έχουμε έναν σύνθετο τελεστή που κάνει τις μετρήσεις, τότε μπορούμε να κατασκευάσουμε ένα σύνθετο κβαντικό κύκλωμα το οποίο θα περιέχει έναν απλούστερο τελεστή παρατήρησης, ο οποίος θα βασίζεται στον αρχικό σύνθετο τελεστή.
Το γεγονός ότι έχουμε ένα variational model μας δίνει την ασφάλεια πως ό,τι κι αν χρειαστεί για το στάδιο της παρατήρησης, θα είναι διαθέσιμο μέσω της διαδικασίας μάθησης των παραμέτρων. Με κάποιο τρόπο εξασφαλίζεται ότι θα έχουμε και τον όποιο τελεστή χρειαστούμε για να κάνει τις παρατηρήσεις αργότερα. Οπότε όπως και να έχει, παρατηρούμε το τι μας έδωσε το variational model στο τέλος, κάθε qubit ξεχωριστά. Άρα αν επαναλάβουμε τις πράξεις του μοντέλου αρκετές φορές, παίρνουμε μια κατανομή πιθανοτήτων από πιθανές βασικές κβαντικές καταστάσεις. Και όταν αυτές είναι διαθέσιμες, συσχετίζουμε τα αποτελέσματα με μια κλασική συνάρτηση που μας δίνει περιγραφή των δεδομένων μας. Με άλλα λόγια αντιστοιχούμε την κατανομή πιθανοτήτων των βασικών καταστάσεων που προέκυψαν από το κύκλωμα με μια κατανομή ετικετών που μας δίνει αυτή η συνάρτηση στο τέλος. Η συνάρτηση κατατάσσει τα qubits που προέκυψαν από την εκπαίδευση του μοντέλου σε κάποιες κλάσεις (είτε μιλάμε για δυαδική ταξινόμηση είτε για πολλαπλή) με κάποια πιθανότητα, μας δείχνει δηλαδή πού είναι πιο πιθανό να ανήκει το qubit που μετρήσαμε.

5.1 Παραδείγματα κώδικα Κβαντικής Μηχανικής Μάθησης

Εδώ θα δούμε μερικά παραδείγματα κβαντικής μηχανικής μάθησης από τα βασικά εργαλεία που μας παρέχουν εταιρείες όπως η Google, η IBM και η Xanadu. Τα εργαλεία αυτά είναι 1) η βιβλιοθήκη Cirq της Google για κβαντικό προγραμματισμό, το οποίο σε συνεργασία με το Tensorflow Quantum μπορούν και πραγματοποιούν μάθηση σε υβριδικά μοντέλα, με το βασικό κομμάτι των πράξεων να πραγματοποιείται κβαντικά και η βελτιστοποίηση γίνεται σε κλασικό επίπεδο 2) η βιβλιοθήκη Qiskit, το αντίστοιχο εργαλείο της IBM για κβαντικά κυκλώματα (να σημειωθεί ότι και οι 2 βιβλιοθήκες είναι στη γλώσσα python) η οποία σε συνδυασμό με τη γνωστή βιβλιοθήκη PyTorch πραγματοποιούν υβριδική μάθηση ακριβώς αντίστοιχα με την Google (θα δούμε αναλυτικό παράδειγμα) και 3) το πακέτο Pennylane της Xanadu για κβαντικό προγραμματισμό και μάθηση, το οποίο από μόνο του κάνει και τα 2 και μάλιστα συνεργάζεται με όποια άλλη βιβλιοθήκη κβαντικού προγραμματισμού και μάθησης που υπάρχει στην αγορά.

Cirq

Ας δούμε το πρώτο εργαλείο κβαντικού προγραμματισμού που θα χρησιμοποιήσουμε. Η cirq λοιπόν είναι μια βιβλιοθήκη της python με την οποία μπορούμε να κατασκευάζουμε, να βελτιστοποιούμε και να τρέχουμε κβαντικά κυκλώματα σε κβαντικούς υπολογιστές της σημερινής τεχνολογίας, τους επονομαζόμενους και NISQ (Noisy Intermediate Scale Quantum) ή προσομοιωτές. Δημιουργήθηκε από την ερευνητική ομάδα της Google, QuantumAI Team και παρουσιάστηκε το 2018. Τα προγράμματα στην Cirq χρησιμοποιούν μια κλάση που ονομάζεται Circuit για την κατασκευή κβαντικών κυκλωμάτων, η οποία στην ουσία είναι μια συλλογή από στιγμές (Moments) η οποία στιγμή με τη σειρά της είναι μια σειρά από κβαντικές πύλες που επιδρούν στα qubits την ίδια χρονική στιγμή. Αλλά ας δούμε ένα πρώτο παράδειγμα κατασκευής κβαντικού κυκλώματος χρησιμοποιώντας την Cirq.

1) Παράδειγμα απλού κβαντικού κυκλώματος με cirq

Κάνουμε αρχικά τα απαραίτητα imports (χρησιμοποιήθηκε το Google Colab) και τις απαραίτητες εγκαταστάσεις [23]:

```
! pip install --quiet cirq
import cirq
```

Μετά κατασκευάζουμε ένα κβαντικό κύκλωμα χρησιμοποιώντας την κλάση Circuit

```
mycircuit=cirq.Circuit()
```

το οποίο θέλουμε να περιέχει 6 qubits, οπότε γράφουμε:

```
mycircuit.append(cirq.H(myqubits[0]))
mycircuit.append(cirq.H(myqubits[1]))
mycircuit.append(cirq.H(myqubits[2]))
mycircuit.append(cirq.H(myqubits[3]))
mycircuit.append(cirq.H(myqubits[4]))
mycircuit.append(cirq.H(myqubits[5]))
```

και θέλουμε να τα φέρουμε σε κατάσταση υπέρθεσης, οπότε:

```
mycircuit=cirq.Circuit([cirq.H(qubit) for qubit in myqubits])
```

κι άρα εχουμε το σχήμα:

```
0: ———H———
1: ———H———
2: ———H———
3: ———H———
4: ———H———
5: ———H———
```

Θα μπορούσαμε να εκτελέσουμε εναλλακτικά την εντολή:

```
mycircuit=cirq.Circuit(cirq.H.on_each(myqubits))
```

και θα είχαμε το ίδιο αποτέλεσμα, με τη χρήση της on_each σε κάθε qubit, που πάλι εφαρμόζουμε την πύλη Hadamard. Στο παρόν παράδειγμα θέλουμε να κατασκευάσουμε μια συγκεκριμένη μορφή κυκλώματος:

```
Σχήμα1_cirq
0: ———————— H———@————————————H———M———
1: ————————H———┼———@————————H———M———
2: ————————H———┼———┼———@————H———M———
3: ————————H———┼———┼———┼————H———M———
```

```
4: ──────────H───┼───┼───┼───H───M───
5: ────────X───H───X───X───X───H───M───
```

Επομένως πάμε να κάνουμε όλα τα βήματα ξεχωριστά. Πρώτα θα εφαρμόσουμε μια X gate για να αλλάξουμε την κατάσταση του 6ου qubit από |0> σε |1> (να σημειωθεί πως αν δεν ορίσουμε αρχική κατάσταση, το cirq, όπως και το qiskit που θα δούμε παρακάτω, έχουν ως default κατάσταση την |0>). Οπότε κάνουμε την X-gate και μετά Hadamard:

```
mycircuit=cirq.Circuit(cirq.X(myqubits[5]))
mycircuit.append(cirq.H.on_each(myqubits))

print(mycircuit)
```

και το αποτέλεσμα:

```
0: ───H─────────
1: ───H─────────
2: ───H─────────
3: ───H─────────
4: ───H─────────
5: ───X───H─────
```

Για να δούμε και πως λειτουργούν οι moments στην cirq, ας δούμε το παρακάτω: Κατασκευάζουμε 2 ξεχωριστές στιγμές, μία για την πύλη X στο 6ο qubit και μία ακόμη για την πύλη Hadamard σε όλα τα qubits.

```
moment0= cirq.Circuit(cirq.X(myqubits[5]))
moment1= cirq.Circuit(cirq.H.on_each(myqubits))
```

αν τρέξουμε την print για το πρώτο σκέλος θα έχουμε:

```
5: ───X───
```

ενώ με τη δεύτερη στιγμή θα έχουμε:

```
0: ───H───
1: ───H───
2: ───H───
3: ───H───
```

```
4: ———H———
5: ———H——
```

Βλέπουμε δηλαδή πως οι στιγμές είναι οι πράξεις με τελεστές που συνέβησαν στον ίδιο χρόνο, η πρώτη στιγμή μας δείχνει μόνο την αλλαγή κατάστασης από 0 σε 1 στο 6ο qubit ενώ η δεύτερη στιγμή μας δείχνει μόνο τις πύλες Hadamard σε όλα τα qubits χωρίς κάποια αναφορά στην X gate που προηγήθηκε χρονικά. Για να φτάσουμε στο σχήμα που περιγράψαμε (σχήμα1_cirq) ακολουθούμε τα παρακάτω βήματα:

Πρώτα ορίζουμε όλο το κύκλωμα από την αρχή:

```
mycircuit=cirq.Circuit()
```

```
myqubits=cirq.LineQubit.range(6)
```

Κατασκευάζουμε ένα κύκλωμα με 6 qubits ξανά, τα θέτουμε στην αρχική κατάσταση |0>, κατασκευάζουμε επίσης στιγμές για κάθε τι που κάνουμε, όπως π.χ. η αλλαγή του τελευταίου qubit από 0 σε 1 και η εφαρμογή της Hadamard σε όλα τα qubits:

```
reset_all_qubits=cirq.Moment([cirq.reset(qubit) for qubit in myqubits])
```

```
mycircuit.append(reset_all_qubits)
```

```
setup_detect_qubit=cirq.X(myqubits[-1])
mycircuit.append(setup_detect_qubit)
```

```
H_gates=cirq.Moment(cirq.H.on_each(myqubits))
mycircuit.append(H_gates)
```

Έπειτα βεβαίως θα εφαρμοστεί η πύλη CNOT, που όπως ήδη αναφέραμε χρειάζεται 2 qubits για να λειτουργήσει, εδώ θα χρειαστούμε την CNOT να επιδράσει στα 0-5, 1-5 και 2-5:

```
mycircuit.append([
            cirq.CNOT(myqubits[0] , myqubits[5]),
            cirq.CNOT(myqubits[1] , myqubits[5]),
            cirq.CNOT(myqubits[2] , myqubits[5]),
])
```

```
mycircuit.append(H_gates)
```

```
print(mycircuit)
```

με το αποτέλεσμα στην οθόνη να είναι το:

```
0: ———R———————H———@———————————H———
```

```
| 1: ———R———————H———|———@———————H———
| 2: ———R———————H———|———|———@———H———
| 3: ———R———————H———|———|———|———H———
| 4: ———R———————H———|———|———|———H———
| 5: ———R———X———H———X———X———X———H———
```

Το τελευταίο που έμεινε να κάνουμε είναι να παρατηρηθούν τα αποτελέσματα και να χρησιμοποιήσουμε έναν προσομοιωτή για να δούμε τί θα υπολογίσει:

```
meas_gates=cirq.Moment([cirq.measure(qubit) for qubit in myqubits])
```

```
mycircuit.append(meas_gates)
print(mycircuit)
```

και έχουμε:

```
  0: ———R———————H———@———————————H———M———
| 1: ———R———————H———|———@———————H———M———
| 2: ———R———————H———|———|———@———H———M———
| 3: ———R———————H———|———|———|———H———M———
| 4: ———R———————H———|———|———|———H———M———
| 5: ———R———X———H———X———X———X———H———M———
```

ο προσομοιωτής είναι από τους βασικούς που χρησιμοποιεί η cirq:

```
sim=cirq.Simulator()
```

```
result= sim.run(mycircuit)
```

```
print(result)
```

και μας δίνει:

```
q(0)=1 q(1)=1 q(2)=1 q(3)=0 q(4)=0 q(5)=1
```

αυτό που περιμέναμε,με το 1ο, το 2ο, το 3ο και το 6ο να έχουν αλλάξει από 0 σε 1 και τα άλλα δύο (το 4ο και το 5ο) να μην έχουν επηρεαστεί.

Pennylane

Η pennylane είναι μια βιβλιοθήκη της python για κβαντικό προγραμματισμό και κβαντική μηχανική μάθηση, η οποία κατασκευάστηκε από την ερευνητική εταιρεία Xanadu Quantum Technologies με έδρα τον Καναδά. Όπως λένε οι κατασκευαστές του, είναι ένα λογισμικό

ιδανικό για διαφορίσιμο προγραμματισμό (differentiable programming) ένα ιδιαίτερα ενδιαφέρον προγραμματιστικό παράδειγμα, το οποίο θα περιγράψουμε σε λίγες γραμμές.

Η ιδέα του διαφορίσιμου προγραμματισμού γεννήθηκε από τη Βαθιά Μάθηση (Deep Learning). Η βαθιά μάθηση όπως γνωρίζουμε χρησιμοποιεί νευρωνικά δίκτυα ως μοντέλα για να επιτελεί τις διάφορες διεργασίες της. Θα μπορούσαμε να πούμε με 2 λόγια πως όλη η ιδέα του deep learning είναι να κατασκευάσουμε μοντέλα μηχανικής μάθησης χρησιμοποιώντας διαφορίσιμες συναρτήσεις (ή πιο σωστά, συνθέσεις συναρτήσεων). Ο διαφορίσιμος προγραμματισμός επεκτείνει αυτή την ιδέα και αναφέρεται σε διαφορισιμότητα όχι μόνο των μοντέλων μηχανικής μάθησης αλλά σε ολόκληρα κομμάτια κώδικα! Ο κώδικας προφανώς θα πρέπει να περιέχει αριθμητικές τιμές ή σε κάθε περίπτωση να έχει στοιχεία που μπορούν να διαφοριστούν, όμως και πάλι είναι κάτι το διαφορετικό αν μη τι άλλο. Οι υπεύθυνοι της Xanadu αναφέρουν πως στη δική τους περίπτωση, χάρη στο διαφορίσιμο προγραμματισμό

εκπαιδεύεται ένας κβαντικός υπολογιστής όπως θα εκπαιδευόταν ένα κλασικό νευρωνικό δίκτυο .

Για το pennylane τώρα, πρόκειται για ευέλικτο λογισμικό, μπορεί να τρέξει σε διαφορετικά περιβάλλοντα, χρησιμοποιεί automatic differentiation για να κατασκευάζει κβαντικά κυκλώματα, συνδέει βιβλιοθήκες όπως το tensorflow, το pytorch και τη numpy με κβαντικές συσκευές και προσομοιωτές κατασκευάζοντας υβριδικά μοντέλα και γενικότερα είναι έυχρηστο [40] . Ας δούμε ένα παράδειγμα για να καταλάβουμε πώς λειτουργεί.

