SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. xxxx

PRIMJENA METODA STROJNOG UČENJA ZA KVANTNA RAČUNALA

Jan Ljubas

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. xxxx

PRIMJENA METODA STROJNOG UČENJA ZA KVANTNA RAČUNALA

Jan Ljubas

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, lipanj 2023.

ZAVRŠNI ZADATAK br. xxxx

Pristupnik:	**Jan Ljubas (0036531431)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Goran Delač

Zadatak: **Primjena metoda strojnog učenja za kvantna računala**

Opis zadatka:

Proučiti i opisati postupke strojnog učenja primjerene za kvantna računala. Poseban naglasak staviti na potporu za kvantne konvolucijske neuronske mreže. Odabrati primjereno programsko okruženje, primjerice Qiskit i u njemu pokrenuti pokazni primjer. Opisati programsko ostvarenje pokaznog primjera.

Rok za predaju rada: 12. lipnja 2023.

SADRŽAJ

Uvod

Motivacija

Poznati Mooreov zakon govori o brzini razvoja klasičnih računala – udvostručavanje računalne moći svakih ~2 godine. Budući da zakoni fizike određuju teorijske granice klasičnih čipova, predviđanja stručnjaka su da će tijekom ovoga stoljeća čovječanstvo doseći tu granicu [1] [2].

Upravo to je glavna motivacija za razvoj računala koja rade na drukčiji način od "klasičnih" računala. Jedan od koncepata računala koje pokušava zaobići fizikalne limite postavljene na klasična računala je kvantno računalo.

Unazad nekoliko desetljeća napredak u području je značajan i nada je da će kroz nekoliko godina postojati praktična, stabilna i skalabilna kvantna računala.

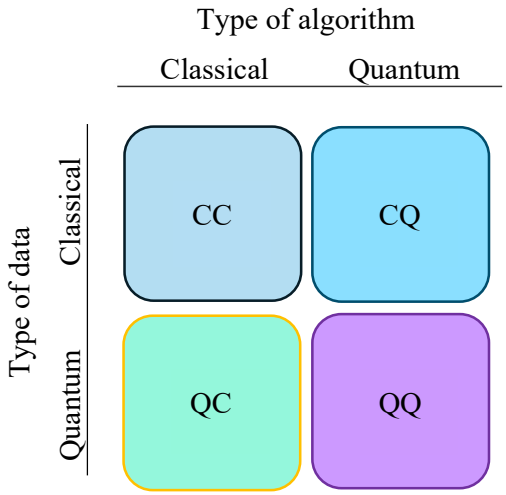
Uz tu pretpostavku znanstvenici su još krajem prošloga stoljeća počeli razvijati algoritme kojima bi na takvim računalima mogli izvoditi programe koji, korištenjem svojstava kvantne mehanike, nadmašuju i teoretski najefikasnija klasična računala. [3]

Strojno učenje i kvantna računala

Jedno zanimljivo područje primjene kvantnih računala je nadogradnja algoritama strojnog učenja. Uz trenutno razumijevanje, kvantna bi računala trebala moći nuditi 'speed-up' u određenoj vrsti problema kojoj pripadaju simulacije kvantnih fizikalnih sustava, računanje interakcija kemijskih spojeva, rješavanje problema u znanosti o materijalima i dr. [4]

Interdisciplinarno područje kvantnog strojnog učenja (QML) obuhvaća suradnju kvantnih računala i strojnog učenja iz više aspekata.

Vrlo čest prikaz tipova podataka i tipova algoritama koji se primjenjuju nad tim podatcima na koje se dijeli QML:



Slika 1: Pristupi QML-u; adaptirano prema [5]

Područje od trenutno najvećeg interesa predstavljaju polja *QC* – primjena *klasičnih* algoritama strojnog učenja nad *kvantnim* podatcima i *CQ* – primjena *kvantnih* algoritama nad *klasičnim* podatcima. Više riječi o ova dva pristupa nalazi se u Poglavlju 3.

Cilj ovog rada je predstaviti osnove o kvantnim računalima i strojnom učenju, zatim povezati te dvije ideje pri opisu *cutting-edge* tehnologija i pregleda područja kvantnog strojnog učenja.

1. Kvantna računala

Najambiciozniji i radikalno drukčiji koncept suočavanja s neizbježnim fizičkim ograničenjima računala (uključujući sva poboljšanja trenutnih, klasičnih računala) jest koncept kvantnih računala. Razlog zbog kojeg su kvantna računala tako temeljno drugačija u usporedbi s klasičnim računalima je novi, revolucionaran način obrade informacija.

Osim gledanja u (ne tako daleku) budućnost s teorijskim ograničenjima, motivacija za napredak vezan za način rada trenutnih, klasičnih računala je njihova energetska neučinkovitost.
Osim arhitekture računala, optimiranja procesa pomoću operacijskog sustava, poboljšanja učinkovitosti softvera itd., glavni uzrok neučinkovitosti je energetska disipacija osnovnih tranzistorskih sklopova. [6]

Iako postoje i zanimljive ideje koje inspiraciju crpe iz bioloških sustava, poput neuromorfnog računarstva, *kognitivnih* računala, memristorskog stohastičkog računanja i sličnih alternativa trenutnim tehnologijama, za kvantna je računala bitan razvoj teorije o reverzibilnosti informacije i reverzibilnim logičkim vratima.

Općenito, klasična su logička vrata (drugim riječima, funkcije - skup pravila koja određuju kako preslikati ulazne parametre u izlazne parametre) nereverzibilna. To znači da je na temelju izlaza iz logičkog kruga nemoguće odrediti točne vrijednosti ulaza u logička vrata. Primjer je operacija logičko I (engl. AND):

Tablica 1: Tablica istinitosti operatora AND

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

Ako je poznato da je izlaz iz logičkog kruga AB = 0, nije poznato koje su točno vrijednosti bitova A i B.

U literaturi se često spominje da je upravo ta ireverzibilnost uzrok gubitka energije i neučinkovitosti računala. Budući da najmanja teorijska količina energije jednog bita potrebna za računanje prati Landaureovo pravilo, $E > k_B * T * ln(2) \approx 3.6 \times 10^{-21}\, J$, ispostavlja se da CPU brzinu već od oko 100 GHz klasična računala ne mogu postići.

Poznato je da je reverzibilnost logičkih operacija nužan uvjet za ponovno korištenje uložene energije, odnosno smanjenje disipacije energije računala.

Stoga je i izrada klasičnih sklopova s reverzibilnim vratima tema brojnih istraživanja danas, jer trenutno ne postoje standardne arhitekture koje na jednostavan način mogu podržati isključivo reverzibilna logička vrata.

Postoje brojne sličnosti između klasičnih reverzibilnih računala i kvantnih računala – glavna je ta da se grade na principu reverzibilnih logičkih vrata. U potpoglavlju 1.2 bit će više riječi o qubitima, kvantnih logičkim vratima i njihovoj reverzibilnoj prirodi.

1.1. Povijest kvantnih računala

Osim razvoja kvantne mehanike, teorijske podloge kvantnih računala koja opisuje fizikalne pojave čiji se efekti koriste, povijest kvantnih računala veže se i uz povijest teorije informacije. Konkretno, uz odnos informacije i termodinamike te pravila fizike kojima podliježu.

Stoga je teško odrediti koji je pravi početak teorije o kvantnim računalima – Maxwellov *demon* iz 1871.g.[1], početci kvantne mehanike početkom 20. stoljeća, EPR/ER[2] znanstveni radovi i Bellov teorem iz 1964.g. koji propitkuju implikacije svojstva spregnutosti, ...

Sigurno je da su svi navedeni znanstveni pothvati bili ujedno i koraci prema daljnjim, većim otkrićima i razvoju tehnologije kvantnih računala.

[1] Poznata polemika oko informacije, entropije i kršenja 2. zakona termodinamike.
[2] Einstein, Podolsky, Rosen (EPR) autori su značajnog rada objavljenog 1931. godine u kojem provode *gedanken* eksperiment koji naizgled vodi do paradoksa vezanog za spregnutost i putovanje informacije brže od brzine svjetlosti. Zaključak autora bio je da je tadašnja teorija kvantne mehanike nepotpuna i manjkava.
Kasniji, naizgled nepovezan rad Einsteina i Rosena (ER) predstavlja rješenja (Einstein-Rosenov most) vezana za odvojeni problem u području generalne relativnosti. U znanstvenoj je zajednici desetljećima kasnije popularna postala *EPR = ER* pretpostavka, koja povezuje spregnutost, elektromagnetizam i crvotočine.

Valja istaknuti fizičare Bennetta, Fredkina i Toffolija s napretcima oko reverzibilnosti računanja (u kontekstu teorije informacije) i postavljanja okvira za kvantne logičke krugove tijekom 70-ih godina 20. stoljeća.

Često se, ipak, nobelovac Richard Feynman spominje kao osoba koja je odigrala ključnu ulogu u ranom razvoju kvantnih računala. Još 1981. godine na konferenciji američkog sveučilišta UCLA, Feynman je održao predavanje na kojem je predložio način iskorištavanja jedinstvenih i zanimljivih svojstava kvantnih objekata. Sljedeće je godine u povijesnom radu „*Simulating physics with computers"* detaljnije opisao koncepte koji danas čine temelje ideje kvantnih računala, opisujući kako bi u teoriji kvantni sustavi mogli obavljati izračune brže nego i najbrža klasična računala. [7]

Feynmanovo predavanje i prateći rad prvi su izričito razmatrali izgradnju stroja koji bi djelovao na temelju kvantno-mehaničkih principa.

Potaknuti idejama koje su predstavili Feynman i njegovi prethodnici, brojni znanstvenici iz područja fizike, matematike i računarske znanosti pokušavali su konstruirati slučaj u kojemu bi kvantni pristup računanju nadmašio mogućnosti klasičnih računala.

Prvi značajni iskorak 1985. godine napravio je fizičar David Deutsch, predstavljanjem prvog algoritma koji je demonstrirao, uz pretpostavku računala koje može računati na kvantni način, eksponencijalnu *kvantnu nadmoć* – ubrzanje koje kvantno računalo nudi u odnosu na klasično računalo. Ohrabreni takvim rezultatima, znanstvenici su sve više počeli istraživati tzv. *kvantne algoritme*; krajem 20. stoljeća nastaje područje kvantne informacijske znanosti.

Uslijedili su poznati algoritmi Bernstein-Vazirani iz 1992., protokol kvantne teleportacije, Groverov algoritam te javnosti najpoznatiji i medijski popraćeni Shorov algoritam.

Prve se fizikalne implementacije kvantnog bita pojavljuju krajem 90-ih godina korištenjem nuklearne magnetske rezonance, a početkom 21. stoljeća na razne su načine postignuta kvantna logička vrata i kvantni logički krugovi.

Kanadska tvrtka D-Wave 2011. izrađuje prvi hibridni klasično-kvantni računalni čip, IBM 2017. gradi IBM QX3 procesor sa 16 kvantnih bitova, za koji i nudi javno dostupnu web-uslugu izvršavanja kôda.

Google, IBM, Microsoft i ostali tehnološki divovi od tada predvode istraživanja vezana za kvantna računala.

1.2. Osnovna terminologija i notacija

Općeniti klasični bit predstavlja bilo koji sklop je koji se u svakom trenutku nalazi u jednom od dva moguća stanja – „0“ ili „1“. Dovoljno dug niz bitova omogućuje zapisivanje proizvoljno veliku količinu klasične informacije, a manipulacijama nad stanjima bitova obrađuju se informacije.

Gradivna jedinica kvantnih računala i najmanja jedinica kvantne informacije su kvantni bitovi – qubiti. Qubitom se smatra bilo koji sustav opisan pojavama superpozicije i spregnutosti i koji se može izmjeriti u jednom od dva stanja. Općeprihvaćena je praksa zapisivati ta stanja koristeći tzv. Diracovu ili *bra-ket* notaciju: $|0\rangle$ i $|1\rangle$.

Stanje u kojem se kvantni bit prije mjerenja nalazi je *superpozicija* tih dvaju osnovnih stanja – normirana linearna kombinacija $|0\rangle$ i $|1\rangle$. Zapisuje se na ovaj način:

$$|\psi\rangle = \lambda|0\rangle + \mu|1\rangle, \qquad |\lambda|^2 + |\mu|^2 = 1,\ \lambda, \mu \in \mathbb{C} \tag{1}$$

Matematički se objekti $|\psi\rangle, |0\rangle, |1\rangle$ zovu *ket-objekti*, a $\langle\psi|, \langle 0|, \langle 1|$ *bra-objekti*. U kontekstu kvantne mehanike i bra-ket notacije, ketovi su vektori u n-dimenzionalni vektori u kompleksnom Hilbertovom prostoru, $\mathcal{H}^{(n)}$.

Ukratko, $\mathcal{H}^{(n)}$ je prostor koji je zatvoren na unutarnji produkt i koji zadovoljava određene aksiome, poput očuvanja norme vektora (unitarnost evolucije stanja), što je pogodno za opis kvantnih pojava.

Budući da su qubiti kvantni objekti, koristi se formalizam kvantne mehanike za opis njihove stohastičke prirode.

Vjerojatnost *mjerenja stanja* $|\psi\rangle$ *u stanju* $|\phi\rangle$ jednaka je:

$$P[\psi \to \phi] = |\, a[\psi \to \phi]\,|^2 \;=\; |\,\langle\psi|\phi\rangle\,|^2\,, \tag{2}$$

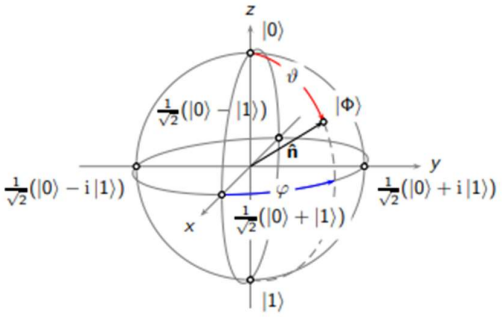
gdje je $a[\psi \to \phi]$ amplituda vjerojatnosti prijelaza sustava.

Kao što je to čest slučaj u linearnoj algebri, zbog određenih povoljnih svojstava i lakoće zapisa složenijih operacija uvijek se bira ortonormirana baza za opis linearnih kombinacija stanja qubita. Drugim riječima, u dvodimenzionalnom slučaju vrijedi da su $|0\rangle$ i $|1\rangle$

ortogonalni, odnosno da je njihov unutarnji produkt $\langle 0|1\rangle$ jednak 0. Ovo je bitna postavka za mjerenje kvantnih sustava, zato što se ovime govori da je vjerojatnost da qubit koji je u stanju $|0\rangle$ izmjerimo u stanju $|1\rangle$ jednaka nuli.

Nakon što se na ovaj način definirao vjerojatnosni sustav, korištenjem teorije vjerojatnosti i statistike lako se mogu izraziti i kompleksnije formule i svojstva qubita, poput očekivane vrijednosti mjerenja qubita ovisno o određenim parametrima, pripadajuće vjerojatnosne distribucije itd.

