MARIANO MOLLO

IMPLEMENTAZIONE DI UN ALGORITMO KNN MULTICLASSE SU HARDWARE QUANTISTICO

[8 settembre 2019 at 15:30 – classicthesis v4.6]

Università degli Studi di Napoli "Federico II"

Scuola Politecnica e delle Scienze di Base

Area Didattica di Scienze Matematiche Fisiche e Naturali

Dipartimento di Fisica "Ettore Pancini"

Laurea Triennale in Fisica

IMPLEMENTAZIONE DI UN ALGORITMO KNN MULTICLASSE SU HARDWARE QUANTISTICO

Relatori:
Giovanni Acampora
Autilia Vitiello

Candidato:
Mariano Mollo
Matr. N85000880

Anno Accademico 2019/2020

[8 settembre 2019 at 15:30 – classicthesis v4.6]

SOMMARIO

Il campo del quantum machine learning è tra gli ambiti di ricerca più promettenti per quanto riguarda il miglioramento dell'analisi dei big data. I problemi predominanti con cui devono confrontarsi i computer classici sono i limiti di memoria e la capacità di elaborazione in tempi appropriati. Sfruttando le proprietà quantomeccaniche della materia, i computer quantistici possono veicolare nuove metodologie d'analisi dati. A tal fine si esplorano le possibilità offerte dall'implementazione di un algoritmo KNN, attraverso il kit di sviluppo software Qiskit e la piattaforma online IBM Q Experience.

ABSTRACT

Quantum Machine Learning is among the most anticipated fields when dealing with improvements in big data analysis. The main issues classical computers have to confront with are limits in memory and enormous computation times. Making use of quantum mechanical properties of matter, quantum computers can spur the development of new methods to improve big data analysis. With that in mind, this thesis explores the capabilities of an implementation of a KNN algorithm, using the Qiskit software development kit and the online platform IBM Q Experience.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [5]

RINGRAZIAMENTI

Regarding the typography and other help, many thanks go to Marco Kuhlmann, Philipp Lehman, Lothar Schlesier, Jim Young, Lorenzo Pantieri and Enrico Gregorio[1], Jörg Sommer, Joachim Köstler, Daniel Gottschlag, Denis Aydin, Paride Legovini, Steffen Prochnow, Nicolas Repp, Hinrich Harms, Roland Winkler, Jörg Weber, Henri Menke, Claus Lahiri, Clemens Niederberger, Stefano Bragaglia, Jörn Hees, Scott Lowe, Dave Howcroft, José M. Alcaide, David Carlisle, Ulrike Fischer, Hugues de Lassus, Csaba Hajdu, Dave Howcroft, and the whole LaTeX-community for support, ideas and some great software.

Regarding LyX: The LyX port was intially done by *Nicholas Mariette* in March 2009 and continued by *Ivo Pletikosić* in 2011. Thank you very much for your work and for the contributions to the original style.

1 Members of GuIT (Gruppo Italiano Utilizzatori di TeX e LaTeX)

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

ELENCO DEGLI ALGORITMI

ACRONIMI

ML machine learning

CQ computer quantistico

QC quantum computer

MLQ machine learning quantistico

QML quantum machine learning

QRAM quantum random access memory

KNN k-nearest neighbours

FF-QRAM flip-flop QRAM

QDB quantum database

Parte I

TEORIA

Introduzione al campo del quantum computing e del quantum machine learning, presentazione della teoria di base e degli strumenti usati.

1 INTRODUZIONE

Chiunque abbia un cellulare di nuova generazione è venuto in contatto con il campo del machine learning, la disciplina che si pone l'obiettivo di rendere i computer capaci di apprendere dall'esperienza. Gli esseri umani sono strettamente legati al processo di apprendimento, a partire dalla prima infanzia ed in maniera più o meno evidente durante tutto il corso vitale. Sebbene tale processo sembri automatico, in realtà è espressione di un complesso sistema di feedback, che in fin dei conti è codificato dal nostro codice genetico. Il machine learning (ML) (tradotto *apprendimento automatico*) prevede la scrittura del codice di programmazione che permetta ai computer di effettuare qualcosa di analogo all'apprendimento. Tali algoritmi sono il motore dei moderni assistenti vocali e dei sistemi che ci suggeriscono nuovi prodotti da acquistare o video da guardare, ma anche nel prevenire le frodi su carta di credito, per filtrare lo spam dalle nostre caselle email, nell'individuare e diagnosticare malattie, e tanto ancora [9].

Un esempio recente di uso del ML è la funzione di gestione adattiva della batteria, lanciata in Android 9, che usa il machine learning per predire quali applicazioni l'utente userà nelle prossime ore e quali no, in modo da dedicare la carica solo per le applicazioni più usate dal proprietario. [1]

Stando all'IBM [3], ogni giorno vengono creati circa $2,5 \times 10^{18}$ byte di dati: questo numero in costante crescita mette chi ha a che fare con i dati di fronte alla necessità di dotarsi di algoritmi avanzati che possando fare ordine in questo oceano di informazioni. Per trovare modelli e correlazioni su insiemi molto grandi è necessario un numero di operazioni che richiede molto tempo ed energia per essere portate a termine; trovare metodi efficienti per completare questi compiti diventa necessario sotto molti punti di vista.

Diversi lavori hanno evidenziato il ruolo nodale rispetto a questo problema che può avere l'uso della computazione quantistica, oggetto di intensa ricerca teorica da vari decenni e recentemente di sperimentazione da parte di studenti [2] e ricercatori [10].

Un computer quantistico (CQ) (o quantum computer (QC)) usa le proprietà peculiari dei sistemi quantomeccanici per manipolare ed elaborare l'informazione in modi inaccessibili ai normali computer classici. Così come un computer classico manipola bit, un computer quantistico fa uso dei cosiddetti bit quantistici (qubit). I bit e i qubit sono entrambi entità binarie, il che significa che possono assumere solo i valori 0 o 1. Ad ogni modo, un bit classico non probabilistico[1] può assumere solo uno di questi valori alla volta, laddove un qubit può trovarsi in una sovrapposizione lineare dei due stati. Il fatto di lavorare con una sovrapposizione di stati di un sistema, invece che con uno

1 Per informazioni sui bit classici probabilitici si faccia riferimento alla sezione 3.2 di [15].

stato definito alla volta, implica che si possono effettuare determinate operazioni che coinvolgono più stati contemporaneamente.

Nonostante ciò, uno degli ostacoli principali di questo campo è rappresentato proprio dalla caratteristica alla base della meccanica quantistica: quando si ha un qubit in una sovrapposizione di stati, non possiamo conoscerne i dettagli intrinseci in un determinato istante, a meno di effettuare una misura; se ci provassimo, otterremmo un risultato ben definito, facendo collassare la funzione d'onda che descrive il qubit, riducendolo in fin dei conti ad un bit con uno stato definito (esattamente quello che misuriamo), e perdendo tutte le informazioni quantistiche contenute nel qubit.

Per avere un'idea di come un CQ si rapporti con un computer ordinario, si possono considerare diversi algoritmi quantistici, che forniscono accelerazioni esponenziali se confrontati con le loro controparti classiche; l'algoritmo di fattorizzazione in numeri primi di Shor è uno tra i più famosi [12].