2) Πρώτο Παράδειγμα με pennylane

```
import pennylane as qml
from pennylane import numpy as np
# In[8]:
dev=qml.device("default.qubit" , wires=2)
# In[10]:
@qml.qnode(dev)
def circuit(theta):
  qml.PauliX(wires=1)
  qml.CNOT(wires=[1,0])
  qml.RY(theta , wires=0)
```

```
    return qml.expval(qml.PauliZ(wires=0))
# In[11]:
print(circuit(np.pi))
# In[13]:
thetas=np.arange(-np.pi , np.pi  ,0.01)
# In[14]:
measurements=np.zeros(len(thetas))
# In[15]:
for i,theta in enumerate(thetas):
    measurements[i]=circuit(theta)
# In[16]:
plt.plot(thetas , measurements)
plt.show
```

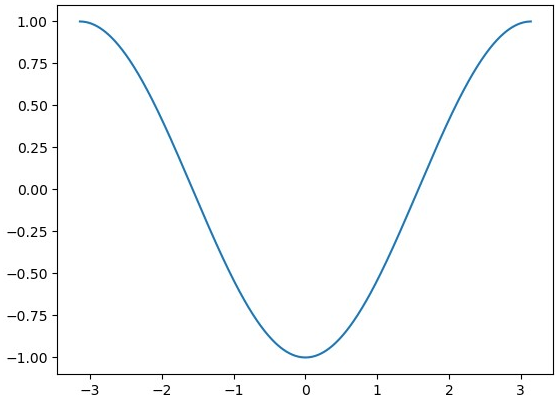
Σε ένα πρώτο παράδειγμα (paradeigma1.py) βλέπουμε την κατασκευή ενός απλού κβαντικού κυκλώματος με τη βοήθεια του λογισμικού pennylane [24]. Αφού κάνουμε τα βασικά imports που χρειάζονται, των βιβλιοθηκών pennylane και numpy, προχωρούμε στα 2 βασικά στοιχεία για τη δημιουργία ενός κβαντικού κυκλώματος στο pennylane που είναι η συσκευή στην οποία θα τρέξουμε τον κώδικα και μια κβαντική συνάρτηση. Η συσκευή στην ουσία είναι ένα αντικείμενο που μπορούμε να εκτελέσουμε κβαντικές πράξεις και να μας επιστρέφει τα αποτελέσματα της μέτρησης σε κλασική μορφή. Η συσκευή αυτή μπορεί να είναι μια πραγματική κβαντική συσκευή με την οποία προφανώς συνδεόμαστε απομακρυσμένα, είτε κάποιος προσομοιωτής. Γενικώς τέτοιες συσκευές έχουν υπολογιστική ισχύ μερικών μόλις qubits και σε κάποιες περιπτώσεις όπως της IBM και της Xanadu, η χρήση είναι δωρεάν, ενώ στην περίπτωση της Google και της Microsoft δεν υπάρχει η δυνατότητα χρήσης των δικών τους κβαντικών υπολογιστών. Αρχικά ορίζουμε τον προσομοιωτή στον οποίο θα τρέξει ο κώδικας μας. Η συσκευή ονομάζεται default.qubit και έχουμε πρόσβαση σε 2 qubits. Το επόμενο βήμα είναι να ορίσουμε μια κβαντική συνάρτηση, που δίνει στη συσκευή τις εντολές που χρειάζεται για να τρέξουμε τις πράξεις που θέλουμε. Η συνάρτηση που θα χρησιμοποιήσουμε είναι όμοια με κάθε συνάρτηση που γράφεται στην python.

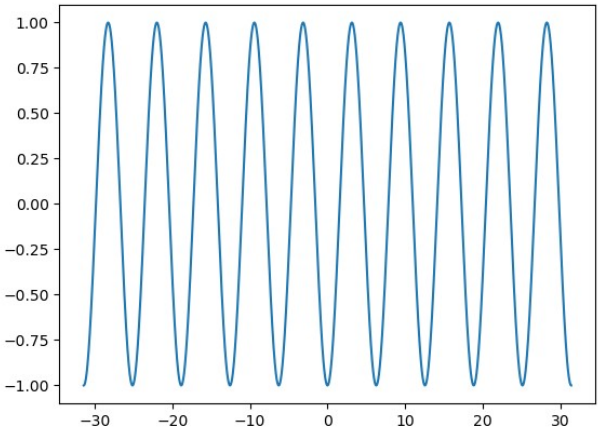
Η πρώτη πράξη είναι μια περιστροφή ως προς τον άξονα των X κατά π rad (180 μοίρες) , η δεύτερη όπως είχαμε δει είναι πράξη που χρειάζεται 2 qubits, το πρώτο είναι το qubit ελέγχου και το 2ο το qubit στόχος και η τρίτη πράξη είναι μια περιστροφή κατά τυχαία γωνία θ γύρω από τον άξονα Y του 1ου qubit. Η συνάρτηση επιστρέφει μια αναμενόμενη τιμή (expectation

value ή expval) που θα ορίσουμε εμείς και σε αυτή την περίπτωση είναι η τιμή που θα προέλθει από τον τελεστή Pauli Z που θα επιδράσει πάνω στο 1ο qubit. Για να το δούμε πιο αναλυτικά, σημαίνει πως θα υπολογιστεί μια ποσότητα της μορφής <ψ|Z|ψ>, όπου |ψ> είναι ένα διάνυσμα διάστασης 2^n που παράγεται από το κύκλωμα, με n να είναι ο αριθμός των qubits στο κύκλωμα. Ο Z είναι ένας τελεστής που επιδρά πάνω στο 1ο qubit και μόνο. Όταν λοιπόν τρέχουμε το κύκλωμα μας, για γωνία πχ θ=0.5, θα πάρουμε έναν αριθμό που αντιστοιχεί ακριβώς στην αναμενόμενη τιμή που έχει υπολογίσει η συνάρτηση.

Τώρα, για να εκτελεστούν όλες αυτές οι εντολές που γράψαμε πρέπει να ορίσουμε την έννοια του Quantum Node, που στην ουσία είναι ένα γενικότερο πλαίσιο στην Pennylane

που περιλαμβάνει την κβαντική συσκευή (τον προσομοιωτή σε αυτή την περίπτωση) και την κβαντική συνάρτηση, γι'αυτό και πριν από τη συνάρτηση έχουμε το συμβολισμό @gml.qnode. Τα υπόλοιπα βήματα είναι η εφαρμογή της συνάρτησης πάνω σε ένα σύνολο από γωνίες, με τις οποίες μπορούμε να πειραματιστούμε μέσω της συνάρτησης της python που είναι η arange και να δοκιμάσουμε το κύκλωμα σε γωνίες πχ από (-π,π) όπως κάναμε εδώ ή και οποιοδήποτε άλλο διάστημα, πχ (-10π,10π) και πάντα το γράφημα είναι μια συνημιτονοειδής συνάρτηση.





Qiskit

Το τελευταίο λογισμικό δημιουργίας κβαντικών κυκλωμάτων που θα δούμε είναι το qiskit. Είναι και αυτό,όπως και τα άλλα 2, ένα λογισμικό ανοιχτού κώδικα, που κατασκευάστηκε από την IBM Research για λογαριασμό της IBM και μας παρέχει εργαλεία για να κατασκευάσουμε και να βελτιστοποιήσουμε κβαντικά κυκλώματα. Είναι στην ουσία το αντίστοιχο εργαλείο με το Cirq της Google που είδαμε προηγουμένως. Αποτελεί και αυτό βιβλιοθήκη της Python, οπότε είναι αρκετά εύχρηστο λογισμικό με μεγάλη κοινότητα προγραμματιστών. Η διαφορά με την Google είναι η πρόσβαση που παρέχει σε πραγματικό κβαντικό υπολογιστή δωρεάν, οπότε όλο και περισσότεροι προγραμματιστές ή απλώς ενθουσιασμένοι με τον Κβαντικό Προγραμματισμό μπορούν να πειραματιστούν εξ' αποστάσεως με πραγματική κβαντική συσκευή και να τεστάρουν τα αποτελέσματα σε πραγματικές συνθήκες. Ας δούμε κι εδώ ένα απλό παράδειγμα και μετά θα πάμε και στη μάθηση.

3) παράδειγμα qiskit με fourier μετασχηματισμό

Εδώ θα δούμε ένα πάλι απλό παράδειγμα εφαρμογής ενός αλγορίθμου που στην κβαντική του εκδοχή υπερέχει του κλασικού αντίστοιχου [25]. Θα εξετάσουμε τον αν και κατά πόσο συσχετίζεται ο μετασχηματισμός fourier της συνάρτησης g με τη συνάρτηση f. Το κύκλωμα θα υπολογίσει μια πιθανότητα για την κατάσταση |00>. Αν αυτή η πιθανότητα p(f,g) είναι μεγαλύτερη ή ίση από 0.05, τότε λέμε πως ο μετασχηματισμός fourier της g συσχετίζεται με την f. Κάνουμε import αρχικά σε κάποιες βιβλιοθήκες που θα μας φανούν χρήσιμες

import qiskit.quantum_info as qi

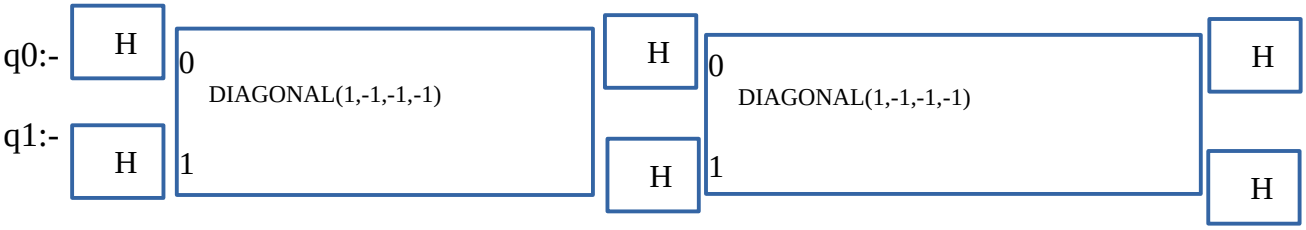
from qiskit.circuit.library import FourierChecking

from qiskit.visualization import plot_histogram

και στη συνέχεια ορίζω 2 συναρτήσεις, που στο παρόν παράδειγμα θα είναι απλώς 2 λίστες

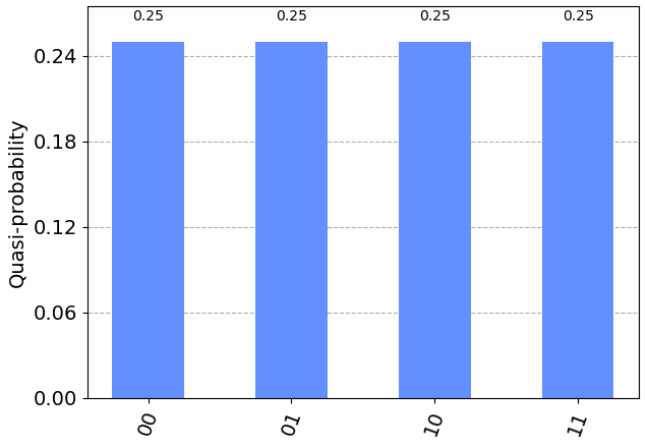
f=[1,-1,-1,-1]
g=[1,1,-1,-1]

ως προς την οπτικοποίηση του κυκλώματος,έχουμε το παρακάτω σχήμα:



Τα υπόλοιπα είναι ο υπολογισμός αυτής της πιθανότητας μέσω προσομοιωτή και η εμφάνιση του τελικού αποτελέσματος:

```
zero=qi.Statevector.from_label('00')
sv=zero.evolve(circ)
probs=sv.probabilities_dict()
plot_histogram(probs)
```



Μας ενδιαφέρει όπως αναφέραμε μόνο η τιμή για την πιθανότητα στην κατάσταση |00>,η οποία είναι όπως βλέπουμε 0.25,που είναι μεγαλύτερο από το 0.05,επομένως ο μετασχηματισμός Fourier της g σχετίζεται με την f.

4) παράδειγμα με qiskit και pytorch

Εδώ έχουμε ένα παράδειγμα υβριδικού κλασικού-κβαντικού μοντέλου, όπου ένα μέρος της μάθησης πραγματοποιείται κλασικά και το υπόλοιπο κβαντικά [26]. Για την ακρίβεια, αρχικώς γίνονται κάποιοι υπολογισμοί από κλασικούς νευρώνες και το αποτέλεσμα περνάει στο “κρυφό” επίπεδο που στην περίπτωση αυτή είναι ένα παραμετροποιημένο κβαντικό κύκλωμα. Έπειτα γίνεται παρατήρηση των αποτελεσμάτων του κβαντικού τμήματος και αυτά γίνονται είσοδος για το τελευταίο κομμάτι του μοντέλου μας που είναι κλασικό και πάλι. Όπως είδαμε

προηγουμένως αναλυτικά, θα χρησιμοποιηθεί η μέθοδος της αλλαγής παραμέτρων (parameter shift rule) αντί του backpropagation αλγορίθμου, που έχει παρόμοια ισχύ με τις τεχνικές gradient descent που ξέρουμε από την κλασική μάθηση. Στον παρόν παράδειγμα θα βάλουμε όλες τις κβαντικές συναρτήσεις που θα χρειαστούμε σε μια κλάση, θα εκμεταλλευτούμε το γεγονός ότι η python είναι και αντικειμενοστραφής. Εδώ επίσης θα χρησιμοποιήσουμε απλώς ένα κύκλωμα με ένα qubit και μια παράμετρο θ που θα χρειαστεί να βελτιστοποιήσουμε. Το κύκλωμα είναι της μορφής:

το επόμενο βήμα είναι να ορίσουμε αυτή την κλάση με ό,τι χρειαζόμαστε από κβαντικές συναρτήσεις:

```
class QuantumCircuit:
    """
Η κλάση αυτή θα μας βοηθήσει να χρησιμοποιήσουμε το κβαντικό κύκλωμα
    """

    def __init__(self, n_qubits, backend, shots):
        # --- Circuit definition ---
        self._circuit = qiskit.QuantumCircuit(n_qubits)

        all_qubits = [i for i in range(n_qubits)]
        self.theta = qiskit.circuit.Parameter('theta')

        self._circuit.h(all_qubits)
        self._circuit.barrier()
        self._circuit.ry(self.theta, all_qubits)

        self._circuit.measure_all()

        self.backend = backend
        self.shots = shots

    def run(self, thetas):
        t_qc = transpile(self._circuit,
```

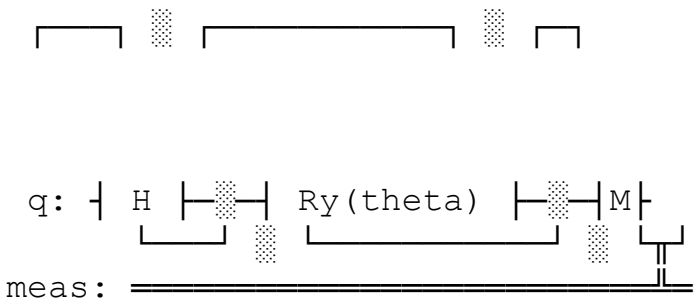
```
            self.backend)
        qobj = assemble(t_qc,
                shots=self.shots,
                parameter_binds = [{self.theta: theta} for theta in thetas])
        job = self.backend.run(qobj)
        result = job.result().get_counts()

        counts = np.array(list(result.values()))
        states = np.array(list(result.keys())).astype(float)

        # Υπολογίζουμε τις πιθανότητες για κάθε βήμα ξεχωριστά
         probabilities = counts / self.shots

        expectation = np.sum(states * probabilities)

        return np.array([expectation])
```

τεστάρουμε το πώς λειτουργεί η κλάση:

```
simulator = qiskit.Aer.get_backend('aer_simulator')

circuit = QuantumCircuit(1, simulator, 100)

print('Expected value for rotation pi {}'.format(circuit.run([np.pi])[0]))

circuit._circuit.draw()
```

και μας δίνει ως αποτέλεσμα:

```
Expected value for rotation pi 0.54
```



εννοείται πως κάθε φορά που εκτελούμε και πάλι αυτό το κομμάτι του κώδικα,μας δίνει και διαφορετική γωνία ως αποτέλεσμα.