Treba se prisjetiti da su bra i ket objekti zapravo normirani vektori u Hilbertovom prostoru. To znači da ih se može poistovjetiti s vektorima na jediničnoj sferi kojoj su, opisujući s koordinatama u Kartezijevom 3D sustavu, točke (0, 0, 1) i (0, 0, -1) ekvivalentne stanjima $|0\rangle$ i $|1\rangle$. Mnogi se koncepti i vizualizacije rada kvantnih računala obavljaju pomoću opisane Blochove sfere. Koordinate bilo kojeg stanja sada se mogu zapisati pomoću dva kutna parametra, što olakšava matematičku notaciju i poistovjećivanje matematičkih teorema sa svijetom kvantnih računala.



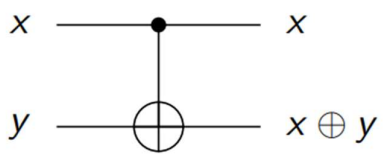
Slika 2: Prikaz Blochove sfere, preuzeto sa [8]

Budući da su svi kvantni operatori normirani, oni zapravo predstavljaju rotaciju vektora po površini Blochove sfere. S obzirom na simetričnosti i očuvanje norme (unitarnost) vezane za kvantne operatore, vrijedi da su sva kvantna logička vrata nužno reverzibilna.

Nastavno na temu reverzibilnosti klasičnih računala s početka poglavlja, treba istaknuti postojanje univerzalnih reverzibilnih logičkih vrata, koliko god ih je teško u praksi ostvariti u obliku sklopovlja. Poznato je da se cijela digitalna logika računala može ostvariti pomoću jedne vrste logičkih vrata, npr. NAND vrata. Na vrlo sličan način znanstvenici Toffoli i

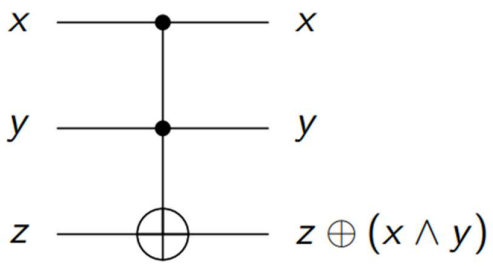
Bennett konstruirali su CNOT vrata (engl. Controlled-NOT) i CCNOT vrata (poznatija pod nazivom Toffoli vrata), kojima se može ostvariti NAND vrata.

Sva se kvantna stanja i logički operatori (logička vrata), osim u obliku tablica istinitosti, mogu izraziti i u matričnom obliku. Stanja $|0\rangle$ i $|1\rangle$ zapisuju se pomoću poznatog *one-hot* načina kodiranja: $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$, $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$.



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

Slika 3: Shema CNOT vrata i pripadajuća matrica - x je *kontrolni(upravljajući)* bit, čija vrijednost utječe na vrijednost bita y. Ako je $x = 1$, tada će se nad y izvršiti *bit-flip*.



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

Slika 4: Shema Toffolijevih vrata i pripadajuća matrica. Slično kao i CNOT, Toffolijeva vrata sadrže dva kontrolna bita i jedan bit na koji utječu

Na primjer, neka su $|a\rangle=|111\rangle$ i $|b\rangle=|110\rangle$ qubiti dimenzije 3. Vrijedi

$$\text{Toffoli}(|a\rangle) = \begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix} \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} = |b\rangle$$

Reverzibilnost nekog operatora očituje se u tome da je matrični umnožak operatora sa samim sobom jednak matrici identiteta I. Lako se izračuna da za CNOT i Toffolijeva vrata vrijedi to svojstvo.

Postojanje univerzalnih reverzibilnih vrata stoga stvara bitnu implikaciju: svaki je klasični algoritam moguće provesti korištenjem reverzibilnog logičkog kruga. [9]

U kontekstu kvantnih računala, to osigurava da se svaki klasični algoritam može provesti i na kvantnom računalu.

1.3. Kvantni algoritmi

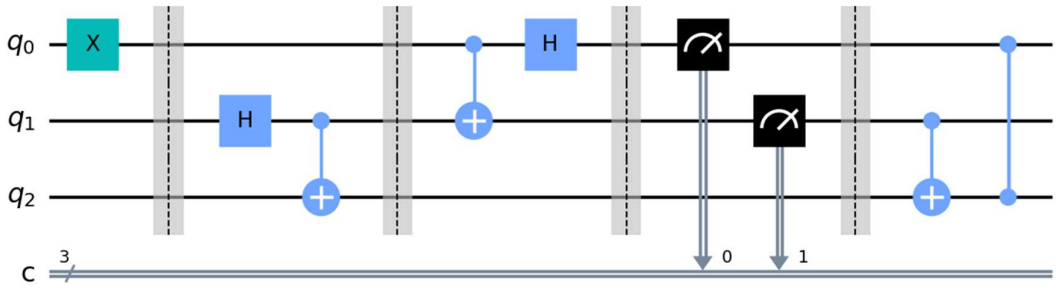
Kvantni algoritam definira se kao algoritam koji se oslanja na tehnologiju kvantnih računala – korištenje svojstava superpozicije i spregnutosti[3].

Još 90-ih godina 20. stoljeća znanstvenici su otkrili načine da u posebnim slučajevima kvantna računala barem u načelu mogu nad klasičnim računalima ostvariti prednost brzinom računanja. U ovom potpoglavlju slijedi kratki opis nekih od jednostavnih povijesno značajnih algoritama i pojašnjenje opće strukture kvantnih logičkih krugova.

Jedan od problema s kojim su se znanstvenici trebali suočiti bio je prijenos podataka sadržanih u jednom qubitu na drugi qubit – kako ostvariti *kvantni registar* koji pohranjuje podatke. Ključna spoznaja iz kvantne mehanike koja je predstavljala prepreku u tome je teorem o nemogućnosti kloniranja kvantnog stanja (engl. No Cloning Theorem):

Nije moguće stanje jednog kvantnog sustava (preciznije: kvantna informacija, parametri Schrödingerove valne jednadžbe qubita) prenijeti na drugi kvantni sustav bez da se stanje prvog sustava pritom ne promijeni. To znači da se qubit ne može izmjeriti i zatim sva njegova informacija prenijeti na drugi qubit. Srećom, na dovitljiv način istraživači su dokučili kako iskoristiti spregnutost za prijenos informacija među qubitima (bez postupka mjerenja). Procedura kojom se to ostvaruje poznata je pod nazivom protokol kvantne teleportacije.

[3] U kvantnoj informacijskoj znanosti često se koristi i termin *kvantna paralelnost* (engl. quantum parallelism), koji opisuje iskorištavanje kvantnih svojstava za izvođenje velikog broja operacija (računanje) paralelno. Kvantna paralelnost razlikuje se od pojma paralelnosti u domeni klasičnih računala.

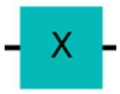
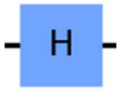
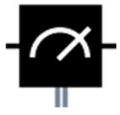


Slika 5: Shema kvantnog logičkog kruga protokola kvantne teleportacije

Cilj kvantnog logičkog kruga sa Slike 5 je prenijeti informaciju pohranjenu u qubitu *q0* na qubit *q1*. Potreban je i pomoćni qubit *q2*, a klasični registar koji je zaslužan za interpretaciju mjerenja qubita označen je s *c*.

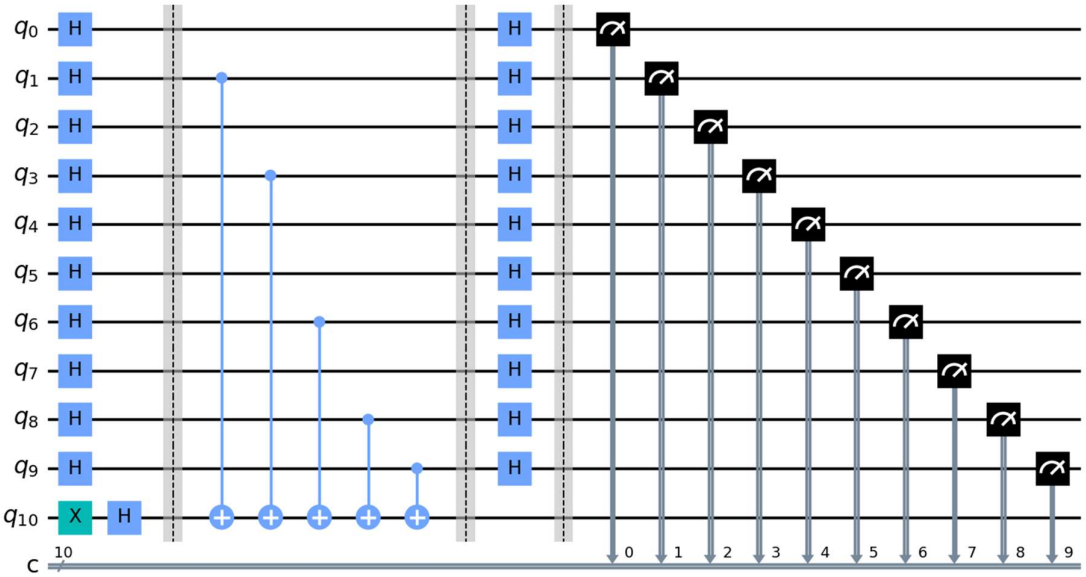
Unutar kruga nalaze se razne komponente, poput jednobitnih i višebitnih kvantnih operatora, oznaka za mjerenje i barijera (isprekidane sive linije čija je svrha isključivo preglednost pri ovakvom prikazu).

Tablica 2: Objašnjenje elemenata kvantnog kruga sa Slika 5

	**NOT vrata** Jednobitna vrata, ista funkcionalnost kao i u klasičnoj digitalnoj logici
	**Hadamardova vrata** Korištena za ostvarivanje spregnutosti
	**CNOT vrata** Gornji qubit je kontrolni qubit, a donji qubit se mijenja ovisno o vrijednosti kontrolnog qubita
	**Mjerenje** Operacija nakon koje qubit gubi kvantna svojstva

Bez ulaženja u detalje algoritma, bitno je za znati da se pomoćni qubit koristi kao spregnuti par s preostala dva qubita. U laičkim terminima to znači da se informacijama o 2-qubitnom sustavu može pristupiti bez mjerenja oba qubita. Domišljatim načinom rasporeda mjerenja i uparivanja određenih qubita moguće je u nekoliko koraka prenijeti kvantno stanje.

Razinu složeniji kvantni algoritam je Bernstein-Vazirani algoritam, nastao kao poopćenje Deutschovog algoritma. Za određenu klasu funkcija, otkrivanje informacije o tome koji je n-bitovni binarni broj predan kao parametar funkcije u klasičnom slučaju zahtijeva n procedura provjeravanja parnosti bita (po jedna za svaki bit tog broja). Bernstein-Vazirani algoritam osigurava otkrivanje iste informacije korištenjem samo 1 poziva procedure!



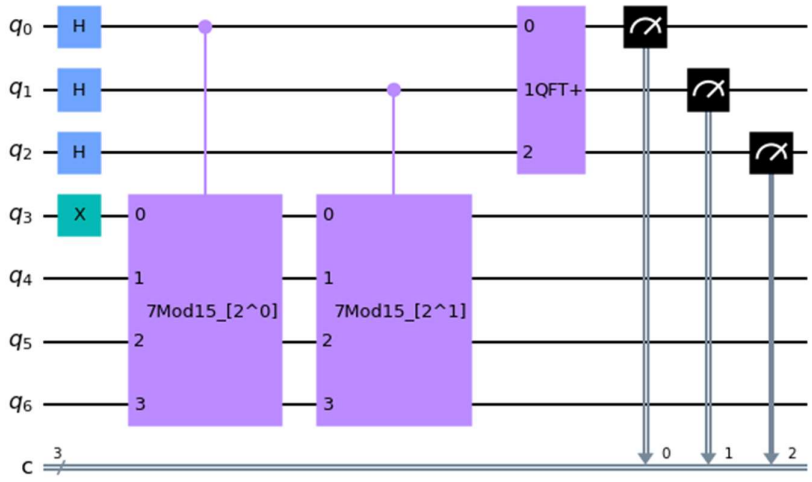
Slika 6: Shema kvantnog logičkog kruga za izvršavanje Bernstein-Vazirani algoritma u slučaju 10-bitnog broja

Kvantni krug Bernstein-Vazirani algoritma ima generalnu strukturu korištenu u mnogo drugih kvantnih algoritama: dva sloja Hadamardovih operatora nad svim mjerenim qubitima između kojih se nalazi tzv. *phase kick-back* sloj – skup operatora s kontrolnim qubitima.

Takva arhitektura omogućuje jednostavno korištenje spregnutosti i podešavanje parametara nad željenim qubitima.

Algoritam zanimljiv stručnjacima za sigurnost je slavni Shorov algoritam. Nastao je 1994. godine kao kulminacija problema diskretnog logaritma i brzog faktoriranja velikih prostih brojeva. U srži Shorovog algoritma je kvantni Fourierov transformat (QFT), kojim se postiže eksponencijalna kvantna prednost nad klasičnim algoritmima Fouriereove transformacije.

Jedan od glavnih uzroka za stvaranje sigurnosnih protokola koji koriste tzv. *post-kvantne algoritme* je upravo Shorov algoritam. Razlog tomu je taj što Shorov algoritam „ruši“ većinu algoritama asimetrične enkripcije prisutnim u današnjim internetskim protokolima – RSA enkripciju.



Slika 7: Pojednostavljena shema kvantnog kruga za izvršavanje Shorovog algoritma. Potprocedure algoritma (modularna eksponencijacija, QFT) radi preglednosti označene su ljubičastom bojom i prikazane kao *black box*

1.4. Stanje danas

Doba uporabljivih kvantnih računala i dalje je pred nama. Ono što je ohrabrujuće je tempo napretka u posljednjem desetljeću, u kojem su se veličine qubita uvišestručile (trenutni rekorder među čipovima je IBM-ov *Osprey* s 433 qubita [10]), tehnologija i pristup edukacijskim materijalima sve više je javno dostupan, a ulaganja velikih tehnoloških tvrtki i brojnih vlada svijeta u kvantnu industriju čini ju jednom od najbrže rastućih tehnoloških sektora.

Unatoč uzbudljivim mogućnostima za napredak koje nude čak i najraniji i jednostavni kvantni algoritmi, u ovom trenutku oni ne pružaju praktički nikakvu korist. Razlog tomu je taj što ljudi još nisu uspjeli stvoriti stabilna i dovoljno velika kvantna računala koja bi kvantne algoritme mogla izvršavati pouzdano.

Čak i uz nedavne napretke iz područja znanosti o materijalima i fizici supravodljivosti za koju postoji nada da će omogućiti jednostavniju i jeftiniju izradu budućih kvantnih računala, kvantna računala današnjice vrlo su osjetljivi, skupi i suviše nestabilni uređaji i za kakvu ozbiljnu primjenu. Fizičke implementacije funkcionalnih qubita i kvantnih čipova zasad su jedino moguće pri ekstremno niskim temperaturama.

Bilo kakva buka[4] iz vanjskog svijeta lako poremeti stabilno stanje kvantnih čipova, nakon čega qubiti gube sva kvantna svojstva.