Alla luce di questi fatti, negli ultimi anni c'è stata particolare attenzione nei riguardi del nuovo campo del machine learning quantistico (MLQ) (o quantum machine learning (QML)), che mette insieme il ML con il quantum computing.

Più nello specifico, si parla di machine learning migliorato quantisticamente quando la computazione quantistica è usata per migliorare algoritmi di ML classico. Ciononostante, entrambi i campi sono benefici l'un l'altro, poiché il ML classico può anche essere usato per migliorare aspetti della computazione quantistica [6]. Questa tesi si occuperà solo del campo del machine learning aumentato quantisticamente.

1.1 MOTIVAZIONE

Il QML è un campo molto giovane e in fermento. Stiamo assistendo a qualcosa di analogo a quella che era la nascita dei computer classici, ovvero lo sviluppo di un numero sempre crescente di algoritmi, tecniche ed hardware di base in un ambiente in cui le cose consolidate sono ben poche: il QML è solo una delle branche che sono state reputate promettenti per quanto riguarda i vantaggi portati dai computer quantistici [8]. In ogni caso c'è necessità di inventare nuove strategie per risolvere anche i problemi più semplici: per avere risultati affidabili spesso si richiede un numero considerevole di qubit ed unità di memorizzazione che possano conservare informazioni quantistiche, come la quantum random access memory (QRAM). Al giorno d'oggi il numero massimo di qubit a superconduzione, da quanto viene riferito, è di 50 [4].

I campi che si prevede otterranno accelerazioni maggiori grazie al quantum computing sono il machine learning, la simulazione di sistemi quantomeccanici, i problemi di ottimizzazione e l'analisi finanziaria.

Negli ultimi anni ci sono state varie implementazioni innovative di algoritmi di MLQ [14]: seguendo i passi che hanno portato alla nascita del ML nel XX secolo, Tacchino et al. [13] hanno implementato

un percettrone su hardware quantistico; Schuld et al. [10] hanno invece implementato un algoritmo k-nearest neighbours (KNN). Questa tesi parte proprio da quest'ultimo articolo, analizzando le possibilità di miglioramento in conseguenza delle novità nell'hardware e nelle tecniche di codifica dei dati.

1.2 DOMANDA DI RICERCA

Prendendo come punto di partenza gli algoritmi proposti nell'ambito di ricerca del QML questa tesi proverà a rispondere alla seguente domanda:

> quali sono le prestazioni di un algoritmo k-nearest neighbours di quantum machine learning implementato su hadware quantistico di piccola e media dimensione attualmente disponibili e come scala all'aumentare delle risorse?

I capitoli seguenti introdurranno le basi teoriche necessarie e gli strumenti usati per rispondere a questa domanda.

2 FONDAMENTI TEORICI

*Si noti che tutti i concetti introdotti nelle sezioni 2.1 e 2.2 possono essere trovati in qualunque libro di testo di base sull'informazione quantistica e quindi non ci saranno riferimenti individuali ad essi. Entrambe queste sezioni sono principalmente basate sul libro di testo di Noson e Mannucci[**noson**].*

2.1 BIT QUANTISTICI

2.1.1 Sistemi a singolo qubit

I computer classici manipolano i bit, mentre l'unità più fondamentale di un computer quantistico è chiamata *quantum bit*, spesso abbreviato come *qubit*. Invece che 0 e 1, gli stati dei qubit sono denotati $|0\rangle$ e $|1\rangle$ per ragioni che verranno evidenziate successivamente.

Un buon esempio di un'implementazione fisica di un qubit è il singolo elettrone di un atomo di idrogeno abbozzato in figura. Solitamente l'elettrone si trova nel suo stato fondamentale, che può essere definito come lo stato $|0\rangle$ del qubit. Usando un impulso laser, l'elettrone può essere eccitato verso il guscio di valenza successivo più energetico, che può essere definito come lo stato $|1\rangle$ del qubit. Dopo del tempo t l'elettrone decadrà per decoerenza verso il suo stato fondamentale $|0\rangle$; questo tempo è chiamato *tempo di decoerenza longitudinale* o *di smorzamento dell'ampiezza* ed è un parametro importante per misurare la vita dei qubit.

Un bit classico non probabilistico può assumere solo uno dei suoi due possibili valori alla volta. Al contrario, i qubit obbediscono alle leggi della meccanica quantistica, il che dà luogo all'importante proprietà che, oltre ad essere in un definito stato $|0\rangle$ o $|1\rangle$, possono anche essere in una sovrapposizione di due stati. Matematicamente ciò è espresso attraverso la combinazione lineare degli stati $|0\rangle$ e $|1\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}, \tag{2.1}$$

dove α e β sono coefficienti complessi e ci si riferisce spesso a loro come ampiezze. Qualsiasi ampiezza η può essere ulteriormente suddivisa in un fattore di fase complesso $e^{i\varphi}$ ed un numero reale non negativo ξ, cosicché

$$\eta = e^{i\varphi}\xi. \tag{2.2}$$

Nella prima equazione, $|0\rangle$ è la rappresentazione nella notazione di Dirac del fatto che il qubit sia nello stato 0 e può essere rappresentato

come un elemento di uno spazio vettoriale complesso bidimensionale H_2, detto spazio di Hilbert. Inizialmente definito da Dirac, l'oggetto

$$|\varphi\rangle\,, \tag{2.3}$$

è chiamato *ket* e il suo coniugato hermitiano

$$|\varphi\rangle^\dagger = \langle\varphi| \tag{2.4}$$

è chiamato *bra*. Il coniugato hermitiano, denotato con una daga (†), di p.e. un vettore colonna c bidimensionale a valori complessi c_1 e c_2,

$$c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \tag{2.5}$$

è ottenuto prendendo il complesso coniugato di ciascun valore e trasponendo il vettore risultante:

$$c^\dagger = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}^\dagger = \begin{pmatrix} c_1^* & c_2^* \end{pmatrix}, \tag{2.6}$$

dove i complessi coniugati sono indicati con un asterisco in apice (∗). Il coniugato hermitiano è definito sia per vettori che per matrici quadrate. Il prodotto interno tra un bra e un ket è chiamato *bra-ket* ed è scritto

$$\langle\varphi|\varphi\rangle\,. \tag{2.7}$$

Si noti che tutte le sezioni seguenti faranno uso intensivo della notazione bra-ket. Gli stati quantistici $|0\rangle$ e $|1\rangle$ sono chiamati la base computazionale e costituiscono una base ortonormale di H_2. Quando un qubit è espresso in termini dei due stati $|0\rangle$ e $|1\rangle$, si dice che sia nella sua *base normale*. Per motivi di chiarezza, $|0\rangle$ e $|1\rangle$ possono essere rappresentati con i vettori bidimensionali

$$|0\rangle \cong \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ e } |1\rangle \cong \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.8}$$