Το επόμενο βήμα είναι από τα πιο σημαντικά,η κατασκευή της υβριδικής συνάρτησης:

```
class HybridFunction(Function):
  """ Hybrid quantum - classical function definition """

  @staticmethod
  def forward(ctx, input, quantum_circuit, shift):
    """προς τα εμπρός υπολογισμός"""
    ctx.shift = shift
    ctx.quantum_circuit = quantum_circuit

    expectation_z = ctx.quantum_circuit.run(input[0].tolist())
    result = torch.tensor([expectation_z])
    ctx.save_for_backward(input, result)

    return result

  @staticmethod
  def backward(ctx, grad_output):
    """προς τα πίσω υπολογισμός"""
    input, expectation_z = ctx.saved_tensors
    input_list = np.array(input.tolist())

    shift_right = input_list + np.ones(input_list.shape) * ctx.shift
    shift_left = input_list - np.ones(input_list.shape) * ctx.shift

    gradients = []
    for i in range(len(input_list)):
      expectation_right = ctx.quantum_circuit.run(shift_right[i])
      expectation_left  = ctx.quantum_circuit.run(shift_left[i])

                          gradient = torch.tensor([expectation_right]) -
torch.tensor([expectation_left])
      gradients.append(gradient)
    gradients = np.array([gradients]).T
     return torch.tensor([gradients]).float() * grad_output.float(), None,
None

class Hybrid(nn.Module):
  """ ορισμός του υβριδικού επιπέδου"""
```

```
    def __init__(self, backend, shots, shift):
        super(Hybrid, self).__init__()
        self.quantum_circuit = QuantumCircuit(1, backend, shots)
        self.shift = shift

    def forward(self, input):
        return HybridFunction.apply(input, self.quantum_circuit, self.shift)
```

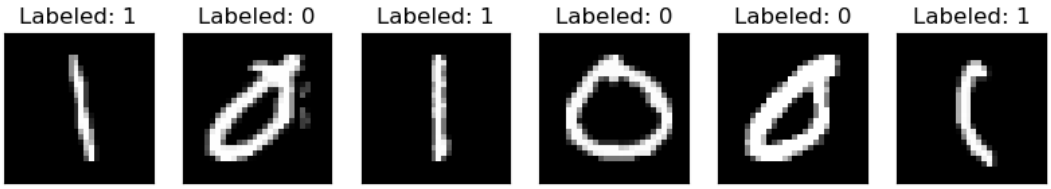
Εδώ κατασκευάσαμε 2 συναρτήσεις,που θα χρησιμοποιηθούν στην εκτέλεση του backpropagation αλγορίθμου στην κλασική περίπτωση. Μετά φορτώνουμε τα δεδομένα και κάνουμε μια πρώτη εκπαίδευση. Τα δεδομένα είναι εικόνες από τη βάση δεδομένων MNIST που περιέχουν εικόνες των ψηφίων 0 και 1

Εκπαίδευση των δεδομένων:

```
n_samples = 100

X_train = datasets.MNIST(root='./data', train=True, download=True,
                         transform=transforms.Compose([transforms.ToTensor()]))

# Leaving only labels 0 and 1
idx = np.append(np.where(X_train.targets == 0)[0][:n_samples],
                np.where(X_train.targets == 1)[0][:n_samples])

X_train.data = X_train.data[idx]
X_train.targets = X_train.targets[idx]

train_loader = torch.utils.data.DataLoader(X_train, batch_size=1, shuffle=True)


n_samples_show = 6
data_iter = iter(train_loader)
fig, axes = plt.subplots(nrows=1, ncols=n_samples_show, figsize=(10, 3))
while n_samples_show > 0:
    images, targets = data_iter.__next__()


    axes[n_samples_show - 1].imshow(images[0].numpy().squeeze(), cmap='gray')
    axes[n_samples_show - 1].set_xticks([])
    axes[n_samples_show - 1].set_yticks([])
    axes[n_samples_show - 1].set_title("Labeled: {}".format(targets.item()))
```

n_samples_show -= 1

με το αποτέλεσμα να είναι η εικόνα:



Τεστάρουμε πάλι τα δεδομένα:

n_samples = 50

```
X_test = datasets.MNIST(root='./data', train=False, download=True,
                        transform=transforms.Compose([transforms.ToTensor()]))
idx = np.append(np.where(X_test.targets == 0)[0][:n_samples],
                np.where(X_test.targets == 1)[0][:n_samples])
X_test.data = X_test.data[idx]
X_test.targets = X_test.targets[idx]
test_loader = torch.utils.data.DataLoader(X_test, batch_size=1, shuffle=True)
```

Μέχρι στιγμής η όλη δουλειά μας είναι να έχουμε τα δεδομένα μας και μια κλάση που στην ουσία είναι το κβαντικό κύκλωμα που περιέχει όπως αναφέραμε 1 παράμετρο που μπορεί να βελτιστοποιηθεί. Αυτή η παράμετρος θα είναι input για το κλασικό νευρωνικό δίκτυο μαζί με κλασικές παραμέτρους για να δημιουργήσουν το υβριδικό μοντέλο. Επίσης έχουμε κατασκευάσει τις backward και forward συναρτήσεις για να μπορέσουμε να κάνουμε την κλασική μάθηση. Τώρα αυτό που έμεινε είναι να κάνουμε και το τελικό νευρωνικό δίκτυο με τη βοήθεια του Pytorch. Το μοντέλο μας θα πρέπει να είναι συμβατό με το κβαντικό hidden layer που προηγείται, ως προς τη διάσταση, κι εφόσον έχουμε 1 κβαντική παράμετρο θ όπως αναφέραμε, οπότε θα έχουμε ουσιαστικά ένα συνελικτικό νευρωνικό δίκτυο (convolutional neural network) με 2 επίπεδα πλήρως συνδεδεμένα μεταξύ τους. Η τιμή του τελευταίου νευρώνα θα είναι η παράμετρος θ που θα είναι είσοδος στο κβαντικό κομμάτι, το οποίο με τη σειρά του όταν μετρηθεί θα μας δώσει αποτέλεσμα 0 ή 1.

class Net(nn.Module):

```
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, kernel_size=5)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5)
        self.dropout = nn.Dropout2d()
        self.fc1 = nn.Linear(256, 64)
        self.fc2 = nn.Linear(64, 1)
        self.hybrid = Hybrid(qiskit.Aer.get_backend('aer_simulator'), 100, np.pi / 2)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = self.dropout(x)
        x = x.view(1, -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.hybrid(x)
        return torch.cat((x, 1 - x), -1)
```

Και πλέον έχουμε όλα όσα χρειαζόμαστε για να κάνουμε την εκπαίδευση του πλήρους υβριδικού μοντέλου. Χρησιμοποιούμε τον adam optimizer και learning rate 0.001.

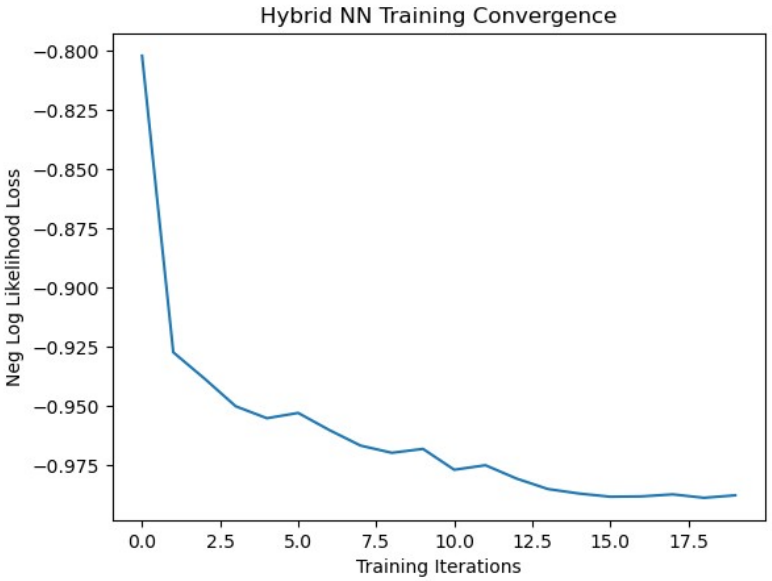
```
model = Net()
optimizer = optim.Adam(model.parameters(), lr=0.001)
loss_func = nn.NLLLoss()
epochs = 20
loss_list = []
model.train()
for epoch in range(epochs):
```

```
    total_loss = []
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = model(data)
         loss = loss_func(output, target)

        loss.backward()
        optimizer.step()

        total_loss.append(loss.item())
    loss_list.append(sum(total_loss)/len(total_loss))
    print('Training [{:.0f}%]\tLoss: {:.4f}'.format(
        100. * (epoch + 1) / epochs, loss_list[-1]))
```

με τα αποτελέσματα να είναι τα ακόλουθα:

```
Training [5%]   Loss: -0.8021
Training [10%]  Loss: -0.9274
Training [15%]  Loss: -0.9384
Training [20%]  Loss: -0.9501
Training [25%]  Loss: -0.9552
Training [30%]  Loss: -0.9529
Training [35%]  Loss: -0.9602
Training [40%]  Loss: -0.9668
Training [45%]  Loss: -0.9698
Training [50%]  Loss: -0.9681
Training [55%]  Loss: -0.9770
Training [60%]  Loss: -0.9750
Training [65%]  Loss: -0.9807
Training [70%]  Loss: -0.9851
Training [75%]  Loss: -0.9870
Training [80%]  Loss: -0.9883
Training [85%]  Loss: -0.9882
Training [90%]  Loss: -0.9874
Training [95%]  Loss: -0.9888
Training [100%] Loss: -0.9878
```

Στο γράφημα βλέπουμε πώς μειώνεται η απώλεια μετά από πολλές επαναλήψεις:



Και τέλος αυτό που έχουμε να κάνουμε είναι να τεστάρουμε το μοντέλο συνολικά:

```
model.eval()
with torch.no_grad():

    correct = 0
    for batch_idx, (data, target) in enumerate(test_loader):
        output = model(data)


        pred = output.argmax(dim=1, keepdim=True)
        correct += pred.eq(target.view_as(pred)).sum().item()
        loss = loss_func(output, target)
        total_loss.append(loss.item())
    print('Performance on test data:\n\tLoss: {:.4f}\n\tAccuracy: {:.1f}%'.format(
        sum(total_loss) / len(total_loss),
        correct / len(test_loader) * 100)
        )
```

Με το αποτέλεσμα να είναι:

```
Performance on test data:
        Loss: -0.9742
        Accuracy: 100.0%
```

5) Παράδειγμα με tensorflow quantum

Εδώ θα δούμε μια εφαρμογή του tensorflow quantum [27,38], ένα παράδειγμα κβαντικού νευρωνικού δικτύου που θα κατηγοριοποιήσει εικόνες αριθμών και θα γίνει σύγκριση με κλασικό νευρωνικό δίκτυο για να δούμε ποιο από τα 2 μοντέλα αποδίδει καλύτερα. Αρχικά λοιπόν εγκαθιστούμε τις βασικές βιβλιοθήκες και όλα τα εργαλεία γενικώς που θα χρησιμοποιήσουμε:

```
!pip install tensorflow==2.7.0
!pip install tensorflow-quantum==0.7.2
import importlib, pkg_resources
importlib.reload(pkg_resources)
import tensorflow as tf
import tensorflow_quantum as tfq

import cirq
import sympy
import numpy as np
import seaborn as sns
import collections

# visualization tools
%matplotlib inline
import matplotlib.pyplot as plt
from cirq.contrib.svg import SVGCircuit
```

Στο παράδειγμα μας,όπως είπαμε θα γίνει κατηγοριοποίηση εικόνων ψηφίων, των αριθμών 3 και 6 μόνο, οπότε τα βήματα είναι να φορτώσουμε τα δεδομένα από το Keras, που είναι το API του tensorflow, θα φιλτράρουμε το γενικό σύνολο δεδομένων για να έχουμε μόνο τα ψηφία 3 και 6, θα πρέπει να μειωθεί η ποιότητα της εικόνας (άρα και ο αριθμός των pixel) για να μπορεί να χρησιμοποιηθεί σε κβαντικό υπολογιστή και τέλος είναι η μετατροπή των εικόνων σε κυκλώματα της βιβλιοθήκης cirq, τα οποία με τη σειρά τους θα γίνουν κυκλώματα του tensorflow quantum.