Stoga je zasad moguće efikasno izvršavati samo vrlo jednostavne kvantne algoritme, s malenim brojem qubita i s malo slojeva kvantnih operatora nad svakime od njih. Više riječi o tzv. *near term & quantum efficient* algoritmima u nadolazećim poglavljima.

Pristupi rješavanju problema krhkosti kvantnih računala uključuju i *error correction*, *error mitigation*[5], korištenje tzv. *lattice* struktura (neki qubiti koriste se direktno u svrhu računanja i pohrane informacija, a dio njih u svrhu skladištenja nuspojava zbog buke) te izradu velikih, ali raspodijeljenih sustava.

[4] Izloženost većoj temperaturi, poremećaj u magnetskom polju, nečistoća materijala, ...
[5] Error correction engleski je naziv za disciplinu kojom se unazadno pokušava ispraviti pogreške kvantnih računala nastale utjecajem buke. Error mitigation postupak je kojim se uz pretpostavku linearnosti količine grešaka ekstrapoliraju rezultati kvantnog računala bez grešaka.

2. Strojno učenje

Znanost o umjetnoj inteligenciji (engl. artificial intelligence, AI) često se definira kao skup tehnika i metoda kojima ljudi pokušavaju stvoriti programe, strojeve i sustave koji djeluju *na inteligentan način*. Opisivanje inteligencije otvoreno je filozofsko pitanje i veliki izazov. Ipak, kada se govori o umjetnoj inteligenciji većini ljudi jasno je o čemu je riječ.

U ovom će se poglavlju razmatrati principi stvaranja inteligentnih strojeva, koji su pristupi doveli do današnje razine znanja i što se dalje očekuje od AI-ja.

2.1. Općenito

Početci AI-ja sežu nazad u sredinu 20. stoljeća, kada su američki znanstvenici na poznatoj konferenciji u Dartmouthu prvi predložili stvaranje umjetne inteligencije kao automatizirane programske pomoći u raznim problemima.

Strojno učenje jest način programiranja računala tako da se optimizira neki kriterij uspješnosti – ne pomoću unaprijed određenog eksplicitnog načina na koji se treba učiti, već temeljem podatkovnih primjera ili prethodnog iskustva.

Načini ostvarenja strojnog učenja kroz povijest mogu se svrstati u 5 kategorija (škola): simbolistički, konekcionistički, evolucijski, bayesovski i analogijski pristup. Svaki od pristupa temelji se na drukčijim pretpostavkama o inteligenciji i kako ju je najlakše umjetno rekreirati.

Uspjeh i popularnost svakog od pristupa često su naglo rasli ili padali povodom novih obećavajućih otkrića i napredaka određenih istraživačkih skupina[6]. Kroz posljednja 2 desetljeća pojavom vrlo uspješnih modela neuronskih mreža (posebice konvolucijskih neuronskih mreža, čija je kvantna inačica obrađena u Poglavlju 5 i Poglavlju 6) dominantno je istraživanje u području dubokog učenja, nove grane unutar strojnog učenja.

[6] Ekspertni sustavih 1960-ih, evolucijsko računarstvo 1990-ih, ...

2.2. Podjele

Danas se prema načinu učenja (a ne prema pretpostavkama o prirodi inteligencije) strojno učenje dijeli u tri grupe: nadzirano učenje, nenadzirano učenje i podržano učenje

2.2.1. Nadzirano učenje

Za ostvarenje strojnog učenja, nadzirani (engl. supervised) algoritmi koriste poznati skup podataka koji im služe kao izvor informacija o domeni unutar koje pokušavaju doći do nekih zaključaka. Podatci o kojima je riječ - skup za treniranje - uglavnom su oblika vektora s određenim brojem parametara i s vrijednošću izlaza (labela, numerička vrijednost, ...).

Ideja je da će model nakon dovoljnog broja obrađenih vektora naučiti kako nad neviđenim podatcima primijeniti pravila implicitno pohranjena u podatcima skupa za treniranje.

Prvi jednostavni modeli nadziranog učenja bili su preteče današnjih neuronskih mreža – npr. TLU perceptron. Ipak, zbog matematičkih ograničenja arhitekture takvih jednostavnih modela, ljudi su kasnije razvili kompleksnije modele koji uspijevaju riješiti širi skup problema. [11]

Dva algoritma koji se najviše spominju kada je riječ o nadziranom učenju su klasifikacija i regresija te razni izvedeni i hibridni algoritmi koji se temelje na tim idejama. U ovom radu bit će riječi o oba ova algoritma u sklopu kvantnog strojnog učenja.

2.2.2. Nenadzirano učenje

Nenadzirano (engl. unsupervised) se učenje temelji na sličnim postavkama kao i nadzirano učenje – postoji skup podataka za treniranje modela i nada je da će se nakon dovoljno pokazanih primjera desiti da model nauči neka pravila po kojima se podatci ravnaju.

Ipak, ključna je razlika ta da se u primjerima ne navodi vrijednost izlaza danog vektora, već se i to kao dodatna nepoznanica predaje modelu. Ovi su algoritmi, razumljivo, najčešće kompleksnije prirode od algoritama nadziranog učenja, zato što imaju kompliciraniji zadatak za riješiti.

Ipak, razvijeni su brojni uspješni klasični algoritmi nenadziranog učenja koji su temeljna motivacija za kvantno ostvarenje rješenja sličnih problema.

Neki popularni nenadzirani algortmi uključuju grupiranje (engl. clustering), PCA algoritam (engl. principle component analysis), GAN (engl. generative adversarial networks), VAE, ...

2.2.3. Podržano učenje

Ideja iza podržanog učenja (engl. reinforcement learning) malo se razlikuje od onih za nadzirano i nenadzirano učenje. Na temelju puno pokušaja i sustava nagrađivanja i kažnjavanja algoritmi pokušavaju naučiti optimalnu strategiju za rješenje specifičnih problema.

Modeli podržanog učenja oslanjaju se na ponavljajuću interakciju s okolinom, koja može po prirodi biti stohastička ili deterministička.

Poznatiji algoritmi podržanog učenja su Q-učenje i DDPG algoritam.

3. Kvantno strojno učenje

U svojoj najširoj definiciji, kvantno strojno učenje (engl. Quantum Machine Learning, QML) skupno je ime za sve programske pristupe koji spajaju metode strojnog učenja s kvantnom informacijskom znanošću. [12]

Podrazumijeva se da se QML zasniva na ML metodama, uz korištenje kvantnih računala kao efikasnijeg ostvarenja postojećih metoda ili proširenja mogućnosti klasičnih metoda.

Početkom 21. stoljeća pojavljuju se ideje o iskorištavanju kvantnih računala kao alata za ostvarenje ML algoritama, s nadom da se barem u teoriji njima može ostvariti prednost nad klasičnim računalima (kao i dotadašnji kvantni, ne-QML algoritmi, o kojima je bilo riječi u Poglavlju 1).

Prije opisivanja načina rada konkretnih QML algoritama potrebno je predstaviti „radni okvir“ i dodatnu terminologiju.

U ovom poglavlju obrađuju se teorija i implementacija nekih od glavnih QML algoritama. Cilj je rekreirati programski kôd i proučiti radove i materijale predstavljene na [3], a zatim komentirati dobivene rezultate.

3.1. Razlika između ML-a i QML-a

Prikaz korišten u uvodu, Slika 1, elegantan je način za pojasniti razlike između klasičnog strojnog učenja i kvantnog strojnog učenja.

CC pristup obuhvaća sve metode strojnog učenja u kojima se koriste klasični algoritmi i pretpostavlja se baratanje s klasičnim informacijama. To znači da se niti u jednom dijelu algoritma ne koriste kvantno-mehanička svojstva za povećanje efikasnosti rada algoritma. Drugim riječima, *CC* pristup je klasični pristup strojnom učenju.

Za *QC* pristup pretpostavlja se da algoritmi mogu obrađivati i klasičnu i kvantnu informaciju – skup podataka čine kvantna stanja, a ne klasični podatci. Primjer problema za koji se može koristiti *QC* pristup je direktno modeliranje (simulacija) kvantnih sustava: nema potrebe za aproksimacijom kvantnih objekata u klasičnom svijetu i pomoću trenutnih spoznaja o pravilima kvantne mehanike, već sami hardver postaje kvantni sustav koji se promatra. Nad tim se podatcima zatim pokušavaju primijeniti klasični ML algoritmi, koji ispunjuju svoju svrhu nad klasičnim podatcima. Prednost u ovom pristupu najviše se ostvaruje time da je pomoću klasičnih računala jako teško simulirati i najjednostavnije kvantne sustave – pomoću *QC* metoda zaobilazi se memorijski i računski skupi problem simulacije .

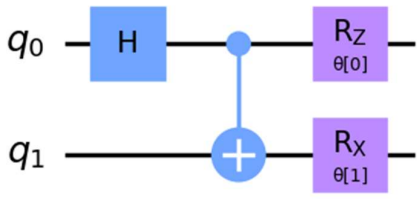
CQ pristup najviše je istražen pristup QML-u jer se najlakše formuliraju načini na koje se trenutni ML algoritmi mogu pospješiti kvantnim računalima. On pretpostavlja obrađivanje klasičnih informacija, ali uz najčešće samo ubrzan način računanja procedura i dijelova algoritma. Većina algoritama opisanih u ovom radu tradicionalno se uglavnom svrstava u *CQ* skupinu.

Posljednja skupina je mješavina *CQ* i *QC* pristupa – ulazni skup podataka s kojim se barata u *QQ* algoritmima su kvantna stanja, a metode koje se koriste za obradu tih podataka pretpostavljaju mogućnost iskorištavanja kvantnih svojstava hardvera na kojima se izvršavaju.

Valja naglasiti da često ne postoje tako strogo definirane granice kao kakve su one u prikazu na Slika 1. Ona je dobar pokazatelj u kojem se pravcu pokušava ostvariti prednost QML algoritma nad klasičnim ML algoritmom.

3.2. Parametrizirani kvantni krugovi

Parametrizirani kvantni logički krugovi (engl. parameterized quantum circuits, parameterized trial states, ansatzes) su oni kvantni logički krugovi čiji operatori i općenita arhitektura ovise o proizvoljno mnogo podesivih parametara θ_1, ... θ_n. Česta je praksa zapisivati sve parametre vektorom $\vec{\theta}$.



Slika 8: Primjer jednostavnog ansatza

```
param = ParameterVector('θ', length=2)

qc = QuantumCircuit(2)

qc.h(0)

qc.cx(0, 1)

qc.rz(param[0], 0)

qc.rx(param[1], 1)
```

Kôd 1: Isječak Qiskit kôda za izradu ovog ansatza

Razlozi za korištenje baš ovakvih krugova, *ansatza*, nije isprva očit, ali ispostavlja se da se su oni vrlo pogodni kandidati za implementaciju algoritama strojnog učenja na kvantnim računalima.

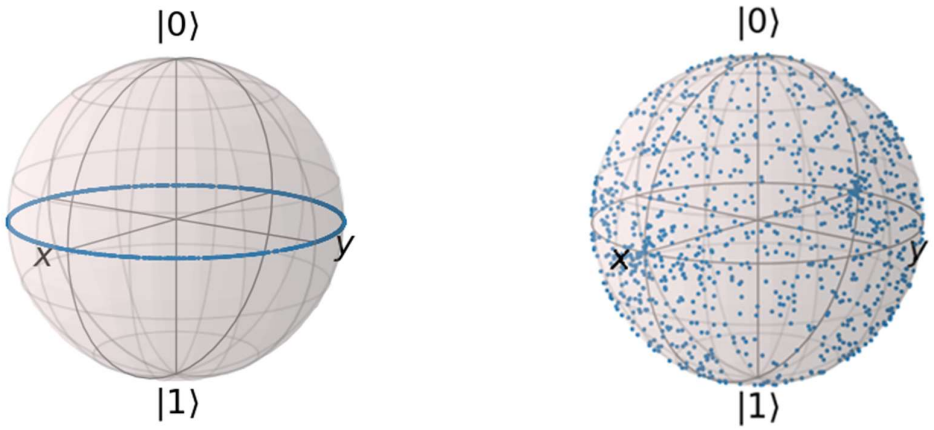
Mogućnosti koje ansatzi nude su relativno lako kodiranje klasične informacije u kvantnu informaciju (kvantno stanje) te jednostavno opisivanje kompleksnog kvantnog stanja[7].

Stručnjaci stoga danas vrlo često pri opisivanju QML algoritama pretpostavljaju korištenje ansatza, u nadi da upravo oni predstavljaju najbolju implementaciju za *near-term* kvantna računala. (referenca Maria Schuld)

Postavlja se pitanje kako odabrati ansatz za neki algoritam i postoji li uopće najbolji takav odabir? Jedan od načina za uspoređivanje ansatza nude autori iz [5], koji mjere dva svojstva ansatza - ekspresivnost i mogućnost postizanja spregnutosti.

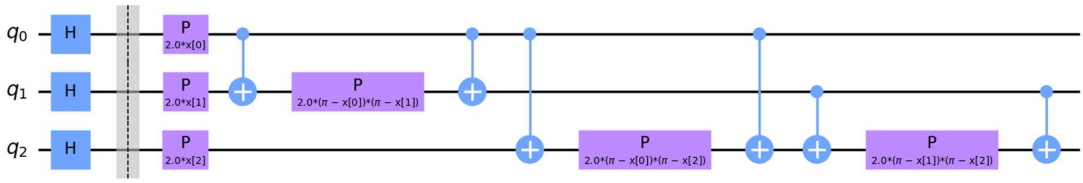
Ekspresivnost su definirali kao dio Hilbertovog prostora u kojem se kvantno stanje uopće može pojaviti, a mogućnost postizanja spregnutosti kao mjeru devijacije od uniformne distribucije kvantnog sustava.

[7] Kompleksna kvantna stanja su ona koja je teško simulirati na klasičnim računalima, ali čija je mjerenja jednostavno očitati na kvantnim računalima [3]

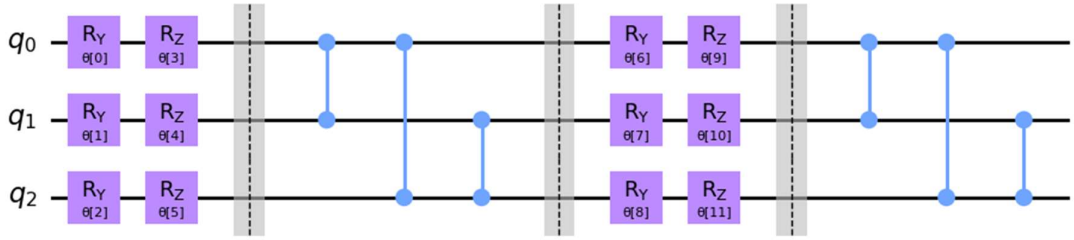


Slika 9: Prikaz na Blochovoj sferi mjerenja 1000 nasumičnih točaka dvaju stanja s različitim svojstvima – lijevo ima puno manju ekspresivnost i mogućnost postizanja spregnutosti

Jedan od ciljeva pri izradi QML algoritama je taj da se oni mogu izvoditi na računalima dostupnim u bliskoj budućnosti. Treba imati na umu da je poželjno da se algoritam može provesti u što manje slojeva kvantnih operatora i nad što manjim brojem qubita. Upravo za takve je algoritme koji prate principe efikasne izgradnje ansatza osmišljen naziv *near-term* QML algoritmi. Na Slika 10 i Slika 11: TwoLocal ansatzSlika 11 prikazane su strukture često korištene kao dijelovi u kompleksnijim QML algoritmima.