Si noti che un ket e la sua rappresentazione vettoriale non sono lo stesso oggetto, dato che, per essere ben definito, un vettore ha bisogno della specificazione di una base, laddove un ket non lo richiede. Questa tesi farà uso del simbolo $\cong$ quando si cambia tra i due diversi modi di rappresentare un ket. Sostituendo i vettori dell'Eq. 2.8 nell'Eq. 2.1 si ottiene la rappresentazione vettoriale di $|\psi\rangle$

$$\psi \cong \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{2.9}$$

Sebbene un qubit possa essere in una sovrapposizione di stati $|0\rangle$ e $|1\rangle$, quando misurato assumerà un valore preciso tra $|0\rangle$, con probabilità

$$P(|0\rangle) = |\alpha|^2, \tag{2.10}$$

e $|1\rangle$, con probabilità

$$P(|1\rangle) = |\beta|^2. \tag{2.11}$$

Il fatto che la probabilità di misurare un particolare stato è uguale al quadrato del modulo della rispettiva ampiezza fu postulato per la prima volta da Born e, dunque, è chiamata regola di Born. Visto che la probabilità totale di misurare un qualunque valore deve essere unitaria, deve soddisfarsi la condizione di normalizzazione

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.12}$$

Se prendiamo uno stato in cui si ha, per esempio, $\alpha = \frac{1}{\sqrt{2}}$ e $\beta = \frac{i}{\sqrt{2}}$, abbiamo che il qubit è in una sovrapposizione quantistica, cosa impossibile da ottenere con un computer classico. Legato a questa caratteristica c'è un altro importante parametro della vita di un qubit: il *tempo di coerenza trasversale* o *di smorzamento della fase*. Esso è misurato preparando la sovrapposizione uniforme $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$: per via dell'inevitabile interazione con l'ambiente, dopo del tempo t il comportamento quantistico sarà perso e lo stato sarà ben determinato tra $|0\rangle$ e $|1\rangle$. Il processo di perdita del comportamento quantistico è chiamato decoerenza. Nei casi in cui $\alpha = 1$ o $\beta = 1$ il qubit non è propriamente in una sovrapposizione ma in uno stato definito, $|0\rangle$ o $|1\rangle$ rispettivamente.

Perciò, un qubit è intrinsecamente probabilistico ma quando viene misurato collassa in un singolo bit classico (0 o 1). Ne segue che una misura distrugge l'informazione sulla sovrapposizione del qubit (ovvero i valori di α e β). Questo costituisce una delle principali difficoltà nel progettare algoritmi quantistici, dato che solo una limitata quantità di informazioni può essere ottenuta riguardo gli stati finali dei qubit nel computer quantistico.

Usando coordinate sferiche polari, un singolo qubit può essere visualizzato sulla cosiddetta sfera di Bloch, parametrizzando α e β dell'Eq. 2.1 come segue:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \tag{2.13}$$

La sfera di Bloch ha raggio 1 ed è, dunque, una sfera unitaria. Lo stato $|0\rangle$ del qubit è definito in modo che si trovi lungo il semiasse z positivo e lo stato $|1\rangle$ è definito in modo che si trovi lungo il semiasse z negativo, come indicato in Figura. A questo punto è importante notare che questi due stati sono mutuamente ortogonali in H_2, sebbene non lo siano sulla sfera di Bloch.

Gli stati del qubit sull'equatore della sfera, come gli assi coordinati x e y, rappresentano sovrapposizioni uniformi dove $|0\rangle$ e $|1\rangle$ hanno entrambi probabilità di misura pari a 0,5. L'asse x, ad esempio, rappresenta la sovrapposizione uniforme $|q\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Come illustrato in Figura, qualunque stato bidimensionale arbitrario $|\psi\rangle$ può essere decomposto nei suoi angoli polari θ e φ e visualizzato come un vettore sulla sfera di Bloch (dopo essere stato normalizzato, se necessario). Tale oggetto è chiamato vettore di Bloch dello stato $|\psi\rangle$ del qubit. La sfera di Bloch sarà lo strumento principale di visualizzazione per le manipolazioni di qubit in questa tesi.

Similmente alle porte logiche in un computer classico, un CQ manipola qubit per mezzo di porte logiche quantistiche, le quali saranno introdotte in dettaglio nella sezione 2.2. In genere, una porta logica quantistica arbitraria U che agisca su uno stato di singolo qubit è una trasformazione unitaria che può essere rappresentata con una matrice 2x2

$$U \cong \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \tag{2.14}$$

la cui azione su $|\psi\rangle$ è definita come

$$U|\psi\rangle \cong \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{pmatrix}. \tag{2.15}$$

La matrice U deve essere unitaria, il che comporta che il suo determinante deve essere unitario:

$$|\det(U)| = 1, \tag{2.16}$$

e il suo coniugato hermitiano $U^\dagger$ deve essere uguale alla sua inversa:

$$UU^\dagger = U^\dagger U = \mathbb{1} = UU^{-1} = U^{-1}U. \tag{2.17}$$

Tutte le porte logiche quantistiche devono essere unitarie poiché in questo modo si conserva la condizione di normalizzazione dello stato del qubit su cui agiscono. L'insieme di tutte le matrici complesse unitarie bidimensionali con determinante uguale ad uno è chiamato gruppo unitario speciale ($SU(2)$) e tutte le porte logiche quantistiche a singolo qubit sono dunque elementi di $SU(2)$. Inoltre, un insieme di porte G costituito da m porte quantistiche $g_1, g_2, \ldots, g_m$ è chiamato *insieme di porte quantistiche universale* allorquando è un sottoinsieme denso di $SU(2)$ come definito nel riquadro seguente.

Definizione: sottoinsieme denso di $SU(2)$

L'insieme di porte G è un sottoinsieme denso di $SU(2)$ quanto, data una qualunque porta quantistica $W \in SU(2)$ e una precisione arbitraria $\varepsilon > 0$, esiste un prodotto J di porte appartenenti a G che è una ε-approssimazione di W (Dawson, Nielsen, 2005).

2.1.2 Sistemi a qubit multipli

Un computer classico coun un bit di memoria non è particolarmente utile e, allo stesso modo, un CQ con un qubit è piuttosto inutile. Per essere in grado di effettuare computazioni grandi e complicate è necessario combinare diversi qubit singoli per creare un grande CQ. Quando ci si muove da un sistema a singolo qubit ad uno a molti qubit c'è bisogno di un nuovo strumento matematico, il cosiddetto prodotto tensoriale (simbolo $\otimes$). Il prodotto tensore di due qubit si scrive

$$|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle\,|\psi_2\rangle = |\psi_1\psi_2\rangle\,, \tag{2.18}$$

dove nelle ultime due espressioni si è omesso il simbolo $\otimes$; queste due sono una stenografia per il prodotto tensore tra due qubit.

Un *registro quantistico* di dimensione j è una maniera alternativa di riferirsi al prodotto tensore di j qubit. Per esempio, lo stato nell'Eq. 2.18 è un registro quantistico composto da due qubit. Negli algoritmi di MLQ, grandi stati quantistici sono solitamente suddivisi in vari registri quantistici che soddisfano vari obiettivi, p.e. memorizzare etichette di dati o di classi. Si consideri, ad esempio, uno stato quantistico $|\Phi\rangle$ che sia suddiviso tra due diversi registri quantistici, un registro dati (d) con n qubit e un registro classe (c) con due qubit. Lo stato $|\Phi\rangle$ si scrive allora

$$|\Phi\rangle = |d;c\rangle = |d_1 \ldots d_n; c_1, c_2\rangle\,, \tag{2.19}$$

dove i punti e virgola sono usati per separare i registri quantistici.