Πρώτα λοιπόν φορτώνουμε τα δεδομένα με το Keras:

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Rescale the images from [0,255] to the [0.0,1.0] range.
x_train, x_test = x_train[..., np.newaxis]/255.0, x_test[..., np.newaxis]/255.0
print("Number of original training examples:", len(x_train))
print("Number of original test examples:", len(x_test))
```

και έχουμε ως αποτέλεσμα:

```
Number of original training examples: 60000
Number of original test examples: 10000
```

Έπειτα φιλτράρουμε τα δεδομένα:

```
def filter_36(x, y):
  keep = (y == 3) | (y == 6)
  x, y = x[keep], y[keep]
  y = y == 3
  return x,y
```

οπότε έχουμε τα νέα σύνολα δεδομένων και εκπαίδευσης:

```
x_train, y_train = filter_36(x_train, y_train)
x_test, y_test = filter_36(x_test, y_test)
print("Number of filtered training examples:", len(x_train))
print("Number of filtered test examples:", len(x_test))
```

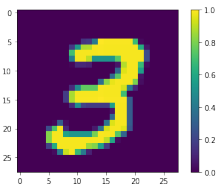
και τα νέα σύνολα εν τέλει είναι:

```
Number of filtered training examples: 12049
Number of filtered test examples: 1968
```

Βλέπουμε το πρώτο παράδειγμα μιας εικόνας ψηφίου:

```
print(y_train[0])
plt.imshow(x_train[0, :, :, 0])
plt.colorbar()
```

που είναι:



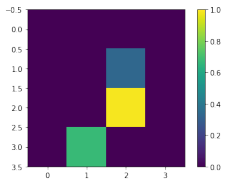
Επειδή οι εικόνες με διαστάσεις 28*28 είναι πολύ μεγάλες για έναν κβαντικό υπολογιστή της τωρινής τεχνολογίας και δεν μπορεί να κάνει υπολογισμούς. Επομένως έχουμε:

```
x_train_small = tf.image.resize(x_train, (4,4)).numpy()
x_test_small = tf.image.resize(x_test, (4,4)).numpy()
```

και πλέον η εικόνα του πρώτου στοιχείου που ήταν το 3 με πολύ ξεκάθαρο τρόπο τώρα πλέον γίνεται:

```
print(y_train[0])
```

```
plt.imshow(x_train_small[0,:,:,0], vmin=0, vmax=1)
plt.colorbar()
```



το επόμενο βήμα, που δεν είναι βασικό σε γενικά προβλήματα μηχανικής μάθησης, είναι να απομακρύνουμε δεδομένα που βρίσκονται για κάποιο λόγο και στις 2 κλάσεις, πχ κάποιο 6 που έτυχε να έχει ως ετικέτα και το 6 και το 3.

```
def remove_contradicting(xs, ys):
  mapping = collections.defaultdict(set)
  orig_x = {}
  # Determine the set of labels for each unique image:
  for x,y in zip(xs,ys):
    orig_x[tuple(x.flatten())] = x
    mapping[tuple(x.flatten())].add(y)

  new_x = []
  new_y = []
  for flatten_x in mapping:
    x = orig_x[flatten_x]
```

```
    labels = mapping[flatten_x]
    if len(labels) == 1:
        new_x.append(x)
        new_y.append(next(iter(labels)))
    else:
        # Throw out images that match more than one label.
        pass

    num_uniq_3 = sum(1 for value in mapping.values() if len(value) == 1 and True in value)
    num_uniq_6 = sum(1 for value in mapping.values() if len(value) == 1 and False in value)
    num_uniq_both = sum(1 for value in mapping.values() if len(value) == 2)

    print("Number of unique images:", len(mapping.values()))
    print("Number of unique 3s: ", num_uniq_3)
    print("Number of unique 6s: ", num_uniq_6)
    print("Number of unique contradicting labels (both 3 and 6): ", num_uniq_both)
    print()
    print("Initial number of images: ", len(xs))
    print("Remaining non-contradicting unique images: ", len(new_x))

    return np.array(new_x), np.array(new_y)
```

Στο επόμενο στάδιο πρέπει να μετατρέψουμε τα δεδομένα σε δυαδικές παραστάσεις, που μπορεί να προκαλέσει κι άλλα προβλήματα. Να σημειώσουμε πως η τεχνική που ακολουθήσαμε προηγουμένως ίσως δεν επιλύσει το πρόβλημα της λάθος κατηγοριοποίησης στο 100%.

```
x_train_nocon, y_train_nocon = remove_contradicting(x_train_small, y_train)
```

```
Number of unique images: 10387
Number of unique 3s: 4912
Number of unique 6s: 5426
Number of unique contradicting labels (both 3 and 6): 49
Initial number of images: 12049
```

```
Remaining non-contradicting unique images: 10338
```

```
_ = remove_contradicting(x_train_bin, y_train_nocon)
```

```
Number of unique images: 193
Number of unique 3s: 80
Number of unique 6s: 69
Number of unique contradicting labels (both 3 and 6): 44
Initial number of images: 10338
Remaining non-contradicting unique images: 149
```

το επόμενο κομμάτι κώδικα είναι από τα πιο βασικά, μιας και μετατρέπει τις εικόνες σε δυαδική μορφή

```
def convert_to_circuit(image):

  """Encode truncated classical image into quantum datapoint."""
  values = np.ndarray.flatten(image)
  qubits = cirq.GridQubit.rect(4, 4)
  circuit = cirq.Circuit()
  for i, value in enumerate(values):
    if value:
      circuit.append(cirq.X(qubits[i]))
  return circuit
x_train_circ = [convert_to_circuit(x) for x in x_train_bin]
x_test_circ = [convert_to_circuit(x) for x in x_test_bin]
```

και στη συνέχεια μετατρέπουμε αυτά τα κυκλώματα που δημιουργήσαμε σε τανυστές για να χρησιμοποιηθούν στο tensorflow quantum:

```
x_train_tfcirc = tfq.convert_to_tensor(x_train_circ)
```

```
x_test_tfcirc = tfq.convert_to_tensor(x_test_circ)
```

Από εδώ και πέρα μπορούμε να χτίζουμε το κβαντικό μοντέλο βήμα-βήμα. Πρώτον κατασκευάζουμε μια κλάση που δημιουργεί επίπεδα με κβαντικές πύλες και τα προσθέτει στο συνολικό κύκλωμα:

```
class CircuitLayerBuilder():

  def __init__(self, data_qubits, readout):
    self.data_qubits = data_qubits
```

```
    self.readout = readout

  def add_layer(self, circuit, gate, prefix):
    for i, qubit in enumerate(self.data_qubits):
      symbol = sympy.Symbol(prefix + '-' + str(i))
      circuit.append(gate(qubit, self.readout)**symbol)
```

Μετά θα χτίσουμε μια συνάρτηση που αναφέρεται σε ένα μοντέλο με 2 επίπεδα πυλών

το οποίο θα συγκρίνει και θα ταιριάξει το μέγεθος των δεδομένων με τις δυνατότητες του κυκλώματος:

```
def create_quantum_model():
  """Create a QNN model circuit and readout operation to go along wit
h it."""
  data_qubits = cirq.GridQubit.rect(4, 4)  # a 4x4 grid.
  readout = cirq.GridQubit(-1, -1)         # a single qubit at [-1,-
1]
  circuit = cirq.Circuit()

  # Prepare the readout qubit.
  circuit.append(cirq.X(readout))
  circuit.append(cirq.H(readout))

  builder = CircuitLayerBuilder(
    data_qubits = data_qubits,
    readout=readout)

  # Then add layers (experiment by adding more).
  builder.add_layer(circuit, cirq.XX, "xx1")
  builder.add_layer(circuit, cirq.ZZ, "zz1")

  # Finally, prepare the readout qubit.
  circuit.append(cirq.H(readout))

  return circuit, cirq.Z(readout)
model_circuit, model_readout = create_quantum_model()
```

Το επόμενο στάδιο είναι επίσης μεγάλης σημασίας, καθώς ερχόμαστε στην ουσία: κατασκευάζουμε ένα μοντέλο με χρήση του keras, που περιέχει κβαντικά μέρη, τροφοδοτείται με κβαντικά δεδομένα μέσω πράξεων από τη βιβλιοθήκη cirq (x_train_cirq) που μετατρέπει σε κβαντικά τα κλασικά δεδομένα που αρχικώς είχαμε. Επίσης, χρησιμοποιεί εργαλεία από το tensorflow quantum (tfq.layers.PQC), το οποίο είναι ένα επίπεδο πυλών με παραμέτρους, για να εκπαιδεύσουμε το κύκλωμα του μοντέλου σε κβαντικά δεδομένα.

```
# Build the Keras model.

model = tf.keras.Sequential([
  # The input is the data-circuit, encoded as a tf.string
  tf.keras.layers.Input(shape=(), dtype=tf.string),
  # The PQC layer returns the expected value of the readout gate, ran
ge [-1,1].
  tfq.layers.PQC(model_circuit, model_readout),
])
```

Μιας κι έχουμε προχωρήσει σημαντικά, πρέπει να κάνουμε μερικές διορθώσεις. Θα χρησιμοποιήσουμε για τη βελτιστοποίηση τη συνάρτηση κόστους hinge loss, αφού τα αποτελέσματα που περιμένουμε θα ανήκουν στο διάστημα [-1,1]. Και για να χρησιμοποιηθεί αυτή η μέθοδος πρέπει να διορθώσουμε τις ετικέτες που χρησιμοποιούν τα δεδομένα μας, από boolean σε [-1,1], όπως επιτάσσει η συνάρτηση hinge loss.

```
y_train_hinge = 2.0*y_train_nocon-1.0

y_test_hinge = 2.0*y_test-1.0
```

Το δεύτερο που θα χρειαστεί να κάνουμε είναι να χρησιμοποιήσουμε τη μετρική για ακρίβεια που ονομάζεται hinge accuracy:

```
def hinge_accuracy(y_true, y_pred):

  y_true = tf.squeeze(y_true) > 0.0
  y_pred = tf.squeeze(y_pred) > 0.0
  result = tf.cast(y_true == y_pred, tf.float32)
  return tf.reduce_mean(result)
model.compile(

  loss=tf.keras.losses.Hinge(),
  optimizer=tf.keras.optimizers.Adam(),
  metrics=[hinge_accuracy])
print(model.summary())
```

Έπειτα είναι η εκπαίδευση του κβαντικού μοντέλου:

```
EPOCHS = 3
BATCH_SIZE = 32

NUM_EXAMPLES = len(x_train_tfcirc)
x_train_tfcirc_sub = x_train_tfcirc[:NUM_EXAMPLES]
y_train_hinge_sub = y_train_hinge[:NUM_EXAMPLES]
qnn_history = model.fit(
  x_train_tfcirc_sub, y_train_hinge_sub,
  batch_size=32,
  epochs=EPOCHS,
  verbose=1,
  validation_data=(x_test_tfcirc, y_test_hinge))

qnn_results = model.evaluate(x_test_tfcirc, y_test)
```

Εδώ κάνουμε εκπαίδευση σε όλο το σύνολο δεδομένων, σε 3 εποχές όπως ονομάζονται τα στάδια εκπαίδευσης και συνολικά 32 παραμέτρους προς βελτιστοποίηση, οπότε τα αποτελέσματα είναι τα ακόλουθα:

```
Epoch 1/3 324/324 [==============================] - 709s 2s/step - loss: 0.3646 - hinge_accuracy: 0.8228 - val_loss: 0.3323 - val_hinge_accuracy: 0.8569

Epoch 2/3 324/324 [==============================] - 711s 2s/step - loss: 0.3490 - hinge_accuracy: 0.8696 - val_loss: 0.3358 - val_hinge_accuracy: 0.9047

Epoch 3/3 324/324 [==============================] - 700s 2s/step - loss: 0.3451 - hinge_accuracy: 0.8896 - val_loss: 0.3307 - val_hinge_accuracy: 0.9062 62/62 [==============================] - 22s 359ms/step - loss: 0.3307 - hinge_accuracy: 0.9062
```

Στο τελευταίο μέρος του παραδείγματος θα γίνει εκπαίδευση ενός κλασικού νευρωνικού δικτύου για να δούμε τις διαφορές στην απόδοση και να γίνει μια σύγκριση συνολικά. Έχουμε

λοιπόν ότι το κλασικό νευρωνικό δίκτυο, χρησιμοποιώντας τις εικόνες όπως πραγματικά είναι,στις διαστάσεις 28*28, μπορεί και πετυχαίνει αποτελέσματα κατά πολύ καλύτερα του κβαντικού, ειδικά στο απλό πρόβλημα που αντιμετωπίσαμε εμείς, της κατηγοριοποίησης δηλαδή εικόνων των 3 και 6.

```
def create_classical_model():
  # A simple model based off LeNet from https://keras.io/examples/
mnist_cnn/
  model = tf.keras.Sequential()
  model.add(tf.keras.layers.Conv2D(32, [3, 3], activation='relu', inp
ut_shape=(28,28,1)))
  model.add(tf.keras.layers.Conv2D(64, [3, 3], activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(128, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(1))
  return model
model = create_classical_model()
model
.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
        optimizer=tf.keras.optimizers.Adam(),
        metrics=['accuracy'])

model.summary()
```

που μας δίνει ως αποτέλεσμα:

```
Model:                                                          "sequential_1"
_________________________________________________________________   Layer   (type)
Output                          Shape                          Param                    #
=================================================================  conv2d  (Conv2D)
(None, 26, 26, 32) 320 conv2d_1 (Conv2D) (None, 24, 24, 64) 18496 max_pooling2d
(MaxPooling2D (None, 12, 12, 64) 0 ) dropout (Dropout) (None, 12, 12, 64) 0 flatten
(Flatten) (None, 9216) 0 dense (Dense) (None, 128) 1179776 dropout_1 (Dropout)
(None,       128)       0       dense_1       (Dense)       (None,       1)       129
=================================================================   Total   params:
```

```
1,198,721    Trainable    params:    1,198,721    Non-trainable    params:    0
```

άρα εκπαιδεύουμε ένα συνελικτικό νευρωνικό δίκτυο με σχεδόν 1.2 εκατομμύρια παραμέτρους! Προφανώς λοιπόν το κλασικό μοντέλο-προς το παρόν τουλάχιστον-υπερέχει κατά πολύ του κβαντικού,και σε πολύ μικρότερο χρόνο:

```
model.fit(x_train,
          y_train,
          batch_size=128,
          epochs=1,
          verbose=1,
          validation_data=(x_test, y_test))

cnn_results = model.evaluate(x_test, y_test)
```

και η ακρίβεια είναι μεγαλύτερη με λιγότερες επαναλήψεις:

```
95/95 [==============================] - 31s 312ms/step - loss: 0.0483 - accuracy:
0.9815     -     val_loss:     0.0028     -     val_accuracy:     0.9995     62/62
[==============================] - 1s 22ms/step - loss: 0.0028 - accuracy: 0.9995
```

και προσεγγίζει σχεδόν το 100%. Και για να γίνει μια πιο δίκαιη σύγκριση, κατασκευάζουμε ένα μοντέλο με 37 παραμέτρους:

```
def create_fair_classical_model():
  # A simple model based off LeNet from https://keras.io/examples/
mnist_cnn/
  model = tf.keras.Sequential()
  model.add(tf.keras.layers.Flatten(input_shape=(4,4,1)))
  model.add(tf.keras.layers.Dense(2, activation='relu'))
  model.add(tf.keras.layers.Dense(1))
  return model


model = create_fair_classical_model()
model
.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
        optimizer=tf.keras.optimizers.Adam(),
        metrics=['accuracy'])

model.summary()
```

με τις παρακάτω λεπτομέρειες:

```
Model:                                                                 "sequential_2"
_________________________________________________________________   Layer    (type)
Output                       Shape                        Param                   #
=================================================================          flatten_1
(Flatten)  (None, 16)  0 dense_2 (Dense)  (None, 2)  34 dense_3 (Dense)  (None, 1)  3
================================================================= Total params: 37
Trainable params: 37 Non-trainable params: 0
```

οπότε όταν το τρέχουμε:

```
model.fit(x_train_bin,
    y_train_nocon,
    batch_size=128,
    epochs=20,
    verbose=2,
    validation_data=(x_test_bin, y_test))

fair_nn_results = model.evaluate(x_test_bin, y_test)
```

παίρνουμε τα εξής αποτελέσματα:

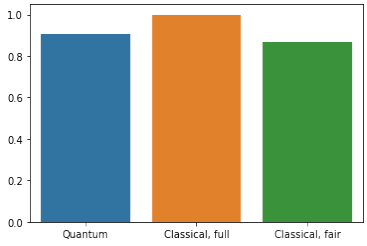
```
Epoch 1/20 81/81 - 1s - loss: 0.6465 - accuracy: 0.6452 - val_loss: 0.6313 -
val_accuracy: 0.6138 - 652ms/epoch - 8ms/step
Epoch 2/20 81/81 - 0s - loss: 0.5947 - accuracy: 0.6919 - val_loss: 0.5722 -
val_accuracy: 0.6280 - 147ms/epoch - 2ms/step
Epoch 3/20 81/81 - 0s - loss: 0.5454 - accuracy: 0.8046 - val_loss: 0.5245 -
val_accuracy: 0.8008 - 153ms/epoch - 2ms/step
Epoch 4/20 81/81 - 0s - loss: 0.5034 - accuracy: 0.8364 - val_loss: 0.4824 -
val_accuracy: 0.8155 - 132ms/epoch - 2ms/step
Epoch 5/20 81/81 - 0s - loss: 0.4645 - accuracy: 0.8422 - val_loss: 0.4411 -
val_accuracy: 0.8552 - 150ms/epoch - 2ms/step
Epoch 6/20 81/81 - 0s - loss: 0.4178 - accuracy: 0.8657 - val_loss: 0.3872 -
val_accuracy: 0.8582 - 136ms/epoch - 2ms/step

Epoch 7/20 81/81 - 0s - loss: 0.3632 - accuracy: 0.8683 - val_loss: 0.3378 -
val_accuracy: 0.8592 - 146ms/epoch - 2ms/step

Epoch 8/20 81/81 - 0s - loss: 0.3190 - accuracy: 0.8686 - val_loss: 0.3023 -
val_accuracy: 0.8587 - 145ms/epoch - 2ms/step

Epoch 9/20 81/81 - 0s - loss: 0.2889 - accuracy: 0.8728 - val_loss: 0.2777 -
val_accuracy: 0.8659 - 144ms/epoch - 2ms/step

Epoch 10/20 81/81 - 0s - loss: 0.2685 - accuracy: 0.8749 - val_loss: 0.2608 -
val_accuracy: 0.8669 - 145ms/epoch - 2ms/step
Epoch 11/20 81/81 - 0s - loss: 0.2547 - accuracy: 0.8777 - val_loss: 0.2495 -
val_accuracy: 0.8669 - 180ms/epoch - 2ms/step
Epoch 12/20 81/81 - 0s - loss: 0.2451 - accuracy: 0.8783 - val_loss: 0.2413 -
val_accuracy: 0.8669 - 156ms/epoch - 2ms/step
Epoch 13/20 81/81 - 0s - loss: 0.2384 - accuracy: 0.8783 - val_loss: 0.2357 -
val_accuracy: 0.8669 - 128ms/epoch - 2ms/step
```

```
Epoch 14/20 81/81 - 0s - loss: 0.2335 - accuracy: 0.8800 - val_loss: 0.2316 -
val_accuracy: 0.8674 - 132ms/epoch - 2ms/step
Epoch 15/20 81/81 - 0s - loss: 0.2300 - accuracy: 0.8805 - val_loss: 0.2285 -
val_accuracy: 0.8679 - 141ms/epoch - 2ms/step
Epoch 16/20 81/81 - 0s - loss: 0.2274 - accuracy: 0.8805 - val_loss: 0.2263 -
val_accuracy: 0.8679 - 140ms/epoch - 2ms/step
Epoch 17/20 81/81 - 0s - loss: 0.2255 - accuracy: 0.8805 - val_loss: 0.2244 -
val_accuracy: 0.8679 - 160ms/epoch - 2ms/step
Epoch 18/20 81/81 - 0s - loss: 0.2241 - accuracy: 0.8808 - val_loss: 0.2231 -
val_accuracy: 0.8679 - 146ms/epoch - 2ms/step
Epoch 19/20 81/81 - 0s - loss: 0.2229 - accuracy: 0.8810 - val_loss: 0.2220 -
val_accuracy: 0.8689 - 162ms/epoch - 2ms/step
Epoch 20/20 81/81 - 0s - loss: 0.2221 - accuracy: 0.8810 - val_loss: 0.2211 -
val_accuracy:      0.8689      -      137ms/epoch      -      2ms/step      62/62
[==============================] - 0s 1ms/step - loss: 0.2211 - accuracy: 0.8689
```

Άρα στην τελική σύγκριση:

```
qnn_accuracy = qnn_results[1]
cnn_accuracy = cnn_results[1]
fair_nn_accuracy = fair_nn_results[1]
sns.barplot(x=["Quantum", "Classical, full", "Classical, fair"],
     y=[qnn_accuracy, cnn_accuracy, fair_nn_accuracy])
```

έχουμε το ακόλουθο:



που σημαίνει πως όταν κάναμε σύγκριση του κβαντικού νευρωνικού δικτύου με το κλασικό των 37 παραμέτρων, η διαφορά δεν είναι και τόσο μεγάλη, τα 2 μοντέλα είναι ισοδύναμα ως προς την απόδοση τους.

6) παράδειγμα κβαντικού κυκλώματος με pennylane και pytorch

Εδώ θα δούμε ένα μικρό και γρήγορο παράδειγμα για το πως κατασκευάζουμε ένα κβαντικό κύκλωμα και πως βελτιστοποιούμε μέσω των βιβλιοθηκών pennylane και pytorch [28].

Θα δημιουργήσουμε ένα κβαντικό κύκλωμα που θα κάνει preparation καταστάσεων με ένα qubit κάθε φορά, οι οποίες καταστάσεις είναι είτε pure είτε mixed (είχαμε δει τους ορισμούς στην αρχή της παρούσας εργασίας).

Κάποιοι ορισμοί που θα χρειαστούμε επιπλέον: γενικώς ένα qubit μπορεί να αναπαρασταθεί με βάση κάποιες έννοιες όπως ο πίνακας πυκνότητας ρ (density matrix) και το διάνυσμα με το οποίο περιγράφεται πάνω στη σφαίρα Bloch. Ο density matrix μπορεί να γραφεί ως γραμμικός συνδυασμός αυτού του διανύσματος της σφαίρας Bloch και των πινάκων Pauli.

$$\boldsymbol{\rho = 1/2 * (1 + \alpha_x \sigma_x + \alpha_y \sigma_y + \alpha_z \sigma_z)}$$

κάθε διάνυσμα Bloch συσχετίζεται με έναν πίνακα πυκνότητας αρκεί να ισχύει η σχέση:

$$\|\vec{a}\| \leq 1$$

Η καθαρότητα μιας κβαντικής κατάστασης γενικά ορίζεται από τη σχέση:

$$p = Tr(\rho^2)$$

και για τα qubit συγκεκριμένα η ποσότητα αυτή πρέπει να βρίσκεται στα όρια:

$$1/2 \leq p \leq 1$$

Για p=1 έχουμε πως η κατάσταση είναι pure, ενώ για p=1/2 η κατάσταση χαρακτηρίζεται ως maximally mixed. Στο παρόν παράδειγμα έχουμε κάποιες καταστάσεις ως στόχους χρησιμοποιώντας ένα τυχαίο διάνυσμα Bloch του οποίου αλλάζουμε το μήκος τόσο ώστε να αποκτήσει μια συγκεκριμένη purity (αυτό στην ουσία είναι όλο το πρόγραμμα που θα κατασκευάσουμε).

Πρώτα λοιπόν έχουμε:

```
import pennylane as qml
```

```
import numpy as np

import torch

from torch.autograd import Variable

np.random.seed(42)

# Δημιουργούμε ένα τυχαίο διάνυσμα 3 διαστάσεων,χρησιμοποιώντας τυχαία κατανομή

v = np.random.normal(0, 1, 3)

#η καθαρότητα(purity)της κατάστασης-στόχου

purity = 0.66
```

```
# κατασκευάζουμε τυχαίο  διάνυσμα Bloch με τη συγκεκριμένη purity που ορίσαμε πάνω

bloch_v = Variable(

     torch.tensor(np.sqrt(2 * purity - 1) * v / np.sqrt(np.sum(v ** 2))),

     requires_grad=False

)



# κατασκευή ενός array από πίνακες Pauli

Paulis = Variable(torch.zeros([3, 2, 2], dtype=torch.complex128), requires_grad=False)

Paulis[0] = torch.tensor([[0, 1], [1, 0]])

Paulis[1] = torch.tensor([[0, -1j], [1j, 0]])

Paulis[2] = torch.tensor([[1, 0], [0, -1]])
```

Ξέρουμε πως οι ορθομοναδιαίοι πίνακες που είναι τα εργαλεία μας για να κάνουμε υπολογισμούς στα κβαντικά κυκλώματα απεικονίζουν pure καταστάσεις σε pure καταστάσεις. Εδώ λοιπόν επειδή θέλουμε να λαμβάνονται υπ’όψιν και mixed states θα προσθέσουμε κι άλλα qubits (2 στο παρόν παράδειγμα) που αρκούν για preparation τυχαίων καταστάσεων.

Στο κύκλωμα έχουμε επαναλαμβανόμενα επίπεδα, καθένα από τα οποία κάνει σε κάθε περίπτωση σε ένα qubit μια περιστροφή γύρω από τους άξονες x,y και z, και στη συνέχεια χρησιμοποιούμε την CNOT πύλη για να κάνουμε διεμπλοκή όλα τα qubits μεταξύ τους. Οι παράμετροι για τις πύλες (με τις οποίες το κύκλωμα θα εκπαιδευτεί) έχουν επιλεχθεί τυχαία από κανονική κατανομή.

```
# πλήθος qubits στο κύκλωμα

nr_qubits = 3

#πλήθος επιπέδων σε κάθε κύκλωμα

nr_layers = 2

# κατασκευή παραμέτρων,που όπως είπαμε πρέπει να αρχικοποιηθούν τυχαία,από κανονική κατανομή

params = np.random.normal(0, np.pi, (nr_qubits, nr_layers, 3))

params = Variable(torch.tensor(params), requires_grad=True)
```

```
# κατασκευή επιπέδου στο βασικό κύκλωμα ansatz

def layer(params, j):

     for i in range(nr_qubits):

          qml.RX(params[i, j, 0], wires=i)

          qml.RY(params[i, j, 1], wires=i)

          qml.RZ(params[i, j, 2], wires=i)



     qml.CNOT(wires=[0, 1])

     qml.CNOT(wires=[0, 2])

     qml.CNOT(wires=[1, 2])
```

Μετά επιλέγουμε όπως και σε άλλες καταστάσεις, τη συσκευή στην οποί α θα γίνει όλη η διεργασία:

```
dev = qml.device("default.qubit", wires=3)
```

Στο επόμενο βήμα θα ορίσουμε το Qnode, που συνενώνει τη συσκευή που θα χρησιμοποιηθεί και τη συνάρτηση (το κύκλωμα). Εκεί θα ορίσουμε κι έναν Ερμητιανό τελεστή που θα είναι η αναμενόμενη τιμή (expectation value):

```
@qml.qnode(dev, interface="torch")

def circuit(params, A):



     for j in range(nr_layers):

          layer(params, j)



     return qml.expval(qml.Hermitian(A, wires=0))
```

Ο σκοπός όπως είπαμε είναι να συσχετίσουμε ένα τυχαίο διάνυσμα Bloch με μια κατάσταση στόχο. Ορίζουμε λοιπόν μια συνάρτηση κόστους:

$$C = \sum_{i=1}^{3} |(a_i - a'_i)|$$

όπου το διάνυσμα a_i είναι το διάνυσμα στόχος και το a'_i είναι το διάνυσμα που προέκυψε από τους υπολογισμούς του κβαντικού κυκλώματος. Ο σκοπός μας είναι η ποσότητα αυτή να τείνει στο 0. Χρησιμοποιούμε τον Adam Optimizer ως εργαλείο βελτιστοποίησης:

```
# cost function

def cost_fn(params):

     cost = 0

     for k in range(3):

          cost += torch.abs(circuit(params, Paulis[k]) - bloch_v[k])



     return cost

# ορίζουμε τον Adam optimizer

opt = torch.optim.Adam([params], lr=0.1)

# πλήθος βημάτων στην βελτιστοποίηση

steps = 200

# θα ελέγχουμε κάθε 10 βήματα το πώς πραγματοποιείται η βελτιστοποίηση

best_cost = cost_fn(params)

best_params = np.zeros((nr_qubits, nr_layers, 3))

print("Cost after 0 steps is {:.4f}".format(cost_fn(params)))

for n in range(steps):

     opt.zero_grad()

     loss = cost_fn(params)

     loss.backward()

     opt.step()

 #  εδώ αποθηκεύουμε τις καλύτερες παραμέτρους

     if loss < best_cost:
```

```
            best_cost = loss

            best_params = params

    #έλεγχος κάθε 10 βήματα στο τί γίνεται

        if n % 10 == 9 or n == steps - 1:

            print("Cost after {} steps is {:.4f}".format(n + 1, loss))

# υπολογισμός του διανύσματος Bloch

output_bloch_v = np.zeros(3)

for l in range(3):

        output_bloch_v[l] = circuit(best_params, Paulis[l])




print("Target Bloch vector = ", bloch_v.numpy())

print("Output Bloch vector = ", output_bloch_v)
```

με τα αποτελέσματα να είναι τα ακόλουθα:

```
Cost after 0 steps is 1.0179
Cost after 10 steps is 0.1467
Cost after 20 steps is 0.0768
Cost after 30 steps is 0.0813
Cost after 40 steps is 0.0807
Cost after 50 steps is 0.0940
Cost after 60 steps is 0.0614
Cost after 70 steps is 0.0932
Cost after 80 steps is 0.0455
Cost after 90 steps is 0.0752
Cost after 100 steps is 0.0301
Cost after 110 steps is 0.0363
Cost after 120 steps is 0.1332
Cost after 130 steps is 0.0687
Cost after 140 steps is 0.0505
Cost after 150 steps is 0.0800
Cost after 160 steps is 0.0644
Cost after 170 steps is 0.0813
Cost after 180 steps is 0.0592
Cost after 190 steps is 0.0502
Cost after 200 steps is 0.0573
Target Bloch vector =  [ 0.33941241 -0.09447812  0.44257553]
```