Slika 10: ZZFeatureMap ansatz



Slika 11: TwoLocal ansatz

3.3. Kodiranje podataka

Kodiranje podataka (engl. data encoding, data embedding) ključan je korak u QML algoritmima koji snažno utječe na njihovu učinkovitost (imaju li prednost nad klasičnim algoritmima ili ne).

Konkretno, za proizvoljni skup podataka X koji se sastoji od M uzoraka koji su opisani pomoću skupa parametara veličine N:

$$X = \{x^{(1)}, \dots, x^{(i)}, \dots, x^{(M)}\} \ , \quad i \in [1, M] \tag{3}$$

česta je praksa, slično kao i u slučaju klasičnog strojnog učenja, tretirati svaki od M uzoraka kao N-dimenzionalni vektor, a skup podataka X kao matricu s M redaka i N stupaca.

U klasičnim algoritmima strojnog učenja kodiranje podataka ekvivalentno je problemu zapisivanja podataka u numeričkom obliku, s obzirom na to da računala mogu računati samo pomoću brojeva.

Kvantno kodiranje podataka ipak je malo drukčije. Zadatak je kodirati numeričke podatke tako da ih kvantna računala mogu procesirati – u obliku kvantnih stanja.

Nažalost, ne postoji općenita formula za odabir najboljeg načina kodiranja podataka za pojedini QML algoritam. Optimalno kodiranje i dalje otvoreno pitanje, a trenutnu nit vodilju istraživačima najčešće predstavljaju prethodna iskustva (bez potpunog razumijevanja iza uspješnosti nekog načina kodiranja). Stoga je jasno zašto je upravo to trenutno popularna tema istraživanja u području kvantne informacijske znanosti.

Ipak, neka se kodiranja češće primjenjuju te je poznato zašto su neke tehnike učinkovitije od drugih.

U ovom se radu predstavljaju 4 tehnike kodiranja: kodiranje pomoću računalne baze, kodiranje pomoću amplitude kvantnog stanja, kodiranje pomoću kuta i arbitrarno kodiranje.

3.3.1. Kodiranje pomoću baze

Kodiranje pomoću računalne baze (engl. Basis encoding) u domeni klasičnih računala preslikava dekadske brojeve u N-bitne stringove u računalnoj bazi, dok se u slučaju kvantnih računala brojevi preslikavaju u N-qubitna stanja u računalnoj bazi.

Primjerice, broj $x = 6_{10}$ klasično se kodira u $x = 0110_2$, dok je ekvivalentno kvantno kodiranje $|x\rangle$=$|0110\rangle$.

U općenitom slučaju, funkcija kodiranja proizvoljni N-dimenzionalni uzorak iz skupa podataka, $\vec{x} = [b_1 b_2 \dots b_N]$, preslikava u kvantno stanje istih vrijednosti qubita:

$$\vec{x} \mapsto |x\rangle \ , \quad |x\rangle = |b_1 b_2 \dots b_N\rangle, \ b_i \in \{0,1\} \tag{4}$$

Nadalje, cijeli se skup podataka može izraziti kao jedno kvantno stanje kao uniformnu superpoziciju M ovakvih vektora kodiranih u kvantna stanja koristeći kodiranje pomoću računalne baze:

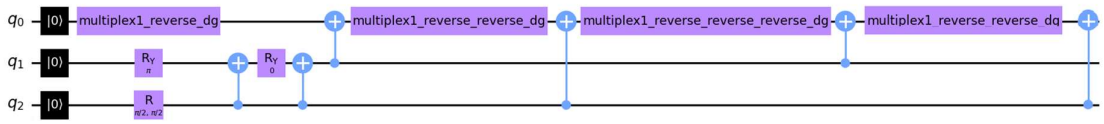
$$X \mapsto |X\rangle \ , \ |X\rangle = \frac{1}{\sqrt{M}} \sum_{i=1}^{M} |x^{(i)}\rangle \tag{5}$$

Neka je $X = \{x^{(1)} = 011, x^{(2)} = 110\}$. Kodiranjem pomoću računalne baze stvara se kvantno stanje $|X\rangle = \frac{1}{\sqrt{2}}(|011\rangle + |110\rangle)$

Uz Qiskit kôd i ugrađenu funkciju `initialize` lakše se uoče neka svojstva ove tehnike kodiranja:

```
import math
from qiskit import QuantumCircuit
desired_state = [ 0, 0, 0, 1/math.sqrt(2), 0, 0, 1/math.sqrt(2), 0 ]
qc = QuantumCircuit(3)
qc.initialize(desired_state, [0,1,2])
qc.decompose().draw(output='mpl').show()
```

Kôd 2: Kodiranje pomoću baze, adaptirano prema [3]



Slika 12: Ansatz za kodiranje pomoću baze, prema Kôd 2

Prednost ovakvog kodiranja je to što je jednostavno i lako za konstruirati u obliku ansatza. S druge strane, budući da se stvaraju raspršene matrice (engl. sparse matrix) kodiranog kvantnog stanja onda kada se u skupu podataka X ne nalaze svi vektori baze[8], ova je tehnika kodiranja izrazito memorijski neefikasna i skupa u pogledu implementacije ansatza.

3.3.2. Kodiranje pomoću amplitude

Kodiranje pomoću amplitude olakšava kodiranje realnih brojeva u složeno kvantno stanje. Ono preslikava numeričke podatke u amplitudu kvantnog stanja, što se može shvatiti kao općenita normirana težinska suma – magnituda pojedinog vektora proporcionano određuje udio utjecaja njenog qubita u složenom kvantnom stanju cijelog skupa podataka.

Stoga će se neki N-dimenzionalni vektor $x^{(i)}$ iz skupa X preslikati na ovaj način:

$$x^{(i)} \mapsto x_i \, |i\rangle \;\; , \;\; i \in [1, M] \tag{6}$$

Globalnim koeficijent w_{norm}, koji ovdje predstavlja normalizacijski koeficijent, računa se uzimajući u obzir svih N elemenata $N-$dimenzionalnog vektora $x^{(i)}$:

$$x_i = \frac{1}{\sqrt{{x_1^{(i)}}^2 + \cdots + {x_N^{(i)}}^2}} \tag{7}$$

Preslikavanje svakog od N vektora skupa X ekvivalentno je izrazu:

$$|X\rangle = \sum_i \gamma_i |i\rangle \,,$$

[8] Na primjer, u pokazanom skupu podataka X , dimenzije N = 3 i sa samo dva od 2^N = 8 mogućih vektora dimenzije N, kodiranje pomoću računalne baze nije dobar izbor

gdje je γ matrica skupa X, normirana s w_{norm}:

$$\gamma = w_{norm}\begin{bmatrix} x_N^{(1)} & \cdots & x_N^{(1)} \\ \vdots & \ddots & \vdots \\ x_N^{(M)} & \cdots & x_N^{(M)} \end{bmatrix},$$

(8)

a $|i\rangle$ je jedno od stanja računalne baze.

Najbolje je promotriti kako tehnika funkcionira na konkretnom primjeru. Ovaj put neka je $X = \{ x^{(1)} = (1.5, 0), x^{(2)} = (-2, 3), x^{(2)} = (1, -1) \}$. Vidljivo je da su N=2 i M=3.

Računa se $w_{norm} = \frac{1}{1.5^2+0^2+(-2)^2+3^2+1^2+(-1)^2} = \frac{1}{\sqrt{17.25}}$, $\gamma = \frac{1}{\sqrt{17.25}}\begin{bmatrix} 1.5 & -2 & 1 \\ 0 & 3 & -1 \end{bmatrix}$.

Na kraju, dobiva se kodirano složeno kvantno stanje

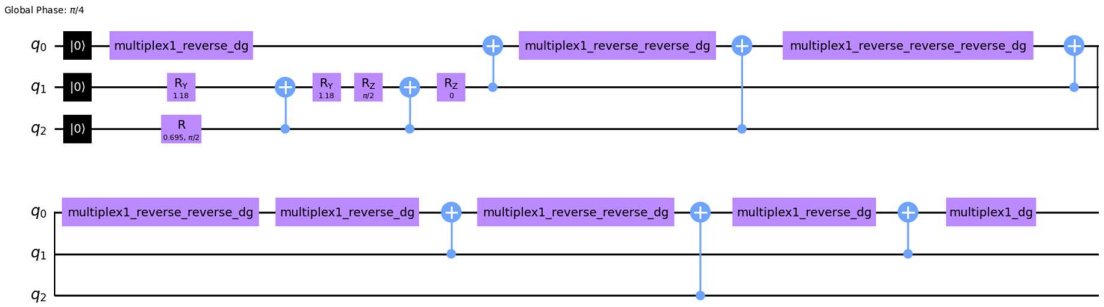
$$|X\rangle = \frac{1}{\sqrt{17.25}}[\ 1.5|000\rangle\ -\ 2|010\rangle + 3|011\rangle + |100\rangle\ -\ |101\rangle\ ].$$

Lako se pokaže da je broj qubita potrebnih za kodiranje podataka X dimenzija $N \times M$ jednak $\lceil \log_2(N \times M) \rceil$.

Ponovo se pomoću metode `initialize` s nekoliko linija Qiskit kôda može se dobiti prikaz ansatza iz primjera:

```
import math
from qiskit import QuantumCircuit
w_norm = math.sqrt(17.25)
desired_state = [1 / w_norm * 1.5, 1 / w_norm * 0,1 / w_norm * -2,
                 1 / w_norm * 3,1 / w_norm * 1,1 / w_norm * -1, 0, 0]
qc = QuantumCircuit(3)
qc.initialize(desired_state, [0 1 2])
qc.decompose().draw(output='mpl').show()
```

Kôd 3: Kodiranje pomoću amplitude



Slika 13: Ansatz za kodiranje pomoću amplitude, prema Kôd 3

Tehnika kodiranja pomoću amplitude očito je korak u dobrom smjeru. Njezina prednost u odnosu na kodiranje pomoću baze je ta što se značajno smanjio broj qubita kojima kodiramo podatke. Ipak, većina današnjih QML algoritama ne koristi niti ovu tehniku jer je čest slučaj da su metode pripreme stanja na ovaj način neefikasne.

3.3.3. Kodiranje pomoću kuta

Kodiranje pomoću kuta (engl. Angle encoding) funkcionira tako da se N elemenata vektora preslika u funkciju kuta rotacije nad parom baznih stanja $|0\rangle$ i $|1\rangle$. Na primjer, vektor $x = [x_1 \dots x_n]$ preslikat će se u :

$$|x\rangle = \bigotimes_{i \in [1,n]} \cos(x_i)|0\rangle + sin(x_i)|1\rangle \tag{9}$$

Ono po čemu se ova tehnika razlikuje od prethodne dvije je to da se funkcija kodiranja odnosi na samo jedan vektor, ne na cijeli skup podataka X.

Dobra karakteristika kodiranja pomoću kuta je malen broj potrebnih qubita – maksimalno njih N.

Kako bi se ova ideja pretočila u implementaciju s kvantnim logičkim vratima, potrebno je definirati unitarnu matricu koja odgovara operaciji kodiranja vektora x. Jedan od načina je korištenjem Y-rotacije i sljedeće unitarne matrice:

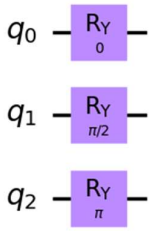
$$U(x_j^{(i)}) = \begin{bmatrix} cos(x_j^{(i)}) & -sin(x_j^{(i)}) \\ sin(x_j^{(i)}) & cos(x_j^{(i)}) \end{bmatrix}$$

$$RY(\theta) = exp\left(-i\frac{\theta}{2}Y\right) = \begin{bmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix}.$$

Tada vrijedi $U(x_j^{(i)}) = RY\left(2x_j^{(i)}\right)$ i implementacija u Qiskitu je trivijalna:

```
qc = QuantumCircuit(3)
qc.ry(0, 0)
qc.ry(2*math.pi/4, 1)
qc.ry(2*math.pi/2, 2)
qc.draw(output='mpl')
```

Kôd 4: Kodiranje pomoću kuta



Slika 14: Ansatz prema Kôd 4

3.3.4. Arbitrarno kodiranje

Arbitrarno kodiranje općenit je slučaj kodiranja pomoću kuta – ne postoji neki razlog zašto bi se koristile eksponencijalna funkcija uparena s trigonometrijskim funkcijama, a ne neke druge funkcije pogodne za rad s unitarnim matricama. Slično kao kodiranje pomoću kuta, arbitrarna tehnika kodiranja primjenjuje se na vektore skupa podataka jedan-po-jedan.

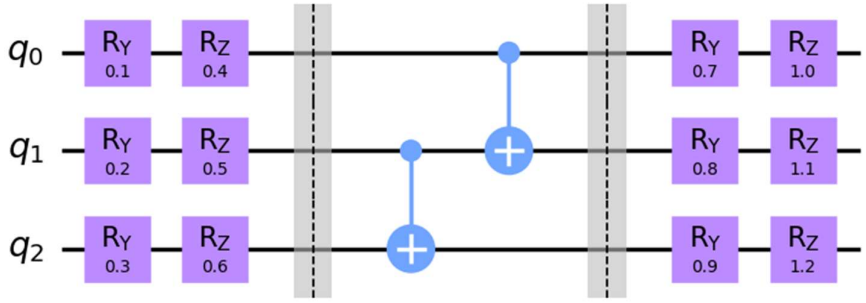
Upravo je ova tehnika najčešće korištena u istraživanjima, razvijeni su predlošci složenih ansatza koji koriste određene načine kodiranja podataka pomoću slojeva rotacija i slojeva sprezanja.

Većina predložaka uspijeva imati broj qubita n manji od dimenzije vektora N, neovisno o QML algoritmu za koji ih se koristi.

U Qiskitu su ugrađene klase efikasnih takvih ansatza, poput `EfficientSU2` ili ranije spomenuti `ZZFeatureMap`:

```
from qiskit.circuit.library import EfficientSU2
circuit = EfficientSU2(num_qubits=3, reps=1, insert_barriers=True)
x = [ 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2 ]
encode = circuit.bind_parameters(x)
encode.decompose().draw(output='mpl').show()
```

Kôd 5: Arbitrarno kodiranje koristeći predložak `EfficientSU2`

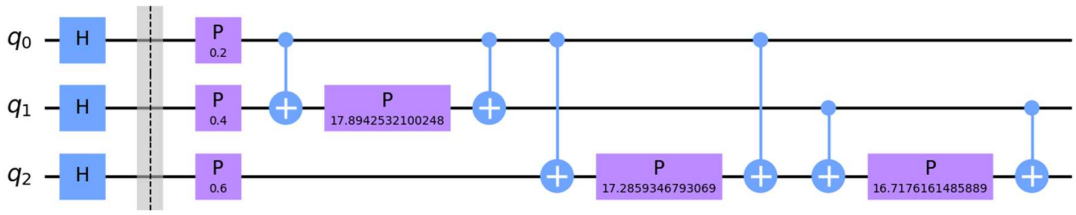


Slika 15: Predložak `EfficientSU2`, prema Kôd 5 .