Nella rappresentazione vettoriale, il prodotto tensore di due ket $|0\rangle$ è definito come

$$|00\rangle = |0\rangle \otimes |0\rangle \cong \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1 * \begin{pmatrix}1\\0\end{pmatrix}\\ 0 * \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}. \tag{2.20}$$

L'ultima espressione nell'Eq. 2.20 mostra che lo stato a due qubit $|00\rangle$ non è più bidimensionale ma quadridimensionale. Dunque, vive in uno spazio di Hilbert quadridimensionale $\mathcal{H}_4$. Una porta quantistica che agisca su molti qubit può allora non avere le stesse dimensioni di una porta a singolo qubit (Eq. 2.14), il che richiede un nuovo formalismo per le porte per sistemi a qubit multipli.

Si provi ad applicare un'arbitraria porta a singolo qubit U (Eq. 2.14) al primo qubit nello stato a due qubit $|00\rangle$. Il secondo qubit dello stato $|00\rangle$ dovrebbe rimanere immutato, che, in altre parole, significa applicare la matrice identità $\mathbb{1}$ 2x2 su di esso. Per effettuare queste

operazioni, si definisce il prodotto tensore delle due porte a singolo qubit come

$$U \otimes \mathbb{1} \cong \begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & b * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ c * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & d * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ c & 0 & d & 0 \\ 0 & c & 0 & d \end{pmatrix}. \tag{2.21}$$

Dunque, il risultato del prodotto tensore $U \otimes \mathbb{1}$ può essere rappresentato come una matrice unitaria 4x4 che può ora essere usata per trasformare il vettore 4x1 che rappresenta lo stato $|00\rangle$ nell'Eq. 2.20:

$$U \otimes \mathbb{1} |00\rangle \cong \begin{pmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ c & 0 & d & 0 \\ 0 & c & 0 & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ c \\ 0 \end{pmatrix}. \tag{2.22}$$

Si può anche effettuare prima le operazioni di singolo qubit sui rispettivi qubit, seguite dal prodotto tensore dei due risultanti vettori:

$$(U \otimes \mathbb{1})(|0\rangle \otimes |0\rangle) = U|0\rangle \otimes \mathbb{1}|0\rangle \cong \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ c \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ c \\ 0 \end{pmatrix}. \tag{2.23}$$

Questo formalismo può essere esteso a qualunque numero di qubit, e l'uso del prodotto tensore porta ad un aumento esponenziale della dimensionalità dello spazio di Hilbert. Così, n qubit vivono in uno spazio di Hilbert 2^n-dimensionale ($\mathcal{H}_2^{\otimes n}$) e possono immagazzinare il contenuto di 2^n bit classici. Per fare un esempio, solo 33 qubit possono contenere l'equivalente di $2^3 3 = 8589934592$ bit (= 1 gigabyte), il che porta chiaramente con sè il potenziale per le enormi accelerazioni nella potenza di calcolo, come verrà mostrato più avanti.

Quando si considerano sistemi a molti qubit, si incontreranno stati quantistici che possono o meno essere fattorizzati. Per esempio, si provi a riscrivere l'ultima espressione dell'Eq. 2.23 come

$$\begin{pmatrix} a \\ 0 \\ c \\ 0 \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + 0 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + c \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + 0 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cong a|00\rangle + c|10\rangle, \tag{2.24}$$

che può essere fattorizzato nel prodotto tensore

$$a|00\rangle + c|10\rangle = (a|0\rangle + c|1\rangle) \otimes |0\rangle . \tag{2.25}$$

Al contrario, si consideri uno dei quattro famosi stati di Bell:

$$|\psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}. \tag{2.26}$$

È semplice verificare che lo stato a due qubit $|\psi^+\rangle$ non può essere fattorizzato in un prodotto tensore di due stati di singolo qubit. Ora si immagini che a due persone, Allegra e Bernardo, vengano dati due elettroni preparati nello stato quantistico $|\psi^+\rangle$. Allegra tiene il primo elettrone nel laboratorio e Bernardo porta il secondo elettrone a casa sua. Dopo del tempo t, Allegra diventa curiosa di misurare se il suo elettrone è nello stato $|0\rangle$ o $|1\rangle$ ed effettua una misura lungo la base normale. Applicando la regola di Born allo stato quantistico $|\psi^+\rangle$, Allegra sa che misurerà il suo elettrone in uno tra gli stati $|0\rangle$ e $|1\rangle$ con uguali probabilità 0,5. Si noti che, sebbene il vettore di stato $|\psi^+\rangle$ sia conosciuto con esattezza, il risultato della misura è ancora incerto. Misurando il suo elettrone lo trova essere nel suo stato $|1\rangle$. A partire dall'Eq. 2.26, sapendo che la misura fa collassare una sovrapposizione, lo stato post misura (PM) $|\psi^+\rangle_{PM}$ è

$$|\psi^+\rangle_{PM} = |1_A 0_B\rangle , \tag{2.27}$$

dove i pedici indicano quale elettrone appartiene ad Allegra (A) e quale a Bernardo (B). Osservando questa espressione, Allegra sa che l'elettrone di Bernardo deve trovarsi nello stato $|0\rangle$ senza aver misurato il secondo elettrone! A casa sua, Bernardo misura il suo elettrone un secondo dopo che Allegra ha effettuato la sua misura e trova che infatti è nello stato $|0\rangle$. Si noti che il secondo elettrone non era per niente vicino ad Allegra ma è stata lo stesso in grado di determinare lo stato del secondo elettrone solo misurando il suo. Dopo aver ripetuto questo esperimento un migliaio di volte, Allegra e Bernardo trovano perfette correlazioni nei loro risultati: ogni volta che Allegra misurava il suo elettrone nello stato $|0\rangle$, Bernardo trovava il suo nello stato $|1\rangle$ e viceversa.

Le correlazioni non locali tra i risultati di misura di qubit sono una peculiare proprietà quantistica degli stati quantistici non fattorizzabili ed è chiamata *entanglement quantistico* (entanglement significa groviglio, garbuglio in inglese). È una parte integrante della calcolo quantistico e la sezione 2.2.2 darà un esempio concreto di come creare uno stato in entanglement in un CQ.

2.2 PORTE LOGICHE QUANTISTICHE

Fino a questo punto, la maggior parte dei concetti introdotti viene dal campo della teoria quantistica pura. Ad ogni modo, questa sezione

segna la fondamentale transizione dal campo della teoria quantistica a quello dell'*elaborazione dell'informazione quantistica* (quantum information processing). Un computer classico elabora l'informazione ed effettua calcoli manipolando sistematicamente bit attraverso l'applicazione di porte logiche come le porte NOT o XOR. Analogamente, un computer quantistico elabora l'informazione ed effettua *calcoli quantistici* manipolando qubit usando *porte logiche quantistiche*, spesso chiamate semplicemente *porte quantistiche*. Solitamente, una sequenza di tali porte quantistiche è richiesta per effettuare un certo compito o per risolvere un particolare problema su un computer quantistico. Tale sequenza di porte quantistiche è chiamata *algoritmo quantistico*.