```
Output Bloch vector =  [ 0.3070776  -0.07421944  0.47393046]
```

7) παράδειγμα VQE με Qiskit

Εδώ θα δούμε ένα παράδειγμα του Variational Quantum Eigensolver [29] που θα μας βοηθήσει να βρούμε τις ιδιοτιμές ενός πίνακα, του

$$U=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Και για να συμβεί αυτό, θα πρέπει να γράψουμε τον U ως γινόμενο πινάκων Pauli, τους οποίους είδαμε προηγουμένως, διότι έχουμε να κάνουμε με κβαντικό υπολογιστή. Ο πίνακας U γράφεται ως γραμμικός συνδυασμός των πινάκων Pauli ως εξής:

$$U=1/2\left(I_1\otimes I_2+Z_1\otimes Z_2\right)-1/2\left(X_1\otimes X_2+Y_1\otimes Y_2\right)(*)$$

Γενικώς ο VQE χρησιμοποιείται σε εφαρμογές της Κβαντικής Χημείας, όταν θέλουμε να βρούμε το ground state ενός μορίου, την κατάσταση δηλαδή που η ενέργειά του είναι η ελάχιστη δυνατή. Είναι ένας αλγόριθμος βελτιστοποίησης και συγκεκριμένα ελαχιστοποίησης.

Από εδώ και πέρα θα αναφερόμαστε στον U ως την Χαμιλτονιανή του προβλήματος (=ο τελεστής που αναφέρεται στην ολική ενέργεια ενός συστήματος) και θα γράφουμε:

$U\rightarrow H$. Θέλουμε την ελάχιστη ενέργεια του συστήματος, της ποσότητας

$$E(\theta)=\langle H\rangle_\theta=\langle\psi(\theta)|H|\psi(\theta)\rangle$$

δοθείσης μιας τυχαίας κατάστασης $|\psi(\theta)\rangle$. Αυτή ακριβώς η κατάσταση είναι ιδιοδιάνυσμα του H με την E(θ) να είναι η αντίστοιχη ιδιοτιμή και σκοπός μας είναι να βρεθεί η ιδιοτιμή με την ελάχιστη τιμή για τον ορθομοναδιαίο πίνακα H. Και οι 2 ποσότητες εξαρτώνται από μία παράμετρο θ, την οποία και θα βελτιστοποιήσουμε.

Θέλουμε να κατασκευάσουμε στην ουσία, κάποια κβαντικά κυκλώματα για υπολογίσουν τις ποσότητες X_1X_2,Z_1Z_2 και Y_1Y_2, από την εξίσωση (*) κι έπειτα να τα συνδυάσουμε για να βρούμε την ποσότητα $E=\langle H\rangle$.

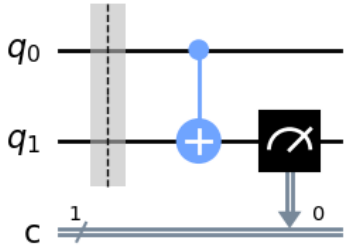
Πρώτον λοιπόν, κατασκευάζουμε το κύκλωμα που αναφέρεται στην ποσότητα Z_1Z_2

```
import qiskit as qk
qc = qk.QuantumCircuit(2,1)
qc.barrier()
qc.cx(0,1)
qc.measure(1,0)
print("Measurement in the ZZ basis")
```

```
qc.draw(output="mpl")
```

το οποίο είναι ένα κβαντικό κύκλωμα με 2 qubits (αρκούν για τον υπολογισμό μας), ένας κλασικός καταχωρητής που θα αποθηκεύσει το αποτέλεσμα της παρατήρησης και η πράξη που συντελείται είναι απλώς μία CNOT μεταξύ του πρώτου και του δεύτερου qubit.

Σχηματικά έχουμε:

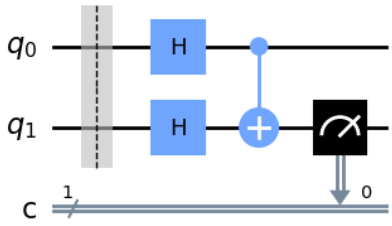


Αντιστοίχως πράττουμε για τις ποσότητες X_1X_2 και Y_1Y_2

Κύκλωμα για X_1X_2

```
qc = qk.QuantumCircuit(2,1)
qc.barrier()
qc.h(0)
qc.h(1)
qc.cx(0,1)
qc.measure(1,0)
print("Measurement in the XX basis")
qc.draw(output="mpl")
```

και σχηματικά το ακόλουθο κύκλωμα:

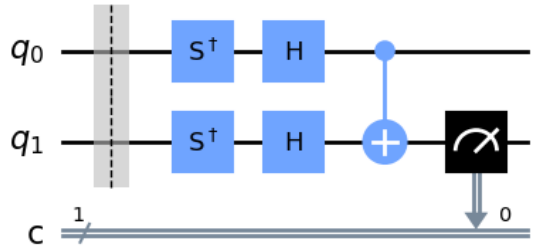


Ομοίως με το προηγούμενο, πάλι 2 qubits, καταχωρητής για την παρατήρηση, μόνο που εδώ επιδρά η πύλη Hadamard στα 2 qubits και τα φέρνει σε υπέρθεση (superposition) και μετά έχουμε μια CNOT και μετρούμε το αποτέλεσμα.

Κύκλωμα για Y_1Y_2

```
qc = qk.QuantumCircuit(2,1)
qc.barrier()
qc.sdg(0)
qc.sdg(1)
qc.h(0)
qc.h(1)
qc.cx(0,1)
qc.measure(1,0)
print("Measurement in the YY basis")
qc.draw(output="mpl")
```

Με σχήμα:



Ακριβώς όπως με τις άλλες 2 περιπτώσεις, 2 qubits και 1 καταχωρητής αποτελέσματος, απλώς εδώ έχουμε και πράξεις με τον συζυγή του πίνακα S, ο οποίος S ορίζεται ως η τετραγωνική ρίζα του πίνακα Z. Από εδώ και πέρα ξεκινάει το σημαντικό κομμάτι του κώδικα. Αρχικά κάποιες βασικές βιβλιοθήκες:

```
from qiskit import Aer
import numpy as np
from scipy.optimize import minimize_scalar, minimize
from numpy import pi
sim_bknd = qk.Aer.get_backend('qasm_simulator')
```

που είναι ο προσομοιωτής και τα όποια εργαλεία χρειάζεται. Κι έπειτα ορίζουμε το πλαίσιο στο οποίο θα δουλέψουμε, το παραμετροποιημένο κβαντικό κύκλωμα που θα κάνει τον υπολογισμό της βελτιστοποίησης της παραμέτρου θ.

```
def ansatz(qc, qr, theta):
"""



Return
qc:επιστρέφει το αρχικό κβαντικό κύκλωμα που του έχουμε προσθέσει πύλες για να κατασκευάσουμε την αρχική τυχαία κατάσταση
"""
qc.h(qr[0])
qc.cx(qr[0],qr[1])
qc.rx(theta, qr[0])


return qc


def measurements(qc, qr, cr, op):
"""
Αυτή η συνάρτηση κάνει υπολογισμούς σε διαφορετικές βάσεις,τις: XX, YY and ZZ.
Τα ορίσματά της είναι
qc:ομοίως με πριν
qr:ομοίως με πριν
cr: κλασικός καταχωρητής, αντικείμενο της κλάσης  ClassicalRegister.
op (str): αλφαριθμητικό με πιθανές τιμές: XX, YY and ZZ.



Return
---------
qc: returns the input quantum circuit added with the appropriate gates to measure in the selected basis.
Επιστρέφει το κύκλωμα με επιπλέον πύλες, κατάλληλες για να μετρήσουμε την αντίστοιχη βάση κάθε φορά.
"""
if op == "XX":
# αλλάζουμε τη βάση, αφού X = HZH
```

```
qc.h(qr[0])
qc.h(qr[1])

#H  CNOT χρησιμοποιήθηκε για να μετρήσουμε τον τελεστή ZZ
qc.cx(qr[0],qr[1])
# Παρατήρηση του qubit 1 και καταχώρηση στον κλασικό καταχωρητή 0
qc.measure(qr[1],cr[0])

elif op == "YY":
#κι εδώ αλλαγή βάσης, αφού Y = (HS†)Z(HS†)
qc.sdg(qr[0])
qc.sdg(qr[1])
qc.h(qr[0])
qc.h(qr[1])

#H  CNOT χρησιμοποιήθηκε για να μετρήσουμε τον τελεστή ZZ
qc.cx(qr[0],qr[1])

# Παρατήρηση του qubit 1 και καταχώρηση στον κλασικό καταχωρητή 0
qc.measure(qr[1],cr[0])

elif op == "ZZ":
#H  CNOT χρησιμοποιήθηκε για να μετρήσουμε τον τελεστή ZZ
qc.cx(qr[0],qr[1])

# Παρατήρηση του qubit 1 και καταχώρηση στον κλασικό καταχωρητή 0
qc.measure(qr[1],cr[0])

else:
print(f"WARNING: Measurement on the {op} basis not supported")
return

return qc
```

def hamiltonian(params):

"""

Υπολογίζει την ενέργεια των τυχαίων καταστάσεων χρησιμοποιώντας τους μέσους όρους των τελεστών XX, YY and ZZ.

όρισμα

params (dict): λεξικό, που περιέχει τις μέσες τιμές των XX, YY, ZZ

Return

en (real): την ενέργεια του συστήματος

"""

H = 1/2 * (Id + ZZ - XX - YY)

en = (1 + params['ZZ'] - params['XX'] - params['YY']) / 2

return en

def vqe_step(theta, verbose = True):

"""

Εδώ έχουμε τον VQE.

Κατασκευάζει και εκτελεί 3 κβαντικά κυκλώματα για κάθε έναν από τους τελεστέςXX, YY and ZZ, κι έπειτα υπολογίζει την ενέργεια.

ορίσματα:

theta (real): η παράμετρος που θα βελτιστοποιηθεί

Return

energy (real):την ενέργεια του συστήματος

qc_list (dict): λεξικό, που περιέχει τα 3 κβαντικά κυκλώματα για τους τελεστές XX, YY, ZZ

"""

#πλήθος επαναλήψεων για κάθε κύκλωμα

shots=8192

vqe_res = dict()

qc_list = dict()

for op in ["XX", "YY", "ZZ"]:

```
qr = qk.QuantumRegister(2, "qr")
cr = qk.ClassicalRegister(1, "cr")
qc = qk.QuantumCircuit(qr, cr)

#Τώρα εφαρμόζουμε το ansatz, το γενικό πλαίσιο που περιέχει όλους τους υπολογισμούς που περιγράψαμε
qc = ansatz(qc, qr, theta)
qc.barrier()

# Μέτρηση στη σωστή βάση κάθε φορά (XX, YY, ZZ)
qc = measurements(qc, qr, cr, op)

# αποθήκευση των αποτελέσμάτων της μέτρησης
counts = qk.execute(qc, sim_bknd, shots=shots).result().get_counts()

# Ελέγχουμε τα αποτελέσματα και υπολογίζουμε τον μέσο όρο
if len(counts) == 1:
try:
counts['0']
mean_val = 1
except:
mean_val = -1
else:
mean_val = (counts['0']-counts['1'])/shots

vqe_res[op] = mean_val
qc_list[op] = qc

energy = hamiltonian(vqe_res)

if verbose:
print("Mean values from measurement results:\n", vqe_res)
print(f"\n{'Theta':<10} {'Energy':<10} {'<XX>':<10} {'<YY>':<10} {'<ZZ>':<10}")
print(f"{theta:<10f} {energy:<10f} {vqe_res['XX']:<10f} {vqe_res['YY']:<10f} {vqe_res['ZZ']:<10f}")
```

return energy, qc_list

else:

return energy

Το επόμενο βήμα είναι να ελέγξουμε αν δουλεύει σωστά το ansatz

#Θέτουμε τιμή για το θ

theta = 0.2

Τρέχουμε τον VQE για να υπολογίσουμε την ιδιοτιμή της χαμιλτονιανής,ή αλλιώς την ενέργεια του συστήματος για δοθέν θ

energy, qc_list = vqe_step(theta)

op = 'YY'

print(f"\nQuantum circuit for the measurement of {op}")

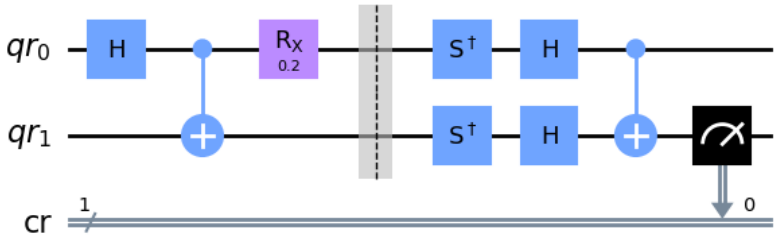
qc_list[op].draw(output="mpl")

το αποτέλεσμα των πράξεων είναι το παρακάτω:

Mean values from measurement results:

```
 {'XX': 1, 'YY': -0.9833984375, 'ZZ': 0.983642578125}

Theta      Energy     <XX>       <YY>       <ZZ>
0.200000   0.983521   1.000000   -0.983398  0.983643

Quantum circuit for the measurement of YY
```

και σχηματικά:



Εδώ βλέπουμε πως για τιμή παραμέτρου θ=0.2,η ενέργεια του συστήματος E(0.2) είναι σχεδόν ίση με 0.98. Μετά λοιπόν, για να βρούμε εν τέλει την ελάχιστη τιμή της ενέργειας, χρησιμοποιούμε έναν optimizer:

minimize_scalar(vqe_step, args=(**False**), bounds = (0, pi), method = "bounded")

που μας δίνει αποτέλεσμα:

```
fun: -1.0
message: 'Solution found.'
nfev: 27
nit: 27
status: 0
success: True
```

```
x: 3.133436557070585
```

βρίσκουμε λύση τελικά και αυτή είναι για θ περίπου ίσο με π, όπου η ενέργεια είναι ίση με 1. Αν θέσουμε θ=π, έχουμε:

lowest, _ = vqe_step(pi)

Mean values from measurement results:

```
{'XX': 1, 'YY': 1, 'ZZ': -1}

   Theta      Energy      <XX>       <YY>       <ZZ>
3.141593   -1.000000  1.000000   1.000000   -1.000000
```

και επομένως στην εξίσωση $\langle H \rangle_{\theta=\pi}=1/2(1+\langle ZZ \rangle)-1/2(\langle XX \rangle+\langle YY \rangle)=-1$,αν $\langle XX \rangle=\langle YY \rangle=1$ και $\langle ZZ \rangle=-1$.

Ένα τελευταίο βήμα είναι να τσεκάρουμε το αποτέλεσμα συγκρίνοντας το με κλασικό υπολογισμό με χρήση της βιβλιοθήκης SciPy.

#ορισμός των πινάκων Pauli για ένα και 2 qubits

```
I = np.array([[1,0],[0,1]])
X = np.array([[0,1],[1,0]])
Y = np.array([[0,-1j],[1j,0]])
Z = np.array([[1,0],[0,-1]])


II = np.kron(I,I)
XX = np.kron(X,X)
YY = np.kron(Y,Y)
ZZ = np.kron(Z,Z)

# Υπολογισμός της Χαμιλτονιανής
H = (1/2) * (II+ZZ) - (1/2) * (XX+YY)

print("Desired Hamiltionian H = \n", H)
```

με αποτέλεσμα:

```
Desired Hamiltionian H =
 [[ 1.+0.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j -1.+0.j  0.+0.j]
 [ 0.+0.j -1.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+0.j  1.+0.j]]
```

που είναι ακριβώς ο αρχικός μας πίνακας U, του οποίου την ιδιοτιμή ψάχνουμε. Με χρήση λοιπόν της SciPy έχουμε:

import scipy

```
# υπολογισμός ιδιοτιμών και ιδιοδιανυσμάτων του  H
eigenvalues, eigenvectors = scipy.linalg.eig(H)
```

```
print("Eigenvalues:", eigenvalues)
```

και αποτέλεσμα:

Eigenvalues: [1.+0.j -1.+0.j 1.+0.j 1.+0.j]

8) παράδειγμα VQE με Pennylane

Κι εδώ θα έχουμε ένα τελευταίο απλό προγραμματιστικό παράδειγμα με χρήση του pennylane, σε ένα ίσως πιο “σωστό” για τον VQE πρόβλημα, μιας και θα αναζητήσουμε το ground state ενός μορίου υδρογόνου [30]. Τα βήματα είναι να κατασκευάσουμε μια χαμιλτονιανή που θα αντιπροσωπεύει το μόριο του υδρογόνου, να κατασκευάσουμε ένα κύκλωμα που θα ορίσει μια κατάσταση για το μόριο (preparation) και μια συνάρτηση κόστους για να υπολογίσουμε την αναμενόμενη τιμή (expectation value) της χαμιλτονιανής. Στο τέλος θα κατασκευαστεί κι ένας κλασικός optimizer και θα τρέξουμε τον VQE με προσομοιωτή.
Αρχικά λοιπόν κάνουμε τα απαραίτητα imports και ορίζουμε το υδρογόνο ως 1) μια λίστα από σύμβολα που ορίζουν το μόριο του υδρογόνου (όπως ξέρουμε αποτελείται από 2 άτομα) κι έπειτα ορίζουμε και μια 2η λίστα με τις αντίστοιχες συντεταγμένες των ατόμων αυτών:

```
from pennylane import numpy as np

symbols = ["H", "H"]
coordinates = np.array([0.0, 0.0, -0.6614, 0.0, 0.0, 0.6614])
Τώρα θα ορίσουμε την χαμιλτονιανή:
import pennylane as qml

H, qubits = qml.qchem.molecular_hamiltonian(symbols, coordinates)
print("Number of qubits = ", qubits)
print("The Hamiltonian is ", H)
```

με αποτέλεσμα:

Number of qubits = 4

```
The Hamiltonian is    (-0.24274501260974596) [Z2]
          + (-0.2427450126097459) [Z3]
          + (-0.04207255194714504) [I0]
          + (0.17771358229071466) [Z1]
          + (0.1771135822907147) [Z0]
          + (0.12293330449309582) [Z0 Z2]
          + (0.12293330449309582) [Z1 Z3]
          + (0.16768338855615295) [Z0 Z3]
          + (0.16768338855615295) [Z1 Z2]
          + (0.1705975927684125) [Z0 Z1]
          + (0.17627661394206545) [Z2 Z3]
        + (-0.04475008406305714) [Y0 Y1 X2 X3]
        + (-0.04475008406305714) [X0 X1 Y2 Y3]
        + (0.04475008406305714) [Y0 X1 X2 Y3]
```

```
+ (0.04475008406305714) [X0 Y1 Y2 X3]
```

Εδώ έχουμε πως το αποτέλεσμα είναι η χαμιλτονιανή που εκφράζεται στην ουσία ως ένας γραμμικός συνδυασμός των πινάκων Pauli. Μετά θα ορίσουμε τη συσκευή μας:

dev = qml.device("default.qubit", wires=qubits)

Θέλουμε γενικά να κατασκευάσουμε μια κατάσταση της μορφής:
$|\Psi(\theta)\rangle=\cos(\theta/2)|1100\rangle-\sin(\theta/2)|0011\rangle$, με το θ να είναι η παράμετρος που θα βελτιστοποιηθεί για να προσεγγίσουμε το ground state.

Ο πρώτος όρος με συντελεστή το συνημίτονο, αναφέρεται στην κατάσταση που 2 ηλεκτρόνια του μορίου βρίσκονται στις τροχιές με τη χαμηλότερη δυνατή ενέργεια και ο όρος με το ημίτονο αναφέρεται στην κατάσταση όπου 2 ηλεκτρόνια μεταφέρονται από τα qubits 0,1 στα 2,3 (συνολικά θα χρειστούμε 4 qubits). Για να κατασκευαστεί αυτή η ποσότητα, γράφουμε το εξής:

```
electrons = 2
hf = qml.qchem.hf_state(electrons, qubits)
print(hf)
```

που μας δίνει ως αποτέλεσμα:

```
[1 1 0 0]
```

και είναι το πρώτο σκέλος, το λεγόμενο και Hartree-Fock state.

Οπότε για να κατασκευαστεί και το 2ο σκέλος, χρειαζόμαστε μια επιπλέον πράξη που θα ενεργήσει και στα 4 qubits:

```
def circuit(param, wires):
    qml.BasisState(hf, wires=wires)
    qml.DoubleExcitation(param, wires=[0, 1, 2, 3])
```

Το basis state αναφέρεται στο πρώτο σκέλος και είναι αυτό που χρησιμοποιήθηκε για να έχουμε αρχικές καταστάσεις στον κβαντικό καταχωρητή. Το επόμενο βήμα είναι να οριστεί η συνάρτηση κόστους που θα υπολογίσει την αναμενόμενη τιμή της χαμιλτονιανής. Όπως έχουμε ξαναδεί, θα πρέπει αυτή η συνάρτηση να περιλαμβάνεται σε ένα γενικότερο πλαίσιο που ονομάζεται Qnode:

```
@qml.qnode(dev)
def cost_fn(param):
    circuit(param, wires=range(qubits))
    return qml.expval(H)
```

Έπειτα έχουμε την κατασκευή ενός optimizer, εδώ θα χρησιμοποιηθεί ένας Gradient Descent:

```
opt = qml.GradientDescentOptimizer(stepsize=0.4)
```

Αρχικοποιούμε το θ=0, κι άρα ξεκινάμε από το Hartree-Folc state

```
theta = np.array(0.0, requires_grad=True)
```

Η βελτιστοποίηση θα πραγματοποιηθεί σε 100 επαναλήψεις και ορίζουμε ως σωστή τη λύση που θα απέχει από την αληθή λύση όχι περισσότερο από μια ποσότητα της τάξης μεγέθους των 10^{-6}.

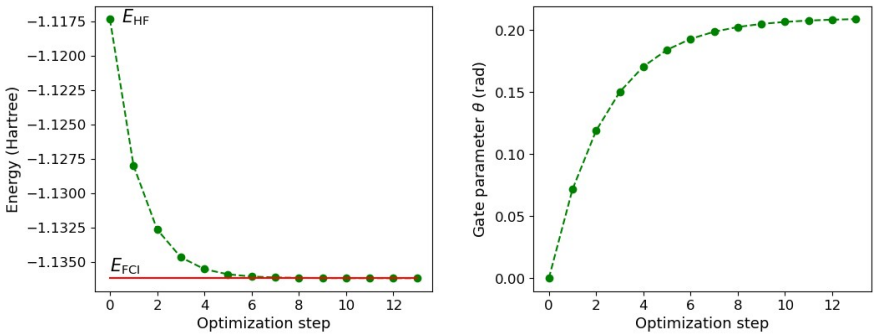
```
#αποθηκέυουμε τις τιμές της συνάρτησης κόστους
energy = [cost_fn(theta)]
# αποθήκευση τιμών της παραμέτρου
angle = [theta]

max_iterations = 100
conv_tol = 1e-06

for n in range(max_iterations):
theta, prev_energy = opt.step_and_cost(cost_fn, theta)

energy.append(cost_fn(theta))
angle.append(theta)

conv = np.abs(energy[-1] - prev_energy)

if n % 2 == 0:
print(f"Step = {n},  Energy = {energy[-1]:.8f} Ha")

if conv <= conv_tol:
break

print("\n" f"Final value of the ground-state energy = {energy[-1]:.8f} Ha")
print("\n" f"Optimal value of the circuit parameter = {angle[-1]:.4f}")
```

το αποτέλεσμα που μας δίνει αυτός ο υπολογισμός είναι:

```
Step = 0,  Energy = -1.12799983 Ha

Step = 2,  Energy = -1.13466246 Ha
Step = 4,  Energy = -1.13590595 Ha
Step = 6,  Energy = -1.13613667 Ha
Step = 8,  Energy = -1.13617944 Ha
Step = 10,  Energy = -1.13618736 Ha
Step = 12,  Energy = -1.13618883 Ha

Final value of the ground-state energy = -1.13618883 Ha

Optimal value of the circuit parameter = 0.2089
```

Το τελευταίο βήμα είναι η οπτικοποίηση του αποτελέσματος:

```
import matplotlib.pyplot as plt
fig = plt.figure()
fig.set_figheight(5)
fig.set_figwidth(12)
E_fci = -1.136189454088
ax1 = fig.add_subplot(121)
ax1.plot(range(n + 2), energy, "go", ls="dashed")
ax1.plot(range(n + 2), np.full(n + 2, E_fci), color="red")
ax1.set_xlabel("Optimization step", fontsize=13)
ax1.set_ylabel("Energy (Hartree)", fontsize=13)
ax1.text(0.5, -1.1176, r"$E_\mathrm{HF}$", fontsize=15)
ax1.text(0, -1.1357, r"$E_\mathrm{FCI}$", fontsize=15)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
ax2 = fig.add_subplot(122)
ax2.plot(range(n + 2), angle, "go", ls="dashed")
ax2.set_xlabel("Optimization step", fontsize=13)
ax2.set_ylabel("Gate parameter $\\theta$ (rad)", fontsize=13)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.subplots_adjust(wspace=0.3, bottom=0.2)
plt.show()
```



Οπότε για θ*=0.208 έχουμε την κατάσταση:

|Ψ(θ*)>=0.994|1100>-0.104|0011>

που είναι ακριβώς το ground state του μορίου του υδρογόνου H2 και το βρίσκουμε μετά από 13 επαναλήψεις.

6 Επίλογος-Συμπεράσματα

Ως μια γενική αποτίμηση, θα μπορούσαμε να αναφέρουμε πως τα λογισμικά που χρησιμοποιήσαμε μας δίνουν αρκετά ικανοποιητικά αποτελέσματα, τηρουμένων πάντα των αναλογιών, λαμβάνοντας δηλαδή υπ' όψιν πως χρησιμοποιήσαμε προσομοιωτές, η υπολογιστική ισχύς των οποίων είναι χαμηλή, και ειδικά στα παραδείγματα που παραθέσαμε, χρησιμοποιήσαμε 2 ή 4 qubits, σε πολύ απλά προβλήματα και βελτιστοποιώντας ως επί το πλείστον 1 παράμετρο. Όπως παρατηρήσαμε και στο παράδειγμα με το tensorflow-quantum, όταν έγινε η σύγκριση στο τέλος, το κλασικό νευρωνικό δίκτυο υπερτερεί κατά πολύ του κβαντικού, αν χρησιμοποιήσει εκατομμύρια παραμέτρους και λειτουργήσει βέβαια σε κλασικό υπολογιστή, ενώ αν η σύγκριση γίνεται με παρόμοιο πλήθος παραμέτρων οι διαφορές είναι ελάχιστες. Το tensorflow-quantum συγκεκριμένα κληρονομεί όλα εκείνα τα στοιχεία που χρειάζονται για εκπαίδευση νευρωνικών δικτύων από το tensorflow (οπότε ξέρουμε πως έχουμε ήδη πολλά εργαλεία στα χέρια μας για να δουλέψουμε) και στην ουσία είναι συνδυασμός αυτού και του cirq, που κατασκευάζει τα κβαντικά κυκλώματα τα οποία στη συνέχεια το TFQ βελτιστοποιεί. Δημιουργεί υβριδικά κλασικά-κβαντικά μοντέλα και δουλεύει με κβαντικά δεδομένα. Ένα αρνητικό στοιχείο είναι ότι στην πράξη συνεργάζεται μόνο με το cirq και σχεδόν με καμία άλλη βιβλιοθήκη και μια ιδιαιτερότητα είναι ότι το ιδανικό περιβάλλον για το TFQ είναι το Google Colab στο οποίο αποδίδει καλύτερα απ'ότι στα υπόλοιπα (VSCode, PyCharm, Jupyter Notebook, Spyder κ.α.).

Ως προς το qiskit, η βιβλιοθήκη της IBM είναι αρκετά πλούσια σε modules και κλάσεις που μας βοηθούν να χτίσουμε ένα απλό κβαντικό κύκλωμα γρήγορα χρησιμοποιώντας τη γλώσσα Python, μας δίνει πρόσβαση σε προσομοιωτές αλλά και σε πραγματικό κβαντικό υπολογιστή μερικών qubits δωρεάν, συνεργάζεται όπως είδαμε στα παραδείγματα και με το άλλο σπουδαίο λογισμικό εκπαίδευσης κλασικών νευρωνικών δικτύων ονόματι PyTorch, κατασκευάζοντας υβριδικό κλασικό-κβαντικό μοντέλο με κλασικά όμως δεδομένα αυτή τη φορά σε αντίθεση με

το TFQ και χρησιμοποιεί το κβαντικό κύκλωμα ως κρυφό επίπεδο του όλου μοντέλου, με τη βελτιστοποίηση στο τέλος να γίνεται κλασικά με εργαλεία του PyTorch. Συνεργάζεται και με το pennylane, που είναι το 3ο λογισμικό που είδαμε (το οποίο pennylane μπορεί και λειτουργεί και αυτό μαζί με το PyTorch).

Αυτό με τη σειρά του είναι ένα εύχρηστο λογισμικό που και αυτό χρησιμοποιεί τη γλώσσα python, συνδυάζεται με όλα σχεδόν τα λογισμικά του κβαντικού προγραμματισμού και κβαντικής μηχανικής μάθησης, είναι γρήγορο και με πολύ καλά αποτελέσματα.

Το βασικό μας πρόβλημα όμως είναι ότι δεν μπορεί να γίνει μια πλήρης σύγκριση ως προς το ποιο είναι το καλύτερο λογισμικό από τα 3, ή όποιο άλλο προϊόν της αγοράς, διότι έχουμε κβαντικούς υπολογιστές λίγων qubits και τα αποτελέσματα είναι παραπλανητικά. Δε μπορούμε να βγάλουμε ασφαλές συμπέρασμα όταν τα αποτελέσματά μας είναι εξαιρετικά και μάλιστα συγκρίσιμα με την κλασική περίπτωση αλλά η ισχύς που χρησιμοποιήσαμε είναι κάτω απο 10 qubits. Δεν ξέρουμε τι συμβαίνει σε μεγαλύτερη κλίμακα. Για να γίνει σύγκριση θα έπρεπε να έχουμε κβαντικούς υπολογιστές μεγάλης ισχύος, ή τουλάχιστον όλοι να είναι από ένα επίπεδο και πάνω (πχ ας θέσουμε ένα όριο 100 qubits). Αυτό θα μας επέτρεπε να βελτιστοποιήσουμε τους κβαντικούς αλγορίθμους (να βρούμε τυχόν κενά ή αλλαγές που πρέπει να γίνουν), να κατασκευάσουμε πιο “σωστά” κβαντικά μοντέλα για να μπορεί να γίνει η σύγκριση και μεταξύ των λογισμικών αλλά και μεταξύ κβαντικής και κλασικής περίπτωσης με πιο αξιόπιστους όρους και να επεκτείνουμε τα υβριδικά μοντέλα που μέχρι στιγμής φαίνεται να είναι το καλύτερο δυνατό μοντέλο που έχουμε. Βέβαια, πάντα θα πρέπει να σκεφτόμαστε πως ακόμη κι αν είχαμε κβαντικούς υπολογιστές με πχ 200 qubits και ήταν ανθεκτικοί σε σφάλματα, μπορεί να μην είχαν αξιοσημείωτα αποτελέσματα στο κομμάτι της μηχανικής μάθησης, σε αντίθεση με την κρυπτογραφία για παράδειγμα, που η χρήση τέτοιων συσκευών θα μπορούσε να φέρει ριζικές αλλαγές.

Ως τελικά συμπεράσματα στο ερευνητικό σκέλος, θα μπορούσαμε να πούμε πως όλος ο χώρος της κβαντικής μηχανικής μάθησης είναι ακόμα σε νηπιακό στάδιο και ως προς την έρευνα που λαμβάνει χώρα αυτή την εποχή, αλλά ειδικά ως προς την πρακτική εφαρμογή στην παραγωγή και την αγορά. Τη δεδομένη στιγμή της παρούσας εργασίας, οι όποιοι κβαντικοί υπολογιστές βρίσκονται σε κάποια πανεπιστήμια όπως το MIT και το Berkeley και σε λίγες εταιρείες, με τις

πιο βασικές να είναι η Google, η IBM και η Microsoft και βέβαια άλλες όπως η Xanadu, της οποίας το λογισμικό Pennylane χρησιμοποιήσαμε, και η χρήση τους περιορίζεται σε ερευνητικούς σκοπούς. Η ισχύς τους είναι επίσης περιορισμένη σε μερικά qubits. Ακόμη, είναι αρκετά ευάλωτοι σε σφάλματα, επηρεάζονται πολύ από το περιβάλλον που βρίσκονται, γι' αυτό και θα πρέπει να είναι προστατευμένοι σε χώρο πολύ χαμηλής θερμοκρασίας. Μην ξεχνάμε πως η έκρηξη της κλασικής μηχανικής μάθησης ήρθε με την εποχή των έξυπνων συσκευών (smartphones, tablets κλπ) και τεχνολογιών όπως το 5G, που στην ουσία έδωσε τα 2 κύρια στοιχεία που χρειάστηκε: η μεγάλη υπολογιστική ισχύς των συσκευών αυτών και η συνδεσιμότητα τους στο διαδίκτυο. Με αυτό τον τρόπο παρήχθησαν τα περισσότερα δεδομένα που είχε ποτέ η ανθρωπότητα και κατέστη δυνατόν να ελέγξουμε και να εκπαιδεύσουμε τους αλγορίθμους (όπως τα νευρωνικά δίκτυα) που τους είχαμε μόνο ως θεωρία στο χαρτί. Πιθανώς να είμαστε στον αντίστοιχο "χειμώνα" της κβαντικής υπολογιστικής όπως ήταν και τα νευρωνικά δίκτυα πίσω στις δεκαετίες του 1980 και 1990. Σε κάθε περίπτωση, τα επόμενα βήματα θα είναι να κατασκευαστούν κβαντικοί υπολογιστές ανθεκτικοί στα σφάλματα (fault-tolerant) κάτι που μάλλον θα πάρει αρκετά χρόνια ακόμα, θα έχουν μεγαλύτερη ισχύ πολλών qubits, κι έπειτα θα πρέπει να μπορούν να γίνουν εμπορικά διαθέσιμοι στο ευρύ κοινό όπως οι κλασικοί υπολογιστές, και γιατί όχι ως φορητές συσκευές στο ίσως πιο μακρινό μέλλον. Ακόμη, να σημειώσουμε πως θα ήταν χρήσιμο να ενταχθούν περισσότεροι φορείς και πανεπιστήμια παγκοσμίως για να γίνει περισσότερη έρευνα στον τομέα. Τη στιγμή αυτή φαίνεται πως δε "συμφέρει" να δαπανηθούν πόροι στις κβαντικές συσκευές και στη κβαντική μηχανική μάθηση, διότι δεν είναι κάτι που θα επιφέρει κέρδος άμεσα, η κλασική μάθηση θα κυριαρχεί για τα επόμενα χρόνια και είναι αμφίβολο γενικώς αν στο μακρινό μέλλον η κβαντική μάθηση όντως θα ξεπεράσει σε ποιότητα την κλασική. Βέβαια όπως αναφέραμε υπάρχει και η άποψη πως αναπόφευκτα θα περάσουμε στην εποχή των κβαντικών υπολογιστών, λόγω του νόμου του Moore. Οπότε μένει να αποδειχθεί στις επόμενες δεκαετίες τι μας επιφυλάσσει η Κβαντική Μηχανική Μάθηση. Τέλος, να αναφέρουμε πως θα είναι εξαιρετικό αν τελικά καταφέρουμε να αποκτήσουμε στο μέλλον και άλλες μορφές πρακτικών κβαντικών υπολογιστών όπως οι τοπολογικοί και οι ολονομικοί, πέραν των κυκλωματικών, για να μπορέσουμε να εξετάσουμε τους αλγορίθμους μάθησης και σε αυτούς.

Βιβλιογραφία

[1] https://www.weforum.org/agenda/2022/05/industrial-data-can-unlock-our-sustainable-future-heres-how/

[2] https://el.wikipedia.org/wiki/Νόμος_του_Μουρ

[3] https://quantumzeitgeist.com/richard-feynman-and-his-contributions-to-quantum-computing-and-nanotechnology/

[4] https://en.wikipedia.org/wiki/Neuromorphic_engineering

[5] https://quantumxc.com/blog/is-quantum-communication-faster-than-the-speed-of-light/

[6] https://en.wikipedia.org/wiki/Mathematical_formulation_of_quantum_mechanics

[7] https://en.wikipedia.org/wiki/Quantum_vacuum_state

[8] https://en.wikipedia.org/wiki/No-cloning_theorem

[9] Χατζησάββας, Κωνσταντίνος (2006, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ)), Μελέτη προβλημάτων για την υλοποίηση κβαντικών υπολογιστών και κβαντική πληροφορία

[10] ΚΒΑΝΤΙΚΟΙ ΥΠΟΛΟΓΙΣΤΕΣ (ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ)ΚΑΡΑΦΥΛΛΙΔΗΣ ΙΩΑΝΝΗΣ

[11] https://en.wikipedia.org/wiki/Deutsch–Jozsa_algorithm

[12] Quantum Associative Memory

Dan Ventura and Tony Martinez

Neural Networks and Machine Learning Laboratory (http://axon.cs.byu.edu)

Department of Computer Science

Brigham Young University

[13] Implementing a distance-based classifier with a quantum interference circuit Maria Schuld,1, ∗ Mark Fingerhuth,1, 2, † and Francesco Petruccione1, 3, ‡ 1Quantum Research Group, School of Chemistry and Physics, University of KwaZulu-

Natal, Durban 4000, South Africa 2Maastricht Science Programme, University of Maastricht, 6200 MD Maastricht, The Netherlands 3National Institute for Theoretical Physics, KwaZulu-Natal, Durban 4000, South Africa

[14] Quantum Neuron: an elementary building block for machine learning on quantum computers Yudong Cao,1, ∗ Gian Giacomo Guerreschi,2, † and Alán Aspuru-Guzik1, 3, ‡

[15] Supervised learning with quantum enhanced feature spaces Vojtech Havlicek1 , ∗ Antonio D. C´orcoles1 , Kristan Temme1 , Aram W. Harrow2 , Abhinav Kandala1 , Jerry M. Chow1 , and Jay M. Gambetta1 1 IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and 2Center for Theoretical Physics, Massachusetts Institute of Technology, USA (Dated: June 7, 2018)

[16] Quantum embeddings for machine learning Seth Lloyd,1, 2 Maria Schuld,2 Aroosa Ijaz,2 Josh Izaac,2 and Nathan Killoran2

[17] https://learn.qiskit.org/summer-school/2021/lec6-2-quantum-feature-spaces-kernels

[18] https://pennylane.ai/qml/glossary/quantum_feature_map.html

[19] Barren plateaus in quantum neural network training landscapes Jarrod R. McClean,1, ∗ Sergio Boixo,1, † Vadim N. Smelyanskiy,1, ‡ Ryan Babbush,1 and Hartmut Neven1 1Google Inc., 340 Main Street, Venice, CA 90291, USA (Dated: March 30, 2018)

[20] Grant, Edward, et al. "An initialization strategy for addressing barren plateaus in parametrized quantumcircuits." Quantum 3 (2019): 214.

[21] Cerezo, M., et al. "Cost-function-dependent barren plateaus in shallow quantum neural networks." *arXiv preprint arXiv:2001.00550* (2020).

[22] Huembeli, Patrick, and Alexandre Dauphin. "Characterizing the loss landscape of variational quantum circuits." arXiv preprint arXiv:2008.02785 (2020).

[23] https://quantumai.google/cirq

[24] https://pennylane.ai/qml/demos/

[25] https://qiskit.org/documentation/tutorials.html

[26] https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html

[27] https://www.tensorflow.org/quantum/tutorials/mnist

[28] https://pennylane.ai/qml/demos/tutorial_state_preparation.html#sphx-glr-demos-tutorial-state-preparation-py

[29] https://github.com/stfnmangini/VQE_from_scratch/blob/master/vqe_from_scratch.ipynb

[30] https://pennylane.ai/qml/demos/tutorial_vqe.html#sphx-glr-demos-tutorial-vqe-py

[31] Quantum Computation and Quantum Information

Michael A. Nielsen & Isaac L. Chuang

[32] Quantum Machine Learning

What Quantum Computing Means to Data Mining Peter Wittek

[33] Introduction to Quantum Information

Stephen M. Barnett

School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, UK

[34] An introduction to quantum machine learning

Maria Schulda, Ilya Sinayskiya;b and Francesco Petruccionea;b

aQuantum Research Group, School of Chemistry and Physics, University of

KwaZulu-Natal, Durban, KwaZulu-Natal, 4001, South Africa

bNational Institute for Theoretical Physics (NITheP), KwaZulu-Natal, 4001, South Africa

[35] Quantum Machine Learning in Feature Hilbert Spaces

Maria Schuld* and Nathan Killoran

Xanadu, 372 Richmond Street West, Toronto M5V 2L7, Canada

[36] Machine Learning With Quantum Computers,second edition

Maria Schuld and Francesco Petruccione

[37] Quantum Deep Learning

Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore

Microsoft Research, Redmond, WA (USA)

[38] TensorFlow Quantum:

A Software Framework for Quantum Machine Learning

Michael Broughton,1, 9, Guillaume Verdon,1, 2, 8, 10, y Trevor McCourt,1, 11 Antonio J. Martinez,1, 8, 12

Jae Hyeon Yoo,3 Sergei V. Isakov,4 Philip Massey,5 Murphy Yuezhen Niu,1 Ramin Halavati,6

Evan Peters,8, 10, 13 Martin Leib,14 Andrea Skolik,14, 15, 16, 17 Michael Streif,14, 16, 17, 18 David Von Dollen,19

Jarrod R. McClean,1 Sergio Boixo,1 Dave Bacon,7 Alan K. Ho,1 Hartmut Neven,1 and Masoud Mohseni1, z

[39] The power of quantum neural networks

Amira Abbas1,2, David Sutter1, Christa Zoufal1,3, Aurelien Lucchi3,

Alessio Figalli3, and Stefan Woerner1,*

1IBM Quantum, IBM Research – Zurich

2University of KwaZulu-Natal, Durban

3ETH Zurich

[40] PennyLane: Automatic differentiation of hybrid quantum-classical computations

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, Nathan Killoran

[41] https://nithecs.ac.za/wp-content/uploads/2020/09/September-2020-mini-school-Amira.pdf

[42] The Variational Quantum Eigensolver: a review of methods and best practices

Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, Jonathan Tennyson

[43] Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices Leo Zhou,1, ∗ Sheng-Tao Wang,1, † Soonwon Choi,1, 2 Hannes Pichler,3, 1 and Mikhail D. Lukin1 1Department of Physics, Harvard University, Cambridge, MA 02138, USA 2Department of Physics, University of California Berkeley, Berkeley, CA 94720, USA 3 ITAMP, Harvard-Smithsonian Center for Astrophysics, Cambridge, MA 02138, USA (Dated: November 12, 2019)

[44] https://en.wikipedia.org/wiki/Quadratic_unconstrained_binary_optimization

[45] Quantum support vector machine for big data classification

Patrick Rebentrost, Masoud Mohseni, Seth Lloyd

[46] Quantum principal component analysis

Seth Lloyd, Masoud Mohseni, Patrick Rebentrost

[47] https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk

[48] Quipper: A Scalable Quantum Programming Language

Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, Benoît Valiron

[49] https://github.com/ProjectQ-Framework/ProjectQ

[50] https://en.wikipedia.org/wiki/Shor%27s_algorithm

[51]https://en.wikipedia.org/wiki/Universal_approximation_theorem