Za kodiranje vektora dimenzije 12 potrebna su mu samo 3 qubita!

```
from qiskit.circuit.library import ZZFeatureMap
circuit = ZZFeatureMap(3, reps=1, insert_barriers=True)
x = [0.1, 0.2, 0.3]
encode = circuit.bind_parameters(x)
encode.decompose().draw(output='mpl').show()
```

Kôd 6: Arbitrarno kodiranje koristeći predložak `ZZFeatureMap`



Slika 16: Predložak `ZZFeatureMap`, prema Kôd 6

3.4. Gradijenti

Problem treniranja ansatza uglavnom se može svesti na minimiziranje funkcije troška i neke vrijednosti očekivanja - $\langle\psi(\vec{\theta})|\widehat{H}|\psi(\vec{\theta})\rangle$.

Postoje brojni načini (*alati*) za takvo optimiranje, no najčešći su korištenjem neke vrste gradijenta. U ovom poglavlju bit će govora o tri vrste gradijenata: obični (*vanilla)* gradijent, prirodni gradijent i SPSA gradijent (klasični i kvantni, QN-SPSA gradijent).

3.4.1. Vanilla gradijent

Jedan od najjednostavnijih načina za definiranje gradijenta kojim postižemo globalni minimum za što veći broj (potencijalno kompleksnih) funkcija troška je tzv. *finite difference gradient* model, još nazivan i *vanilla* gradijent, zato što predstavlja običan slučaj matematičkog gradijenta. Ako želimo procijeniti nagib funkcije f u točki $\vec{\theta}$, ovaj pristup nalaže da se linearno aproksimira gradijent pomoću točaka $f(\vec{\theta}+\epsilon)\ i\ f(\vec{\theta}-\epsilon)$:

$$\vec{\nabla} f(\vec{\theta}) = \frac{1}{2\epsilon}\Big(f(\vec{\theta}+\epsilon) - f(\vec{\theta}-\epsilon)\Big) \tag{10}$$

Brzina učenja također se može kontrolirati koeficijentom η (engl. termin je *learning rate*), često s vrijednosti između 0.01 i 0.1.

Nažalost, ispostavlja se da je ovaj pristup sporiji i osjetljiviji na velik broj funkcija troška.

3.4.2. Prirodni gradijent

Prirodni je gradijent sličan vanilla gradijentu, samo što se u obzir pri računanju uzima da parametri funkcije ne moraju jednako utjecati na njenu vrijednost. Tako je, na primjer, jednostavna linearna funkcija $f(x) = 100x + y$ mnogo osjetljivija na peturbacije u varijabli x. Jedan način kako uračunati različit utjecaj varijabli je korištenjem Jakobijanovih matrica koje početni Kartezijev koordinatni sustav preslikavaju u skalirani koordinatni sustav prikladan funkciji troška.

Kvantna verzija ovakvog gradijenta računa se pomoću sljedećeg izraza:

$$g_{ij}(\vec{\theta}_{n+1}) = \vec{\theta}_n - \eta\, g^{-1}\, \vec{\nabla} f(\vec{\theta}_n)\,,$$

(11)

gdje je g^{-1} tzv. kvantna Fisherova informacija – funkcija koja osigurava skaliranje prema Jakobijanovim transformacijama i usklađenom računanju ne-Euklidske udaljenosti.

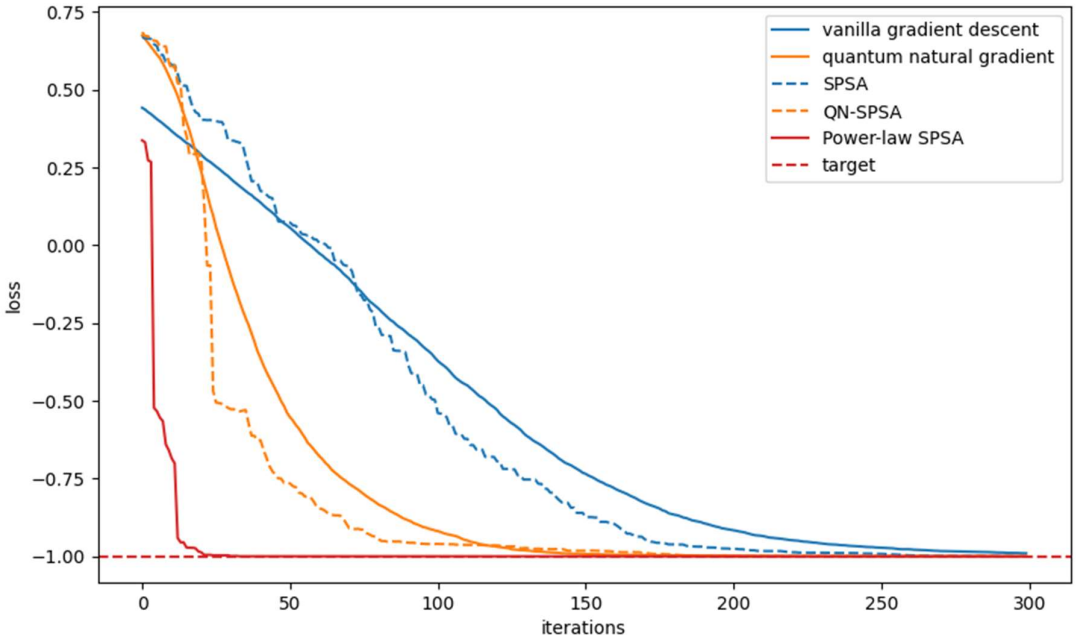
Kasnije će se pokazati da je i u praksi ovakav gradijent efikasniji od vanilla gradijenta i da je to dobar smjer pri unaprijeđenju gradijenta.

3.4.3. SPSA gradijent

SPSA gradijent (engl. Simultaneous Perturbation Stochastic Approximation) efikasan je gradijentni algoritam koji koristi stohastičke modele za osiguravanje konvergencije funkcije greške u minimum, ali na računski puno manje zahtjevan način. Umjesto računanja svakog koraka gradijentnog spusta, SPSA gradijent uzorkuje promjene i za danu razinu pouzdanosti statistički određuje sljedeći korak spusta.

Matematički je dokazano da će za aproksimativnu SPSA metodu uz slučajno uzorkovanje treniranje modela biti jednako uspješno kao i za prirodni gradijent, ali uz kvadratno ubrzanje.

U nekim se slučajevima, ovisno o ansatzu, mogu pretpostaviti neke karakteristike funkcije greške. Stoga se tada znaju koristiti i određene moifikacije na brzinu učenja, tzv. *power-rule* metode.



Slika 17: Rezultat treniranja i usporedba gradijenata

Makar su QN-SPSA i power-law SPSA uvjerljivo najbrži i najefikasniji gradijenti, trenutno se i dalje najčešće korise kvantni prirodni gradijent ili obični SPSA, budući da su prikladniji *near-term* kvantnim računalima.

4. Nadzirani QML

Nadzirani QML algoritmi temelje se na sličnim principima kao i klasični nadzirani ML algoritmi – ideja je da se model uči nad unaprijed označenim primjerima iz kojih se statističkim metodama dovode zaključci i pomoću koje model kasnije predviđa određene vrijednosti nad neviđenim podatcima.

Tim kanadskih znanstvenika predvođen Mariom Schuld [12] pokazao je da se modeli nadziranih kvantnih ML algoritama u velikom slučaju temelje na tzv. *kvantnim jezgrenim metodama* (engl. quantum kernel methods) s *kvantnim kernelima* (engl. quantum kernels). Kao posljedica toga matematički je dokazano da se ovakvi modeli mogu trenirati minimiziranjem relativno jednostavno konstruirane funkcije troška i da će uspješnost modela najviše ovisiti o koraku kodiranja podataka.

U nastavku poglavlja obradit će se poznati nadzirani QML algoritam varijacijske klasifikacije, a zatim će biti i riječi općenito o temeljima nadziranog QML-a i spomenutim kvantnim jezgrenim metodama i kvantnim kernelima.

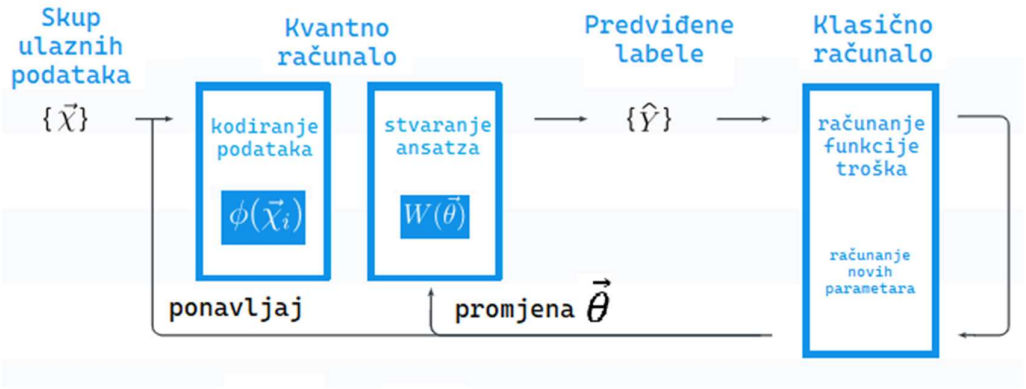
4.1. Varijacijska klasifikacija

Varijacijski kvantni algoritmi prvi su put predstavljeni 2014 u više radova vezanih za *quantum eigensolver* i algoritam aproksimativne kvantne optimizacije. Oni pripadaju skupini *near-term* algoritama, za koje se vjeruje da bi mogli biti rekreirani na stvarnim računalima u bliskoj budućnosti.

Korištenjem parametriziranog kvantnog kruga $U(\vec{\theta})$, priprema se stanje $|\psi(\phi)\rangle = U(\vec{\theta})|0\rangle$. To znači da se nad skupom ulaznih qubita koji su svi u početnom stanju $|0\rangle$ primijenio operator kvantnog logičkog kruga, $U(\vec{\theta})$.

Nakon toga se pomoću kvantnog računala *mjeri* očekivanje $\langle\psi(\vec{\theta})|\hat{H}|\psi(\vec{\theta})\rangle$. Sljedeći je korak algoritma, najčešće korištenjem klasičnog računala, izračunati pred-definiranu funkciju troška $C(\vec{\theta})$ – određivanje koliko je uspješan skup parametara $\vec{\theta}$.

Cilj je pronaći skup $\vec{\theta}$ takav da je vrijednost funkcije troška (engl. cost function) $C(\vec{\theta})$ minimalna.



Slika 18: Shema općenitog oblika varijacijskih algoritama, adaptirano prema [3]

U nastavku se opisuje implementacija s priloženim isječcima programskog kôda i prikazuju se rezultati rada algoritma.

Kao što je to i slučaj s klasičnom klasifikacijom, varijacijski kvantni klasifikator (eng. variational quantum classifier, VQC) uči na temelju labela iz vektora u skupu podataka za treniranje. Algoritam radi procjenu granica klasa hiper-ravnine[9] te pokušava ispravno svrstati svaki vektor u klasu iste labele.

Radi jednostavnosti pri testiranju implementacije, skup podataka bio je *ad-hoc* generiran zadani broj točaka u 2D ravnini, s pripadne dvije klase. Koristila se Qiskitova ugrađena funkcija `ad_hoc_data` za generiranje takvih podataka. Također, kodiranje je odrađeno pomoću `ZZFeatureMap` i `TwoLocal` ansatza, a za pokretanje algoritma postoji ugrađena `VQC` funkcija.

Radi usporedbe rezultata, prvi skup podataka sadrži 20 vektora u skupu za treniranje i 10 u skupu za testiranje. Drugi je skup veći, sa 700 vektora u skupu za treniranje i 300 u skupu za testiranje.

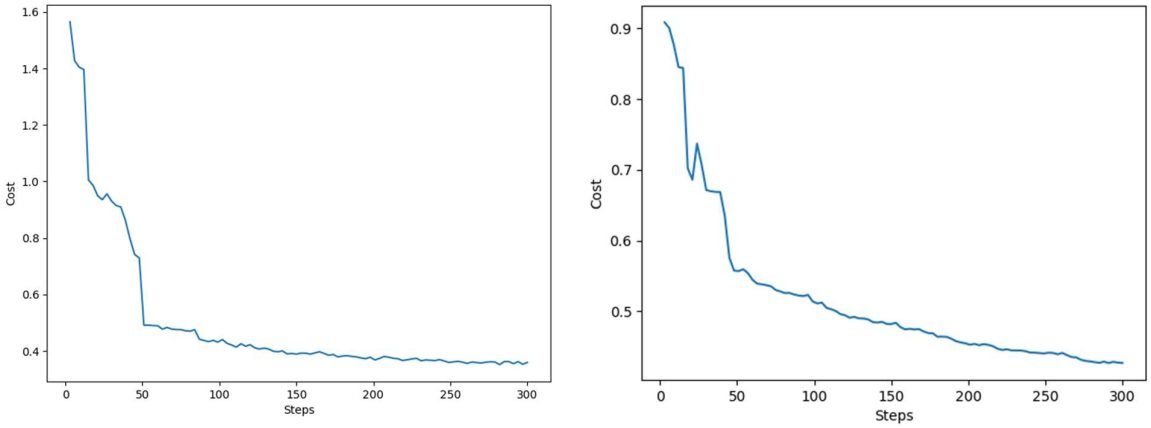
[9] Podprostor za jednu dimenziju manji od originalnog prostora. Npr. ako je skup podataka trodimenzionalan, pokušava se napraviti klasifikacija pomoću granice koja je dvodimenzionalna ravnina.

```
from qiskit.utils import algorithm_globals
algorithm_globals.random_seed = 1558

import numpy as np
np.random.seed(algorithm_globals.random_seed)

from qiskit_machine_learning.datasets import ad_hoc_data
TRAIN_DATA, TRAIN_LABELS, TEST_DATA, TEST_LABELS =
(ad_hoc_data(training_size=20, test_size=10, n=2, gap=0.3,one_hot=False))

FEATURE_MAP = ZZFeatureMap(feature_dimension=2, reps=2)
VAR_FORM = TwoLocal(2, ['ry', 'rz'], 'cz', reps=2)

AD_HOC_CIRCUIT = FEATURE_MAP.compose(VAR_FORM)
AD_HOC_CIRCUIT.measure_all()

encoder = OneHotEncoder()
train_labels_oh =
      encoder.fit_transform(TRAIN_LABELS.reshape(-1,1)) .toarray()
test_labels_oh =
      encoder.fit_transform(TEST_LABELS.reshape(-1,1)).toarray()

vqc = VQC(feature_map=FEATURE_MAP,
          ansatz=VAR_FORM,
          loss='cross_entropy',
          optimizer=SPSA(callback=log.update),
          initial_point=initial_point,
          quantum_instance=BasicAer.get_backend('qasm_simulator'))
```

Kôd 7: Isječci VQC algoritma, slučaj manjeg skupa podataka (adaptirano prema [3])

Nakon 300 iteracija algoritma u

Kôd 7, u oba je slučaja postignuta točnost veća od 90%, s tim da je uspješniji model treniran nad većim skupom podataka.