Ci sono molti modi diversi di realizzare un CQ, per esempio usando ioni intrappolati, fotoni o giunzioni Josephson superconduttive. A seconda del substrato scelto, possono essere implementati diversi insiemi di porte logiche quantistiche e per eseguire un algoritmo quantistico questo deve essere mappato sull'hardware quantistico disponibile. Dunque, gli algoritmi quantistici hanno bisogno di essere tradotti (compilati) in una serie di porte consistenti solo di porte quantistiche dall'insieme universale di porte disponibile. Ci si riferisce a questo procedimento con *compilazione quantistica*. Le sottosezioni seguenti introdurranno alcune porte logiche quantistiche principali a singolo qubit e multipli qubit che saranno usate estensivamente nelle sezioni successive di questa tesi.

2.2.1 Porte a singolo qubit

Le porte logiche quantistiche che agiscono su un sigolo qubit possono essere rappresentate come matrici unitarie 2x2 (vedi Eq. 2.14) le cui azioni su un qubit possono essere visualizzate come rotazioni della sfera di Bloch. Come una porta quantistica a singolo qubit agisca su un qubit, le sue proprietà e rappresentazione matriciale sranno illustrate usando l'esempio dell'equivalente quantistico della porta logica NOT classica: la cosiddetta porta X. La porta X può essere rappresentata dalla matrice unitaria 2x2

$$X \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.28}$$

L'azione della porta X sullo stato di qubit arbitrario $|\psi\rangle$ (Eq. 2.9) può essere analizzata usando la matrice della porta e la rappresentazione vettoriale del qubit. Applicando della semplice algebra lineare si ottiene

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \cong \beta|0\rangle + \alpha|1\rangle. \tag{2.29}$$

Così, applicando la porta X allo stato di qubit $|\psi\rangle$ si scambiano le ampiezze degli stati $|0\rangle$ e $|1\rangle$. Più nello specifico, l'applicazione di X allo stato $|0\rangle$ risulta nello stato $|1\rangle$:

$$X|0\rangle \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cong |1\rangle . \tag{2.30}$$

Nei termini dell'esempio con l'elettrone di valenza di un atomo di idrogeno mostrato in Fig. 2.1, una porta X può essere implementata eccitando l'elettrone dallo stato fondamentale $|0\rangle$ verso il guscio di valenza successivo più energetico dell'elettrone, definito come stato $|1\rangle$, usando un impulso laser controllato.

Lo stato $|0\rangle$ è recuperato quando si applica di nuovo X allo stato $|1\rangle$:

$$X|1\rangle \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cong |0\rangle . \tag{2.31}$$

Sulla sfera di Bloch, la porta X corrisponde ad una rotazione antioraria di π attorno all'asse x, come mostrato in Fig. 2.3.

Dalle Eq. 2.29, 2.30 e 2.31 segue che X è la propria inversa così come il proprio coniugato hermitiano:

$$XX = XX^{\dagger} = \mathbb{1}, \tag{2.32}$$

$$X = X^{-1} = X^{\dagger}. \tag{2.33}$$

Basandosi sul libro di testo di Nielsen e Chuang (2010), l'azione, la rappresentazione del circuito e della matrice e la visualizzazione sulla sfera di Bloch per alcune delle più importanti porte logiche quantistiche a singolo qubit sono riassunte nella tabella 2.1.

2.2.2 Porte a qubit multipli

Le porte logiche quantistiche a qubit multipli agiscono su almeno due qubit allo stesso tempo. Similmente alle porte a singolo qubit, una porta quantistica a n qubit può essere rappresentata come una matrice unitaria $2^n \times 2^n$. Dato che sono coinvolti molteplici qubit, la sfera di Bloch non può più essere usata per visualizzarne l'azione. La porta quantistica NOT controllata a due qubit sarà usata per dimostrare le proprietà, la rappresentazione matriciale e l'azione di una porta quantistica a due qubit. Questo può poi essere facilmente generalizzato a porte quantistiche a n qubit.

La porta NOT controllata o CNOT è data dalla seguente matrice 4×4:

$$\text{CNOT} \cong \begin{pmatrix} \mathbb{1} & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{2.34}$$

La porta CNOT accetta due qubit, uno di controllo e uno bersaglio, come input. Se e solo se il qubit di controllo è nello stato $|1\rangle$, la porta quantistica NOT (X) è applicata al qubit bersaglio. Nelle equazioni, la CNOT sarà sempre seguita da parentesi contenenti il qubit di controllo (c) seguito dal qubit bersaglio (b): CNOT(c,b). La relazione di input-output, chiamata tabella di verità, per la porta CNOT è data nella tabella 2.2.

Per dimostrare l'utilità della porta CNOT si immagini di iniziare con due qubit non in entanglement, entrambi nello stato $|0\rangle$:

$$|\varphi_0\rangle = |0\rangle \otimes |0\rangle = |00\rangle \,. \tag{2.35}$$

Applicando la porta H sul primo qubit si ottiene il seguente stato (ancora non in entanglement):

$$|\varphi_1\rangle = (H \otimes \mathbb{1})\,|\varphi_0\rangle = (H \otimes \mathbb{1})\,|00\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle \,. \tag{2.36}$$

Ora si immagini di applicare la porta CNOT allo stato $|\varphi_1\rangle$, in cui il qubit di controllo è quello di sinistra e il qubit bersaglio è quello di destra:

$$\text{CNOT}(0,1)\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\right) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}(\mathbb{1} \otimes X)\,|10\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle \,. \tag{2.37}$$

L'ultima espressione nell'Eq. 2.37 è uno dei famosi stati di Bell, i quali sono un insieme di quattro stati quantistici in massimo entanglement. Un altro stato di Bell è stato usato nell'esempio di entanglement nella sezione 2.1.2. Dunque, questo esempio mostra come la porta CNOT è cruciale per la generazione di stati in entanglement, poiché applica la porta X ad un qubit bersaglio a seconda dello stato di un secondo qubit di controllo.

Le tre porte quantistiche a qubit multipli più importanti CNOT, Toffoli e nCNOT sono caratterizzate nella tabella 2.3.

2.3 MACHINE LEARNING CLASSICO

Il machine learning (ML), una sottodisciplina dell'intelligenza artificiale, prova a dare la capacità ai computer di imparare dai dati senza che un umano programmi esplicitamente le sue azioni. Può essere suddiviso nei tre maggiori campi di supervisionato, non supervisionato e apprendimento per rinforzo (Schuld, Sinaskiy, Petruccione, 2015). Solo il machine learning supervisionato sarà introdotto poiché questa tesi di ricerca si focalizza esclusivamente su algoritmi di machine learning supervisionato.

L'idea principale del machine learning supervisionato è di addestrare un algoritmo su un insieme di dati *etichettati* contenente, per

esempio, immagini di frutti con i corrispondenti nomi cosicché possa essere usato per etichettare nuove immagini che non fanno parte dell'insieme dati di addestramento. Dato che i campioni nell'insieme dati di addestramento sono etichettati, il processo è detto essere supervisionato.

Più formalmente, nel machine learning supervisionato si è di fronte a paia di variabili di input (x) e di output (o) e si suppone che l'algoritmo di machine learning impari la funzione f che mappa gli input sui relativi output:

$$f(x) = o. \tag{2.38}$$

Così, l'algoritmo dovrebbe approssimare la funzione di mappatura f a tal punto che può predire l'output o per nuovi dati di input sconosciuti $\tilde{x}$ (C. M. Bishop, 2006). La seguente sezione introdurrà un algoritmo ben noto nel campo del ML supervisionato: l'algoritmo k-nearest neighbours classico.

2.3.1 L'algoritmo k-nearest neighbours

Per comprendere la versione quantistica dell'algoritmo di k-nearest neighbours (KNN) pesato sulla distanza proposto da Schuld (2014) che verrà usato successivamente è richiesta una conoscenza pregressa della versione classica dell'algoritmo descritta da Cunningham e Delany (2007) che verrà introdotta in questa sottosezione.

Si immagini di lavorare per una compagnia che gestisce un motore di ricerca e e che venga assegnato il compito di classificare immagini ignote di frutti come mele o banane. Per addestrare l'algoritmo di classificazione, vengono fornite cinque differenti immagini di mele e cinque differenti immagini di banane. Questo verrà chiamato l'*insieme dati di apprendimento* (training data set) D_T. Le immagini in D_T potrebbero essere prese da angolazioni diverse, in condizioni di luce varie ed includere mele e banane di colori diversi.

La maggior parte delle volte, usare la rappresentazione completa dei pixel di ogni immagine per la classificazione non porta a risultati ottimali. Dunque, il passo successivo è di selezionare un certo numero di caratteristiche (features) estratte dalle immagini nell'insime di apprendimento che possono essere usate per differenziare le mele dalle banane. Tali caratteristiche possono essere il codice del colore più frequente tra i pixel, dato che mele e banane hanno spettri di colore diversi. Usare una misura della curvatura dell'oggetto principale nell'immagine è un'altra possibilità, dato che una mela è quasi sferica mentre una banana assomiglia più ad una linea curva.

Selezionando ed estraendo caratteristiche, la dimensionalità dell'insieme dati di apprendimento è drasticamente ridotta da qualche migliaio di pixel ad una manciata di caratteristiche. Le m caratteristiche estratte dalla j-esima immagine sono conservate nel

vettorecaratteristica m-dimensionale $\vec{v}_j$. Matematicamente, l'insieme dati di apprendimento D_T consiste di dieci vettori caratteristica $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_9$, ognuno assegnato alla classe A (mela) o alla classe B (banana). I vettori di apprendimento sono visualizzati come cerchi chiari e scuri in figura.

Data una nuova immagine di una mela o una banana, prima si estraggono le stesse m caratteristiche da essa e si conservano nel vettore d'input $\vec{x}$. Dei tanti algoritmi, si sceglie di usare l'algoritmo KNN, dato che è un classificatore non parametrico, cioè non fa assunzioni pregresse sulla classe della nuova immagine. Dato un nuovo vettore d'input $\vec{x}$ non classificato (stella nella figura), l'algoritmo considera i k elementi più vicini tra quelli dell'insieme dati di apprendimento (usando una misura di distanza predefinita) e classifica $\vec{x}$, basandosi su un voto a maggioranza, come A o B. Così, k è un numero intero positivo, solitamente scelto piccolo ed il suo valore determina il risultato della classificazione. Praticamente, nel caso $k = 3$ in figura, il vettore $\vec{x}$ sarà classificato come appartenente alla classe B (scuro) ma nel caso $k = 6$ sarà etichettato nella classe A (chiaro).

Nel caso $k = 10$, $\vec{x}$ sarà semplicemente assegnato alla classe con più membri. In questo caso, ai vettori di addestramento dovrebbe assegnarsi dei pesi dipendenti dalla distanza (come $\frac{1}{\text{distanza}}$) per aumentare l'influenza dei vettori più vicini al vettore d'input rispetto a quelli più lontani.

2.3.2 Complessità algoritmica: notazione O-grande

Nei campi dell'informatica e della quantum information, la cosiddetta *notazione O-grande*, inizialmente descritta da Bachmann (1894), è spesso usata per descrivere come l'esecuzione di un algoritmo dipende da variabili come l'accuratezza desiderata, il numero di vettori d'input o la loro dimensione. Per questo, essa è un modo di quantificare la *complessità in termini di tempo dell'algoritmo* di un algoritmo quantistico. Nelle sezioni successive di questa tesi, la notazione O-grande sarà usata come uno strumento per quantificare possibili accelerazioni quantistiche negli algoritmi KNN di machine learning quantistico.

3 STRUMENTI

L'intera ricerca per questa tesi è stata effettuata con carta e penna ed un computer con il sistema operativo GNU/Linux Ubuntu. Il linguaggio di programmazione Python, con la sua sintassi molto intuitiva e pacchetti estesi per il calcolo scientifico e il disegno di grafici, è stato usato per i calcoli e per la maggior parte dei grafici in questa tesi. Per l'implementazione di un algoritmo KNN, ci sono principalmente due strade: eseguirlo simulando un CQ o eseguirlo proprio su un CQ reale. Gli strumenti necessari per entrambi gli approcci saranno descritti nelle sezioni seguenti.

3.1 QISKIT

Qiskit è un'interfaccia di programmazione che permette di scrivere circuiti quantistici e simularne l'esecuzione sul proprio computer o inviare un ordine di esecuzione a un vero computer quantistico tramite l'interfaccia offerta dall'IBM Quantum Experience.

3.2 IBM QX

L'IBM Quantum Experience è un servizio offerto gratuitamente che permette a chiunque di avere a che fare con un computer quantistico. Sono presenti risorse didattiche per imparare a scrivere il primo circuito, strumenti di comunità come una piattaforma di domande e risposte e soprattutto un sistema per creare i propri algoritmi.

Parte II

PRATICA

Implementazione dell'algoritmo k-nearest neighbour e della tecnica di costruzione della QRAM; simulazione ed esecuzione su hardware quantistico dei circuiti progettati; analisi dei risultati ottenuti.

4 IMPLEMENTAZIONE

4.1 PREPARAZIONE DI UNO STATO QUANTISTICO

Per analizzare dei dati classici attraverso un computer quantistico abbiamo bisogno di codificare in qualche modo le informazioni contenute nei nostri insiemi. Nel caso specifico, si parla delle coordinate dei vettori nello spazio delle caratteristiche e la classe associata ad ognuno di essi. Per fare questo, costruiamo degli stati quantistici ad hoc che rappresentino i vettori dati in maniera coerente. La procedura usata in questa tesi segue la tecnica di costruzione flip-flop QRAM (FF-QRAM) proposta da Park, Petruccione e Rhee [7].

4.1.1 Flip-flop QRAM

La FF-QRAM è usata per generare un quantum database (QDB) inizializzato in maniera arbitraria. Nell'illustrare l'algoritmo di cotruzione verranno usati due registri quantistici: il primo, denotato genericamente $|j\rangle$, o con il pedice B, indica quale bus di memoria viene usato per il passaggio in corso, mentre il secondo, denotato $|b_l\rangle_R$, con il pedice R, sarà il registro che contiene i valori codificati del vettore dati. I vettori $|j\rangle_B$ vengono anche detti appartenere alla base computazionale. Lo stato finale dei qubit può essere arbitrario e l'ampiezza di probabilità ψ_j con cui è accessibile ciascuno stato $|j\rangle_B$ della base computazionale codifica i valori classici di partenza.