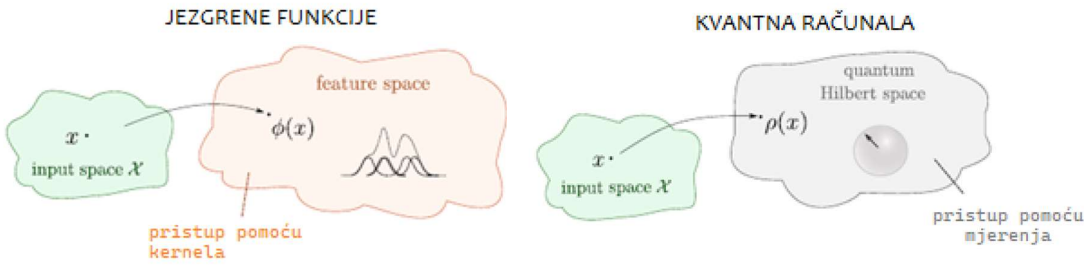
a) b)

Slika 19: VQC cost funkcija nakon 300 koraka algoritma koristeći SPSA gradijent:
a) nad malenim skupom ad-hoc podataka (20 + 10)
b) nad većim skupom ad-hoc podataka (700 + 300)

Slika 20: Točnost od 90% na skupu podataka sa Slika 19 a)

Slika 21: Točnost 99.5% na podatcima sa Slika 19 b)

4.2. Kvantne jezgrene funkcije



Slika 22: Princip rada kvantnih računala sličan je radu kvantnih jezgrenih funkcija, adaptirano prema [12].

Na Slika 22 prikazana je usporedba kvantnih računala s jezgrenim funkcijama (engl. quantum feature maps). Matematički radni okviri obiju klasa metoda temelje se na preslikavanju informacija u više-dimenzionalni prostor. Jezgrene funkcije za pristup tom *feature* prostoru služe se kernelima – matricama unutarnjeg produkta određenih vektora.

Na sličan se način pri radu s kvantnim računalima obrada informacija u više-dimenzionalnom Hilbertovom prostoru koriste mjerenja kvantnih sustava – unutarnji produkti kvantnih stanja tih sustava.

Dakle, za brojne QML algoritme koriste se kvantne jezgrene funkcije oblika

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \langle f(\overrightarrow{x_i}), f(\overrightarrow{x_j}) \rangle \,,$$

(12)

gdje je k jezgrena funkcija s argumentima n-dimenzionalnim vektorima, f predstavlja funkciju koja transformira n-dimenzionalne vektore u m-dimenzionalne vektore.

Prema tome, izraz (12) može se zapisati i pomoću $n \times m$ matrice $K_{i,j}$.

U QML literaturi običaj je da se kvantne jezgrene funkcije koriste za preslikavanje klasičnog vektora $\vec{x}$ u vektor u pripadajućem Hilbertovom prostoru, $|\phi(\vec{x})\rangle \langle\phi(\vec{x})|$. Za to se preslikavanje na praktičan način može koristiti ansatz opisan unitarnom transformacijom $U_\Phi(\vec{x})$.

U Qiskitu postoji ugrađena klasa `PauliFeatureMap` kojom je opisana obitelj jezgrenih funkcija koje imaju povoljna svojstva – mogu se ostvariti u malom broju slojeva qubita i teško ih se može simulirati pomoću klasičnih računala.

Opća formula za opis ovakvih funkcija dubine d je

$$\mathcal{U}_{\Phi(\vec{x})} = \prod_d U_{\Phi(\vec{x})} \, H^{\otimes n} \,,$$

(13)

dok je $U_{\Phi(\vec{x})}$ unitarni operator koji opisuje blok s Pauli matricama I, X, Y ili Z:

$$U_{\Phi(\vec{x})} = exp\left(i \sum_{S \subseteq \lceil n \rceil} \phi_S(\vec{x}) \prod_{k \subseteq S} P_i \right).$$

(14)

n je broj koji označava prosječnu povezanost jednog qubita s drugim qubitima, odnosno mjeru spregnutosti sustava (postoji $\binom{n}{k}$ kombinacija, $k = 1, 2, \cdots, n$.

Formula (13) predstavlja ansatz koji se sastoji od skupa d slojeva Hadamardovih operatora sa operatorom (slojem) $U_{\Phi(\vec{x})}$.

Za specifičan slučaj $k = 2$ i $d = 3$, a $P_i = Z$ Pauli obitelji jezgrenih funkcija je (s razlogom) često korišten ansatz `ZZFeatureMap`

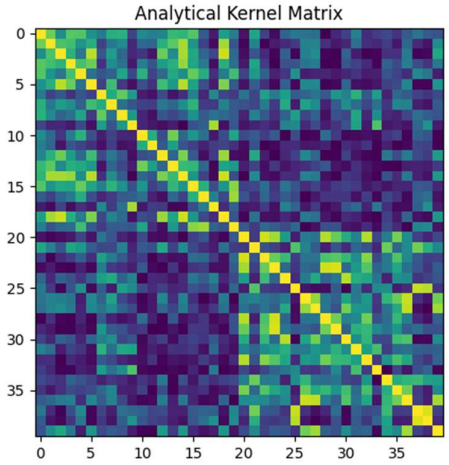
4.3. Kvantni kerneli

Klasični kerneli često se koriste kao mjera usklađenosti za odabir ispravne jezgrene funkcije za dani zadatak koji model pokušava naučiti. Ono što kernele čini efikasnima je to da se podatci mogu transformirati u prostor veće dimenzije, nad kojima se mogu provoditi brze, linearne operacije za određivanje kompleksnih granica klasa.

Na sličan način kvantni kerneli (drugim imenom kvantni filteri) funkcije su definirane nad dvama kvantnim stanjima (npr. iz skupa podataka za treniranje) koji računaju sličnost između ta dva stanja. Često se upravo za tu mjeru sličnosti uzima amplituda vjerojatnosti za prijelaz iz jednog stanja u drugo:

$$K_{i,j} = \left| \langle \phi^{\dagger}(\vec{x}) | \phi(\vec{x}) \rangle \right|^2 = \left| \left\langle \phi^{\dagger}(\vec{x}) \middle| U_{\Phi(\vec{x_i})} U_{\Phi(\vec{x_j})} \middle| \phi(\vec{x}) \right\rangle \right|^2$$

(15)



Slika 23: Prikaz jednog normiranog kvantnog kernela – žutom bojom označene su vrijednosti bliže 1, dok su plavom bojom obojane vrijednosti bliže 0. Jasno je da su uz glavnu dijagonalu vrijednosti jednake 1: amplituda prijelaza iz stanja u samo sebe jednaka je 1. Matrica je također simetrična jer je amplituda vjerojatnosti jednaka za prijelaze $|x\rangle \rightarrow |y\rangle$ *i* $|y\rangle \rightarrow |x\rangle$.

5. Kvantne neuronske mreže

Umjetne neuronske mreže (eng. *Artificial Neural Networks*, *ANNs*) uvjerljivo su najpopularnija bazna arhitektura u današnjim algoritmima strojnog učenja, sa zapanjujuće širokim područjem primjene i uspjehom u raznim problemima strojnog učenja.

Općenita struktura neuronskih mreža sadrži ulazni sloj, određen broj tzv. skrivenih slojeva i izlazni sloj mreže.

Inspirirani ljudskim mozgom i povezanošću bioloških neurona, gradivnu jedinicu klasične neuronske mreže čine umjetni neuroni.

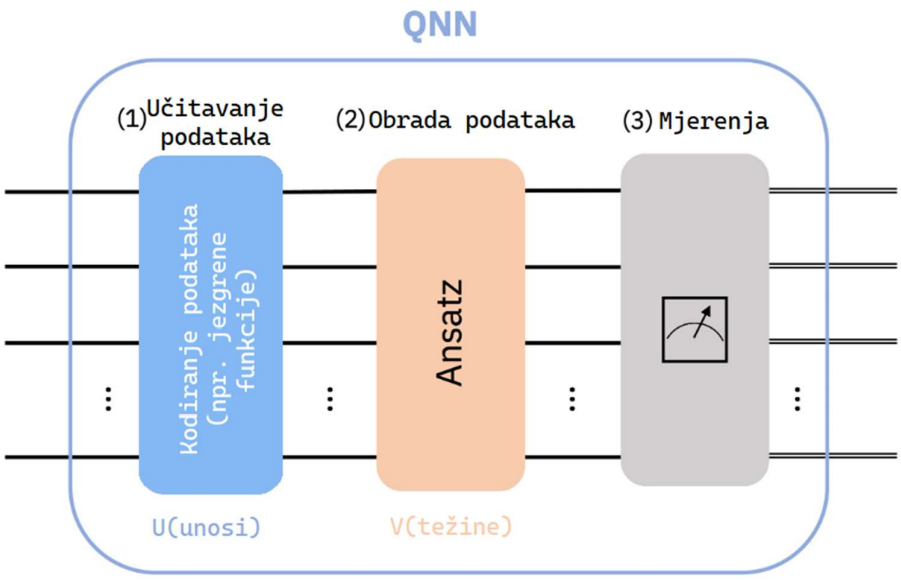
Ideja je da umjetni neuroni i značajnost veza među njima omogućuju pohranu informacije o modelu i na koji način tretirati ulazne podatke za uspješno rješavanje problema.

Neuronske mreže s velikim brojem skrivenih slojeva nazivaju se duboke neuronske mreže, a izrazita uspješnost tih modela razvila je zasebni ogranak strojnog učenja koji se naziva duboko učenje (engl. deep learning). Egzotičnije nadogradnje obuhvaćanju i memoriziranje stanja neurona, tzv. RNN mreže.

Kvantne neuronske mreže (engl. *Quantum Neural Network, QNNs*) motivaciju vuku iz klasičnih umjetnih neuronskih mreža, ali se ipak značajno razlikuju po idejnoj arhitekturi i programskom ostvarenju.

Za razliku od klasičnih ANN-ova, QNN arhitektura ne temelji se ni na kakvoj gradivnoj jedinici poput umjetnog neurona, ali ona primijenjuje ideje poput slojevite strukture i iterativnog mijenjanja težina među slojevima.

Standardni pristup ostvarenju QNN-ova ponovno je parametrizirani kvantni logički krug, tj. ansatz.



Slika 24: Shema standardne QNN arhitekture, adaptirano prema [13]

Budući da QNN-ovi kombiniraju ideje iz strojnog učenja i kvantnih računala, može ih se opisati iz dva različita kuta gledanja:

Iz perspektive strojnog učenja, QNN su modeli koji se mogu trenirati na sličan način kao i klasične ANN – u prvom se koraku pohranjuju dobiveni podatci, zatim se težine na određeni način mogu podešavati (i to čini temelj procesa učenja), što čini obradu podataka. U zadnjem se koraku mjeri učinak trenutnog skupa parametara (težina u mreži), na temelju kojeg se optimizacijskim algoritmom iterativno dolazi do najboljeg takvog skupa parametara.

Iz perspektive kvantnih računala, QNN su kvantni algoritmi čiji se rad temelji na parametriziranim kvantnim logičkim krugovima. Ti algoritmi mogu biti varijacijski algoritmi, trenirani pomoću klasičnog ili kvantnog optimizatora. Cjelokupna se mreža sastoji najčešće od jezgrene funkcije (koja je zaslužna za kodiranje ulaznih podataka) i ansatza čiji se parametri mijenjaju.

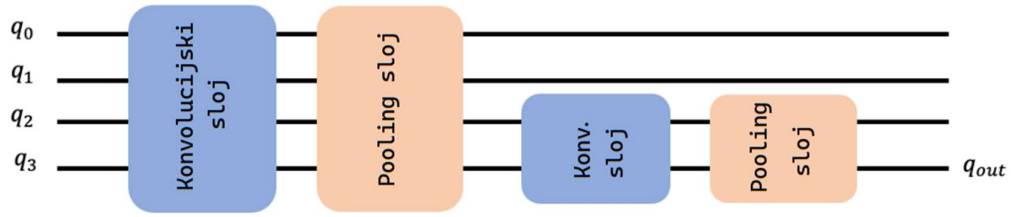
U sklopu Qiskita razvijene su i ugrađene klase i funkcije za programsku potporu stvaranju QNN-a (npr. `NeuralNetwork`, `EstimatorQNN`, `SamplerQNN`, ...). Valja istaknuti nadzirane QML algoritme koji se najlakše ostvaruju pomoću QNN-a: kvantni regresor i kvantni klasifikator.

U nastavku poglavlja bit će riječi o kvantnim konvolucijskim neuronskim mrežama.

5.1. Konvolucijske QNN

Kvantne konvolucijske neuronske mreže, (engl. *QCNNs*) slične su arhitekture kao i klasične konvolucijske neuronske mreže. Njihova je primjena većinski u području obrade digitalnih informacija, npr. prepoznavanje objekata na slikama, segmentacija slika , računalni vid, analizi medicinskih slika, ...

U slučaju klasičnih konvolucijskih neuronskih mreža, početni zadatak je vektorski prikazati sliku – pomoću n piksela duljine m, odnosno matrice $n \times m$. Često se pomoću raznih heurističkih i statističkih postupaka uklanjaju redundantne informacije na slikama, a piksele se promatra kao slučajne varijable.



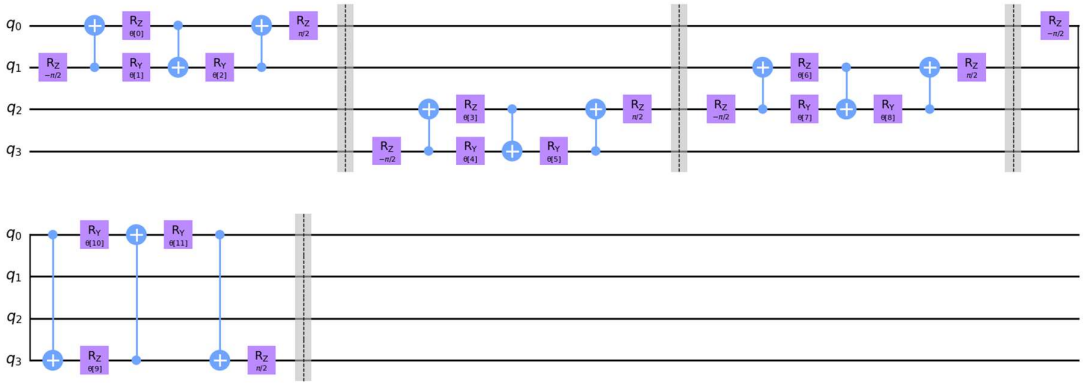
Slika 25: Shema arhitekture konvolucijskih kvantnih neuronskih mreža.
Konvolucijski sloj pomoću kernela prolazi po ulaznim vektorima, a *pooling* sloj za glavni zadatak ima smanjiti dimenziju dobivenih podataka pomoću združivanja informacija susjednih piksela.