L'operazione QRAM sui qubit bus e registro sovrappone un insieme di dati classici D come

$$\mathrm{QRAM}(D) \sum_j \psi_j |j\rangle_B |0\rangle_R \equiv \sum_l \psi_l |\vec{d}^{(l)}\rangle_B |b_l\rangle_R , \tag{4.1}$$

La FF-QRAM è implementata sistematicamente con elementi comuni dei circuiti quantistici, che includono la porta di Pauli X controllata classicamente, $\bar{c}X$, e la porta di rotazione controllata da n qubit, $C^n R_p(\theta)$. La $\bar{c}X$ inverte il qubit bersaglio solo quando il bit classico di controllo è zero. La porta $C^n R_p(\theta)$ ruota il qubit bersaglio di θ attorno all'asse p della sfera di Bloch solo se tutti gli n qubit sono 1.

L'idea soggiacente al modello della FF-QRAM è rappresentata in figura 4.1, che descrive la procedura per sovrapporre due stringhe di bit indipendenti $\vec{d}^{(l)}$ e $\vec{d}^{(l+1)}$ con ampiezze di probabilità desiderate

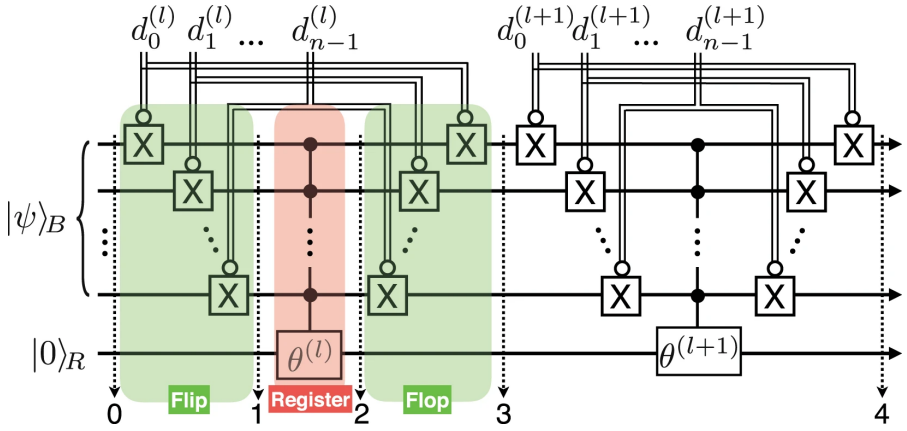


Figura 4.1: Procedimento di costruzione FF-QRAM
Circuito quantistico per FF-QRAM che scrive le stringhe di bit $\vec{d}^{(l)}$ e $\vec{d}^{(l+1)}$ come una sovrapposizione di stato quantistico con ampiezze di probabilità determinate da $\theta^{(l)}$ e $\theta^{(l+1)}$ rispettivamente, usando porte di rotazione controllate da più qubit. Le linee doppie indicano operazioni controllate classicamente, ed il cerchio vuoto (pieno) indica che la porta è attivata quando il bit (qubit) di controllo è 0 (1). Le frecce tratteggiate e numerate indicano i vari passaggi descritti nel testo principale.

nello stato del qubit bus $|\psi\rangle_B$. Lo stato iniziale può essere espresso focalizzandosi su $\vec{d}^{(l)}$ come

$$|\psi_0\rangle_l = \psi_{\vec{d}^{(l)}} |\vec{d}^{(l)}\rangle |0\rangle_R + \sum_{j \neq \vec{d}^{(l)}} \psi_j |j\rangle |0\rangle_R , \tag{4.2}$$

dove $|\psi_s\rangle_l$ denota lo stato degli (n) qubit nel processo di scrittura dell'l-esimo valore dati, osservato all's-esimo passo in figura 4.1.

Le porte $\bar{c}X$ controllate da $\vec{d}^{(l)}$ risistemano gli stati della base computazionale dei qubit bus cosicché $|\vec{d}^{(l)}\rangle$ diventa $|1\rangle^{\otimes n}$, ed il resto dei bit quantistici si invertono di conseguenza:

$$|\psi_1\rangle_l = \psi_{\vec{d}^{(l)}} |1\rangle^{\otimes n} |0\rangle_R + \sum_{|\overline{j \oplus \vec{d}^{(l)}}\rangle \neq |1\rangle^{\otimes n}} \psi_j |\overline{j \oplus \vec{d}^{(l)}}\rangle |0\rangle_R . \tag{4.3}$$

La sovralinea nell'ultimo termine indica che l'inversione del bit avviene se il bit di controllo è 0. Dopo il passo 1, la rotazione controllata dai qubit, $C^n R_y(\theta^{(l)})$, denotata come $\theta^{(l)}$ nella figura, è applicata al qubit registro. Lo stato quantistico al passo 2 diventa

$$|\psi_2\rangle_l = \psi_{\vec{d}^{(l)}} |1\rangle^{\otimes n} |\theta^{(l)}\rangle_R + \sum_{|\overline{j \oplus \vec{d}^{(l)}}\rangle \neq |1\rangle^{\otimes n}} \psi_j |\overline{j \oplus \vec{d}^{(l)}}\rangle |0\rangle_R , \tag{4.4}$$

dove $|\theta\rangle = \cos\theta |0\rangle + \sin\theta |1\rangle$. Le porte $\bar{c}X$ condizionate da $\vec{d}^{(l)}$ sono applicate di nuovo per far ritornare alla condizione precedente lo stato dei bus:

$$|\psi_3\rangle_l = \psi_{\vec{d}^{(l)}} |\vec{d}^{(l)}\rangle |\theta^{(l)}\rangle_R + \sum_{j \neq \vec{d}^{(l)}} \psi_l |j\rangle |0\rangle_R . \tag{4.5}$$

Il secondo giro registra i dati successivi di $\vec{d}^{(l+1)}$ e $\theta^{(l+1)}$:

$$|\psi_4\rangle_{l,l+1} = \psi_{\vec{d}^{(l)}} |\vec{d}^{(l)}\rangle |\theta^{(l)}\rangle_R + \\ + \psi_{\vec{d}^{(l+1)}} |\vec{d}^{(l+1)}\rangle |\theta^{(l+1)}\rangle_R + \sum_{j \neq \vec{d}^{(l)}, \vec{d}^{(l+1)}} \psi_j |j\rangle |0\rangle_R . \tag{4.6}$$

Questo processo può essere ripetuto tante volte quanti sono i dati da inserire. In questo modo, M valori possono essere registrati con pesi non uniformi per generare lo stato

$$\sum_{l=0}^{M-1} \psi_{\vec{d}^{(l)}} |\vec{d}^{(l)}\rangle \left[\cos\theta^{(l)} |0\rangle_R + \sin\theta^{(l)} |1\rangle_R\right] + \sum_{j \notin \{\vec{d}^{(l)}\}} \psi_j |j\rangle |0\rangle_R . \tag{4.7}$$

Infine, il QDB richiesto nell'eq. 4.1 può essere ottenuto selezionando un angolo $\theta^{(l)}$ appropriato che sia legato all'ampiezza di probabilità desiderata b_l, e postselezionando il risultato di misura $|1\rangle_R$. La probabilità di misurare $|1\rangle_R$ è

$$P(1) = \sum_{l=0}^{M-1} |\psi_{\vec{d}^{(l)}} \sin\theta^{(l)}|^2. \tag{4.8}$$

Il costo per preparare uno stato con questo procedimento è di $\mathcal{O}(n)$ qubit e $\mathcal{O}(Mn)$ operazioni quantistiche.