Ono što razlikuje konvolucijske NN od ostalih NN modela upravo je korištenje barem jednog konvolucijskog sloja u skrivenim slojevima mreže. Ideja je da se korištenjem matematičke operacije konvolucije ugrađene u arhitekturu mreže uštedi na algoritamskom načinu učenja modela o informacijama pohranjenim u obliku slike.

To je moguće jer je konvolucija operacija invarijantna na pomak i većinu prostornih rotacija. To znači da npr. konvolucijski model koji zna prepoznati mačku neće imati problema s prepoznavanjem mačke koja se nalazi na desnoj strani slike niti s mačkom koja se nalazi na lijevoj strani slike.

```
def conv_layer(num_qubits, param_prefix):

    qc = QuantumCircuit(num_qubits, name="Convolutional Layer")
    qubits = list(range(num_qubits))
    param_index = 0
    params = ParameterVector(param_prefix, length=num_qubits * 3)

    for q1, q2 in zip(qubits[0::2], qubits[1::2]):
        qc = qc.compose(
            conv_circuit(params[param_index : (param_index + 3)]),
            [q1, q2] )
        qc.barrier()
        param_index += 3

    for q1, q2 in zip(qubits[1::2], qubits[2::2] + [0]):
      qc = qc.compose(
            conv_circuit(params[param_index : (param_index+3)]),
            [q1,q2] )
        qc.barrier()
        param_index += 3

    qc_inst = qc.to_instruction()

    qc = QuantumCircuit(num_qubits)
    qc.append(qc_inst, qubits)
    return qc
```

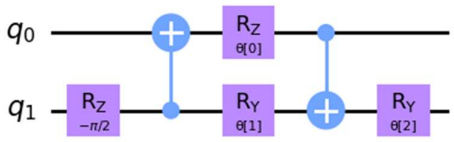
Kôd 8: Programsko ostvarenje konvolucijskog sloja – metoda `conv_layer`



Slika 26: Shema jednog konvolucijskog sloja

```
def pool_circuit(params):
    target = QuantumCircuit(2)
    target.rz(-np.pi / 2, 1)
    target.cx(1, 0)
    target.rz(params[0], 0)
    target.ry(params[1], 1)
    target.cx(0, 1)
    target.ry(params[2], 1)

    return target
```

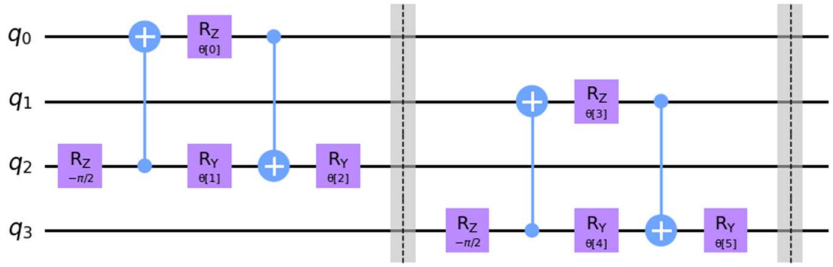
Kôd 9: Ansatz za jedan dio *pooling* sloja



Slika 27: Shema dijela *pooling* sloja

```
def pool_layer(sources, sinks, param_prefix):
    num_qubits = len(sources) + len(sinks)
    qc = QuantumCircuit(num_qubits, name="Pooling Layer")
    param_index = 0
    params = ParameterVector(param_prefix, length=num_qubits // 2 * 3)
    for source, sink in zip(sources, sinks):
        qc = qc.compose(
            pool_circuit(params[param_index : (param_index + 3)]),
            [source, sink] )
        qc.barrier()
        param_index += 3
    qc_inst = qc.to_instruction()
    qc = QuantumCircuit(num_qubits)
    qc.append(qc_inst, range(num_qubits))
    return qc
```

Kôd 10: Programsko ostvarenje *pooling* sloja – metode `pool_circuit` i `pool_layer`



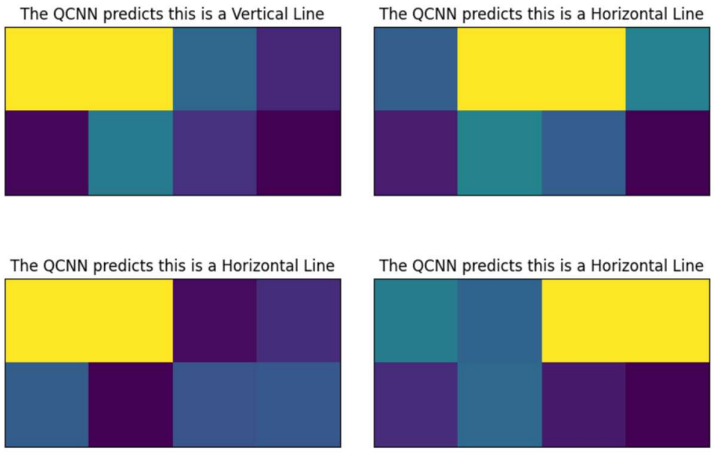
Slika 28: Shema jednog *pooling* sloja

```
feature_map = ZFeatureMap(8)  # data loading and encoding
ansatz = QuantumCircuit(8, name="QCNN Ansatz")

# First Convolutional Layer
ansatz.compose(conv_layer(8, "c1"), list(range(8)), inplace=True)

# First Pooling Layer
ansatz.compose(pool_layer([0, 1, 2, 3], [4, 5, 6, 7], "p1"),
list(range(8)), inplace=True)

# Second Convolutional Layer
ansatz.compose(conv_layer(4, "c2"), list(range(4, 8)), inplace=True)

# Second Pooling Layer
ansatz.compose(pool_layer([0, 1], [2, 3], "p2"), list(range(4, 8)),
inplace=True)

# Third Convolutional Layer
ansatz.compose(conv_layer(2, "c3"), list(range(6, 8)), inplace=True)

# Third Pooling Layer
ansatz.compose(pool_layer([0], [1], "p3"), list(range(6, 8)),
inplace=True)

circuit = QuantumCircuit(8)
circuit.compose(feature_map, range(8), inplace=True)
circuit.compose(ansatz, range(8), inplace=True)

observable = SparsePauliOp.from_list([("Z" + "I" * 7, 1)])

qnn = EstimatorQNN(
    circuit=circuit.decompose(),
    observables=observable,
    input_params=feature_map.parameters,
    weight_params=ansatz.parameters,
)
```

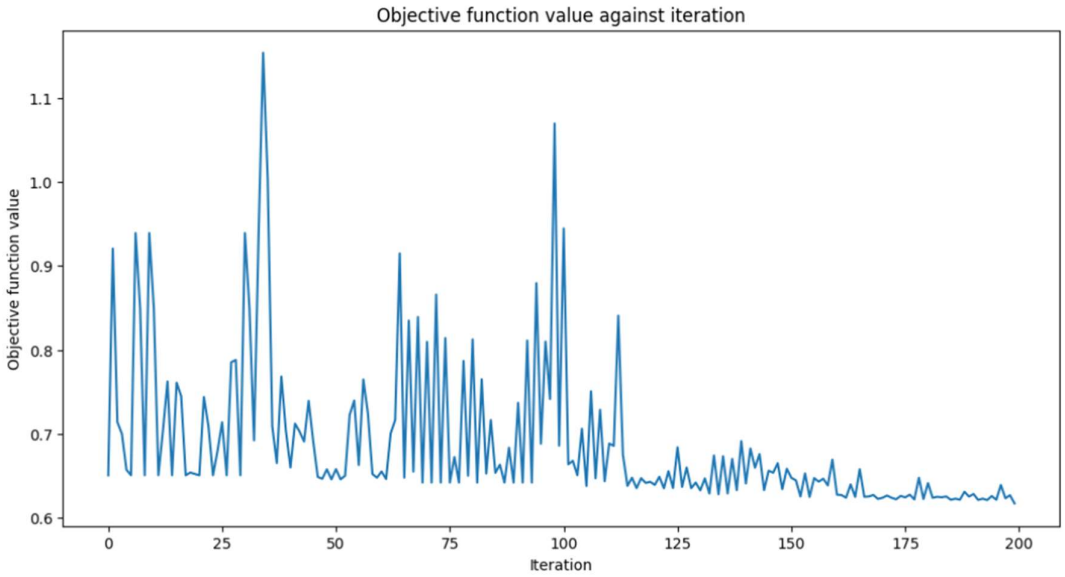
Kôd 11: Izgradnja konkretnog 3-slojnog QCNN modela

```
classifier = NeuralNetworkClassifier(
    qnn,
    optimizer=COBYLA(maxiter=200),
    callback=callback_graph,
    initial_point=initial_point,
    )
x = np.asarray(train_images)
y = np.asarray(train_labels)

objective_func_vals = []
plt.rcParams["figure.figsize"] = (12, 6)
classifier.fit(x, y)

y_predict = classifier.predict(test_images)
x = np.asarray(test_images)
y = np.asarray(test_labels)

for i in range(0, 4):
    ax[i//2, i%2].imshow(test_images[i].reshape(2, 4),aspect="equal")
    if y_predict[i] == -1:
        ax[i//2, i%2].set_title("The QCNN predicts a Horizontal Line")
    if y_predict[i] == +1:
        ax[i//2, i%2].set_title("The QCNN predicts a Vertical Line")
```

Kôd 12: Treniranje i testiranje modela



Slika 29: Testiranje modela – generirani *ad hoc* podatci odnose se na žuti 2×1 pravokutnik – model treba prepoznati radi li se o vertikalnom ili horizontalnom pravokutniku



Slika 30: Prema [13], nad jednostavnim grafičkim *ad-hoc* podatcima koji se nude u Qiskit knjižnici treniranje modela nakon 200 iteracija dolazi do točnosti od oko 80%

U posljednjem poglavlju slijedi pregled jednog nenadziranog QML algoritma koji koristi neuronske mreže za svoju implementaciju.

6. Nenadzirani QML

Unazad nekoliko godina razvijeni su kvantni analogoni velikog broja nenadziranih algoritama, poput kvantne PCA (qPCA), kvantnog varijacijskog autoenkodera (qVAE), kvantnog grupiranja i sl.

Ipak, brojem predloženih konkretnih *near-term* kvantnih krugova i prema najdetaljnije razrađenoj teoriji daleko prednjači kvantni algoritam za generativne suparničke mreže.

U nastavku poglavlja opisat će se klasična i kvantna inačica tog ML modela.

6.1. GAN i q-GAN

Generativne suparničke mreže (engl. Generative adversarial networks, GANs) jedna su od najpopularnijih klasa generativnih modela[10]. Za dani ulazni skup podataka, cilj generativnih modela poput GAN-ova je naučiti kako reproducirati vjerne kopije tih podataka. Zapažena primjena GAN modela je u aplikacijama koje generiraju lažne slike mačaka, pasa, ljudskih lica i sl. [14], [15]

Najčešći općeniti model GAN-a sastoji se od dvije suprotstavljene neuronske mreže, tzv. generatorske mreže i diskriminatorske mreže.

Generatorska mreža (Generator, G) zadužena je za generiranje sintetiziranih, umjetnih podataka na temelju skupa za treniranje. Diskriminatorska mreža (Diskriminator, D) zadužena je za klasifikaciju dobivenog para podataka – jedan primjer iz originalnog skupa za treniranje, jedan primjer koji je stvorila generatorska mreža.

U svrhu boljeg pretraživanja prostora stanja generiranih podataka potrebno je uvesti nedeterminizam pri procesu generiranja novih podataka. Stoga se često koriste tzv. latentni vektorski prostori stanja[11] - distribucije iz kojih se slučajnim uzorkovanjem stvaraju vektori

[10] Generativni modeli glavna su okosnica modernog nenadziranog učenja. Popularnost transformatora (engl. transformer), druge klase generativnih modela, eksplodirala je pojavom tzv. *Large language modela* (LLM), na kojima se temelji alat ChatGPT.
[11] Prostor manje dimenzije od originalnog skupa podataka, ne utječe značajno na kompleksnost modela.

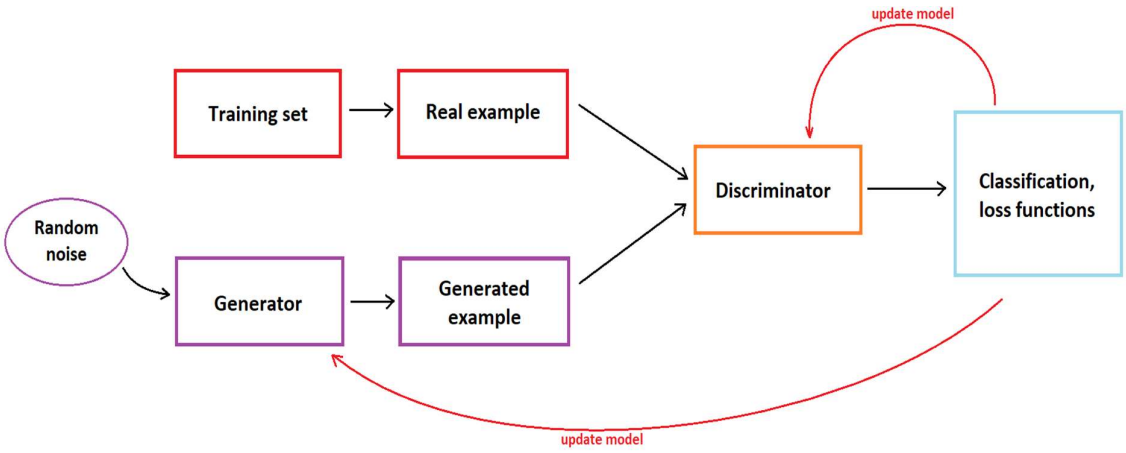
slučajnog šuma. Ti se vektori slučajnog šuma koriste kao ulazi Generatora. Primjeri su Gaussova distribucija (bijeli šum) ili uniformna distribucija. [google]

Generatorov cilj je naučiti kako preslikati vektore latentnog prostora u vektore koji vjerno prate distribuciju vektora iz skupa za treniranje. Drugim riječima, Generator iterativnim promjenama parametara generatorske funkcije pokušava *naučiti distrubuciju* ulaznog skupa podataka.

Diskriminatorska mreža pak ima sasvim suprotan zadatak. Njen je cilj čim uspješnije razlikovati podatke iz skupa za treniranje od podataka koje je stvorio generator. Taj se proces učenja zove i suprotstavljena igra, zato što se može usporediti s poznatom vrstom zadatka strojnog učenja, igranju igara s dva igrača.

Nakon svakog umjetno stvorenog uzorka koji je proizveo Generator, Diskriminatoru se predstavljaju jedan umjetni i jedan uzorak iz skupa za treniranje te on tada donosi odluku koji je od uzoraka umjetan.

Proces treniranja GAN-a gotov je nakon dostizanja trenutaka koji se zove Nashov ekvilibrij. Nakon te točke treniranja GAN-a Generator konzistentno proizvodi uzorke koje *istrenirani* Diskriminator s približno jednakom vjerojatnošću klasificira kao umjetne ili kao stvarne.



Slika 31: Prikaz rada GAN-a / qGAN-a, prema [14], [15]

Na visokoj razini apstrakcije, kvantna generativna suparnička mreža (qGAN) za ostvarenje GAN modela koristi ansatz (koji se implementira kao kvantna neuronska mreža – više riječi o njima u bilo je u Poglavlju 5) u barem jednom od dijelova GAN modela, s ciljem učenja klasičnih ili kvantnih podataka.

U seminalnom paru radova iz 2018. u kojima su predloženi prvi qGAN modeli [16] [17] pokazano je kako se pomoću ansatza mogu ostvariti Generator i Diskriminator i kako se u tom slučaju može računati gradijent pomoću Hesseovih matrica. Također, istraživači su pokazali zašto, barem u teoriji, qGAN za visoko-dimenzionalne podatke nudi ekponencijalno ubrzanje nad klasičnim GAN-ovima.

Tablica 3: Podjela qGAN modela predloženih u istaknutim radovima prema određenim svojstvima – arhitekturi implementacije, koji dio modela je ostvaren kvantno (u kojem se dijelu koristi ansatz) i vrstu podataka koju obrađuje. [18]

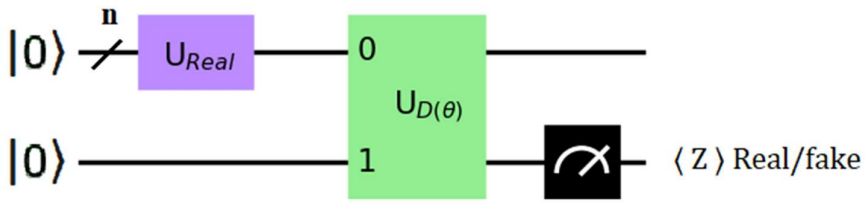
Implementation	Fault-tolerant			Hybrid quantum-classical			
Variable type			Any	Discrete			Continuous
Generator	Q	C	Q	Q	Q	Q	Q, C
Discriminator	Q	Q, C	Q	C	C	Q	Q, C
Data type	Q	Q	C	C	C	Q	C

Ono što je zajedičko svim modelima je način treniranja mreža koji je prisutan i u klasičnom GAN-u. Suprotstavljene kvantne neuronske mreže Generator i Diskriminator iterativno se treniraju i međusobno ovise o trenutnim izlaznim vrijednostima suparnika. Dokazano je da se na isti način kao s klasičnim GAN-om Nashev ekvilibrij postiže onda kada Generator proizvodi distribuciju podataka vrlo sličnu distribuciji skupa za treniranje.

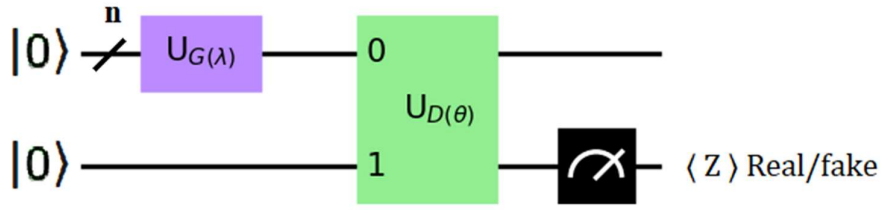
U slučaju qGAN-a, cilj kvantnog Generatora je reproducirati složeno kvantno stanje $|\varphi\rangle$ koje je dovoljno slično kvantnom stanju ulaznog skupa, $|\psi\rangle$.
Kao i u slučaju ostalih varijacijskih kvantnih algoritama, qGAN se iterativno približava optimalnom skupu parametara korištenjem klasičnog optimizatora. Ipak, ono po čemu se

qGAN razlikuje je to da se domena Generatorove funkcije pogreške mijenja usporedno kako napreduje i Diskriminator.

Generator zbog napretka Diskriminatora prima preciznije informacije o prostoru funkcije pogreške te stoga ima bolji uvid o tome kako mijenjati parametre svoje mreže.



Slika 32: Slučaj klasificiranja kvantnih stanja iz skupa za treniranje, prema [3]



Slika 33: Slučaj klasificiranja kvantnih stanja koje je stvorio Generator, prema [3]

Slika 12 i Slika 13 prikazuju prijedlog qGAN modela u dva slučaja klasificiranja dobivenih podataka. Razlika između modela upravo je u izvoru podataka koji se klasificira, odnosno u izvoru stanja predstavljenog Diskriminatoru. Ti „izvori“ u ovom modelu u oba se slučaja ostvaruju određenim unitarnim operatorom koji djeluje nad n qubita, budući da se radi o stanju u n-dimenzionalnom Hilbertovom prostoru. Ovisno o tome što je izmjereno na donjem qubitu nakon Diskriminatora ($|0\rangle$ ili $|1\rangle$) stanje se klasificira kao stanje iz originalnog skupa za treniranje ili kao umjetno stanje.

U prvom slučaju (Slika 12), Generator opisan skupom parametara $\vec{\lambda}$ djeluje nad standardno pripremljenim inicijalnim stanjem qubita $|0^{\otimes n}\rangle$, pa se ukupno djelovanje nad podatcima zapisuje kao

$$|Data_G\rangle = U_{G(\vec{\lambda})}|0\rangle^{\otimes n}$$

(16)

Uzimajući u obzir i donji qubit koji kasnije služi za mjerenje i koji nije spregnut s ostalih n bitova, unitarni zapis djelovanja Diskriminatora je

$$U_{D(\vec{\theta})}(\,|Data_G\rangle \otimes |0\rangle\,).$$

(17)

S obzirom da je riječ o potpuno kvantnom qGAN-u, jasno je da se klasični podatci u slučaju sa Slike 13 moraju kodirati u kvantna stanja. Upravo je to zadaća unitarnog operatora U_{Real}, čiji imlementacijski detalji u ovom trenutku nisu suviše bitni (više u potpoglavlju 4.1.2). Njegova se transformacija nad podatcima može zapisati u obliku

$$|Data_R\rangle = U_R|0\rangle^{\otimes n}\,,$$

(18)

te se takvo stanje prosljeđuje Diskriminatoru koji je u tom koraku treniranja definiran skupom parametara $\vec{\theta}$. Na sličan se način kao i u prethodno opisanom slučaju dolazi do zapisa djelovanja Diskriminatora:

$$U_{D(\vec{\theta})}(\,|Data_R\rangle \otimes |0\rangle\,).$$

(19)

Treba naglasiti da se vektor parametara $\vec{\theta}$ osvježava svakim korakom algoritma nakon djelovanja (klasičnog) optimizatora. Optimizator ima za cilj minimizirati vjerojatnost krivog klasificiranja stanja, što je ekvivalentno minimiziranju očekivanja izmjerene vrijednosti zadnjeg qubita.

Nakon ovako definiranih vrijednosti ansatza i uzevši u obzir da je klasa *stvarnih* podataka ekvivalentna izmjerenom stanju $|0\rangle$, a klasa *generiranih* podataka stanju $|1\rangle$, autori iz [16], [17] na vrlo su intuitivan način (korištenjem poznatog pristupa *minmax* iz teorije igara) opisali izraz koji optimizator pokušava minimizirati:

$$min\ max\ (\ \mathrm{P}[\ D(\vec{\theta}, |Data_R\rangle) = |0\rangle\ ]\ +\ \mathrm{P}[\ D(\vec{\theta}, G(\vec{\lambda})) = |1\rangle\ ]\ )$$

(20)

Drugim riječima, pokušava se minimizirati vjerojatnost da Diskriminator ispravno klasificira umjetno stvorena stanja.

Stoga je i jedna od valjanih funkcija gubitka Generatora izražena kao negirana vjerojatnost ispravnog klasificiranja generiranih podataka:

$$Cost_G = -\ \mathrm{P}[\ D(\vec{\theta}, G(\vec{\lambda})) = |0\rangle\ ]$$

(21)

Funkcija gubitka Diskriminatora može biti ova razlika vjerojatnosti:

$$Cost_D = \mathrm{P}[\ D\ (\vec{\theta}, G(\vec{\lambda})) = |0\rangle\ ]\ -\ \mathrm{P}[\ D(\vec{\theta}, |Data_R\rangle) = |0\rangle\ ]$$

(22)

Minimiziranjem funkcije $Cost_D$ postiže se i cilj Diskriminatora: maksimizirati vjerojatnost ispravnog klasificiranja stvarnih podataka – desna vjerojatnost u (22) – te minimizirati vjerojatnost pogrešnog klasificiranja generiranih podataka – lijeva vjerojatnost u (22).

S obzirom na nedavne velike uspjehe klasičnih GAN-ova, nada je da će qGAN-ovi polučiti slične rezultate te su stoga danas predmet velikog broja istraživanja vezana za primjene ovakve tehnologije.

Zaključak

Jedna od najbrže rastućih područja istraživanja u znanosti su kvantna računala i strojno učenje. Na presjeku tih dviju domena nalazi se mlado područje kvantnog strojnog učenja, čiji je potencijal velik i prepoznat od strane najvećih tehnoloških multinacionalnih tvrtki.

Pretpostavlja se da bi kvantni algoritmi mogli ponuditi rješenja na neka od najtežih simulacijskih i računskih zadataka u medicini, matematici, kemiji, fizici, bankarstvu, telekomunikacijama i dr.

Makar se trenutno još ne osjete prednosti kvantnih modela strojnog učenja nad klasičnim modelima, predviđa se da bi velik korak u tehnološkim mogućnostima mogli pružiti upravo QML algoritmi koji se trenutno razvijaju.

Prema svemu sudeći, era kvantnih računala je na pomolu i stoga ima smisla razvijati kvantnu računarsku teoriju dok se očekuje pojava dovoljno stabilnih i skalabilnih kvantnih strojeva.

Uz relativno jednostavne modifikacije klasičnih ML algoritama i prateći neke jednostavne principe kvantne mehanike, moguće je ostvariti velik broj kvantnih inačica tih ML algoritama.

Literatura

[1] M. Pavičić, *Quantum Computation and Quantum Communication*, New York: Springer, 2006.

[2] S. Kumar, *Fundamental Limits to Moore's Law*, 2015.

[3] IBM, *qiskit.org*, IBM, 2022.

https://learn.qiskit.org/course/ch-algorithms/quantum-circuits. [Pokušaj pristupa 19 12 2022].

[4] I. Cong, S. Choi i M. D. Lukin, *Quantum convolutional neural networks* u

Q2B Conference, San Jose, 2019.

[5] IBM, *qiskit.org, ML and QML* IBM:

https://learn.qiskit.org/course/machine-learning/introduction. [Pokušaj pristupa 2 5 2023].

[6] M. Le Bellac, *A Short Introduction to Quantum Information and Quantum Computation*,

Paris: Cambridge University Press, 2006.

[7] M. Demmer, R. Foncesca i F. Koushanfar, *Richard Feynman: Simulating Physics with Computers*

https://www.fisica.net/computacaoquantica/. [Pokušaj pristupa 1 6 2023].

[8] S. Ilijić, *Glavna skripta kolegija Kvantna računala* Fakultet Elektrotehnike i Računarstva, 2 2023.

http://sail.zpf.fer.hr/labs/kvarac/kvarac.pdf. [Pokušaj pristupa 15 5 2023].

[9] S. Ilijić, *Materijali kolegija Kvantna računala* Fakultet Elektrotehnike i Računarstva, 2 2022.

http://sail.zpf.fer.hr/labs/kvarac/slides/. [Pokušaj pristupa 15 5 2023].

[10] IBM, »IBM Quantum computers,« IBM, 5 2023.

https://www.ibm.com/quantum/ . [Pokušaj pristupa 1 6 2023].

[11] J. Šnajder i M. Čupić, *Nastavni materijali kolegija Uvod u Umjetnu Inteligenciju* Fakultet Elektrotehnike i Računarstva, 5 2023

https://www.fer.unizg.hr/predmet/uuui/materijali. [Pokušaj pristupa 15 5 2023].

[12] M. Schuld i F. Petruccione, *Machine Learning with Quantum Computers*, Toronto, Canada: Springer, 2021.

[13] IBM, *Quantum convolutional neural networks* IBM, 2023.:

https://qiskit.org/ecosystem/machinelearning/tutorials/-11_quantum_convolutional_neural_networks.html. [Pokušaj pristupa 1 6 2023].

[14] Google, *Machine Learning: GAN* Google, 2021.

https://developers.google.com/machine-learning/gan. [Pokušaj pristupa 1 6 2023].

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, W. Warde-Farley, S. Ozair, A. Courville i Y. Bengio, *Generative Adversarial Networks.*

Advances in neural information processing systems, p. 9, 2014.

[16] S. Lloyd i C. Weedbrook, *Quantum Generative Adversarial Learning* ,

Physics Review Letters, br. 121, 2018.

[17] P.-L. Dallaire-Demers i N. Killoran, *Quantum generative adversarial networks,*

Physical Review Letters, p. 10, 2018.

[18] J. Romero i A. Aspuru-Guzik, *Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions.*

Advanced Quantum Technologies, p. 15, 2020.

Sažetak

Primjene metoda strojnog učenja za kvantna računala

Sažetak

Glavna tema ovog završnog rada su metode strojnog učenja primjenjive za kvantna računala. Cilj rada je proučiti koje su metode trenutno ostvarive za izvršavanje na postojećim kvantnim računalima i kako hibridni klasično-kvantni algoritmi pospješuju dosadašnje klasične algoritme.

Za prikaz programskog kôda i simulacije izabran je IBM-ov open-source SDK za programiranje kvantnih računala, Qiskit. [3]

Glavninu rada čini pregled metoda strojnog učenja za koje postoje kvantni pandani [19]. Poseban naglasak stavlja se na programsku potporu za kvantne neuronske mreže, na kvantni GAN i kvantne konvolucijske neuronske mreže. Uz kôd pisan u Qiskitu detaljno se opisuju postupci i teorijska podloga koja omogućava iskorištavanje prednosti kvantnih računala nad klasičnim računalima. Prikaz rezultata simulatora napravljen je dijagramima i tablicama.

Ključne riječi: kvantna računala; strojno učenje; kvantne neuronske mreže; QML; qGAN

Summary

Applications of Machine Learning methods for quantum computers

Summary

The main topic of this bachelor's thesis are machine learning methods applicable for quantum computers. The objective is to study the methods which are applicable on the existing or the near-term quantum computers and explore how the hybrid quantum-classical algorithms (in theory) enhance traditional, classical algorithms.

Implementing the quantum algorithms and running simulations was done using Qiskit, IBM's open-source SDK. [3]

The bulk of the paper consists of an overview of the well-known machine learning methods and their quantum counterparts [19]. In Chapters 3 and 4, special emphasis is placed on quantum neural networks implementation of quantum GAN and quantum convolutional neural networks using variational quantum circuits. Along with the Qiskit code, the description of theoretical foundations behind it is also provided.
The simulated results are presented through diagrams and tables.

Keywords: quantum computers; machine learning; quantum neural networks; QML; qGAN

Skraćenice

ML	*Machine Learning*	strojno učenje
QML	*Quantum Machine learning*	kvantno strojno učenje
QNN	*Quantum Neural Network*	kvantne neuronske mreže
ML	*Machine learning*	strojno učenje
SDK	*Software development kit*	softverski razvojni paket
GAN	*Generative adversarial network*	Generativne suparničke mreže
qGAN	*quantum GAN*	kvantni GAN