4.2 QKNN

L'algoritmo KNN quantistico pesato sulla distanza evidenziato in questa sezione è stato proposto da Schuld, Sinayskiy e Petruccione [11] e implementato da Schuld, Fingerhuth e Petruccione [10] sul computer quantistico a 5 qubit dell'IBM. L'algoritmo viene qui introdotto, per poi discutere nel capitolo successivo i risultati.

L'obiettivo è di classificare una registro di qubit in entrata basandosi su un numero più o meno grande di configurazioni di qubit di apprendimento. Ogni configurazione di apprendimento appartiene ad una certa classe, che è codificata in un registro quantistico di classe in entanglement con la rispettiva configurazione di apprendimento. L'idea principale è che l'algoritmo KNN calcola la distanza tra il motivo in entrata ed ogni configurazione di apprendimento attraverso una serie di porte quantistiche. Successivamente, queste distanze sono invertite in modo che i vettori di apprendimento vicini a quello di input hanno l'inverso della distanza maggiore rispetto ai vettori di apprendimento più remoti. Queste distanze inverse sono poi scritte nelle corrispondenti ampiezze dello stato di classe. Si noti che il modulo quadro di un'ampiezza di un particolare stato di classe determina la probabilità di misurare quella classe. Quindi, la probabilità di misurare una data classe dipende adesso dall'inverso delle distanze tra i

vettori di apprendimento della classe in esame ed il vettore d'input. L'inverso della distanza può essere visto come un peso dipendente dalla distanza, dato che aumenta la probabilità di misurare la classe con i vettori di apprendimento più vicini a quello d'input. I passaggi necessari per l'implementazione dell'algoritmo KNN quantistico sono evidenziati in dettaglio qui di seguito.

4.2.1 Struttura dell'algoritmo

Il primo passo dell'algoritmo consiste nel preparare una *sovrapposizione dell'insieme di apprendimento* che ne contenga i dati codificati. Usando un idoneo schema di preparazione dello stato, il circuito di classificazione porta un sistema di n qubit nello stato

$$|D\rangle = \frac{1}{\sqrt{2MC}} \sum_{m=1}^{M} |m\rangle \left(|0\rangle |\psi_x\rangle + |1\rangle |\psi_{t^m}\rangle\right) |c^m\rangle . \tag{4.9}$$

Qui $|m\rangle$ è un registro indice che prende valori $m = 1, \ldots, M$ e che contrassegna l'm-esimo vettore di apprendimento. Il secondo registro è un singolo qubit ancilla, il cui stato eccitato è in entanglement con il terzo registro che codifica l'm-esimo stato di apprendimento $|\psi_{t^m}\rangle = \sum_{i=0}^{N-1} t_i^m |i\rangle$, mentre il suo stato fondamentale è in entanglement con il terzo registro che codifica il vettore d'input $|\psi_x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$. Il quarto registro codifica la classe nell'ampiezza dei propri qubit. Effettivamente, si crea una funzione d'onda che contiene i vettori di apprendimento insieme ad M copie del nuovo input. La costante di normalizzazione C dipende dal processo di ottimizzazione dei dati; assumeremo in seguito che i vettori di apprendimento siano normalizzati e quindi $C = 1$.

Dopo aver preparato lo stato iniziale, il circuito quantistico consiste di sole tre operazioni. Prima, l'applicazione di una porta Hadamard sull'ancilla fa interferire le copie del vettore d'input con i vettori di apprendimento:

$$\frac{1}{2\sqrt{M}} \sum_{m=1}^{M} |m\rangle \left(|0\rangle |\psi_{x+t^m}\rangle + |1\rangle |\psi_{x-t^m}\rangle\right) |c^m\rangle , \tag{4.10}$$

dove $|\psi_{x\pm t^m}\rangle = |\psi_x\rangle \pm |\psi_{t^m}\rangle$.

La seconda operazione è una misura condizionale che seleziona il ramo con l'ancilla nello stato $|0\rangle$. Questa postselezione ha successo con probabilità $p_{acc} = \frac{1}{4M} \sum_m |x + t^m|^2$. È più probabile che abbia successo se la distanza euclidea al quadrato complessiva dell'insieme dati di apprendimento rispetto al nuovo input è piccola. Se la misura condizionale ha successo, lo stato risultanta è dato da

$$\frac{1}{2\sqrt{M p_{acc}}} \sum_{m=1}^{M} \sum_{i=1}^{N} |m\rangle \left(x_i + t_i^m\right) |i\rangle |c^m\rangle . \tag{4.11}$$

Le ampiezze pesano i qubit classe $|c^m\rangle$ con la distanza dell'm-esimo vettore dati dal nuovo input. In questo stato, la probabilità di misurare il qubit classe nello stato $|s\rangle$

$$p(|c\rangle = |s\rangle) = \frac{1}{4Mp_{acc}} \sum_{m||c\rangle=|s\rangle} |x + t^m|^2, \tag{4.12}$$

riflette la probabilità di predire la classe s per il nuovo input.

La scelta di vettori caratteristica normalizzati assicura che $\frac{1}{4Mp_{acc}} \sum_m |x + t^m|^2 = 1 - \frac{1}{4Mp_{acc}} \sum_m |x - t^m|^2$, e la scelta della classe con maggior probabilità in questo modo implementa il classificatore. Questo significa che se il vettore d'input è molto vicino ai vettori di training di una data classe, quella classe uscirà come risultato più frequentemente delle altre.

5 RISULTATI E DISCUSSIONE

Qui discuteremo i risultati.

Per l'implementazione dell'algoritmo è stato scelto il ben noto Iris data set. Come evidenziato da Schuld[10], l'insieme dati si è dovuto standardizzare e normalizzare. Dopo di ciò, il primo passo è stato di ripetere l'esperienza dell'articolo per comprenderne appieno il funzionamento e poterlo estendere. In questo stato, si usano due caratteristiche delle quattro disponibili e si possono classificare solo due classi alla volta. Inoltre i vettori di apprendimento inseriti sono in numero estremamente esiguo e di certo insufficiente per garantire una classificazione efficiente.

Sono state seguite varie linee per migliorare l'algoritmo e renderlo più efficace:

- aumentare il numero di classi riconosciute;
- aumentare il numero di caratteristiche considerate;
- aumentare il numero di vettori di training.

Discutiamo con ordine i passi effettuati

Per implementare l'algoritmo si è usato il famoso data set Iris, consistente in misure di lunghezze e larghezze di sepali e petali di tre varietà di iris.

5.1 OTTIMIZZAZIONE DEI DATI

Inizialmente i dati si presentano nella forma di fig. 5.1.

La standardizzazione prevede che si traslino i dati affinché abbiano media nulla, dopo di che li scala in modo che abbiano deviazione standard unitaria (vedi fig. 5.2).

A questo punto normalizziamo ciascun vettore (vedi fig. 5.3).

Avendo effettuato queste operazioni preliminari, si deve ora tradurre le coordinate di ciascun vettore nello spazio delle caratteristiche in un angolo di rotazione da applicare ad un qubit nella QRAM.

Avendo visto nell'Eq. 4.8 che lo stato costruito, dopo la misura condizionale, è

$$\sum_{l=0}^{M-1} \psi_{\vec{d}^{(l)}} |\vec{d}^{(l)}\rangle \sin\theta^{(l)} |1\rangle_R \,, \tag{5.1}$$

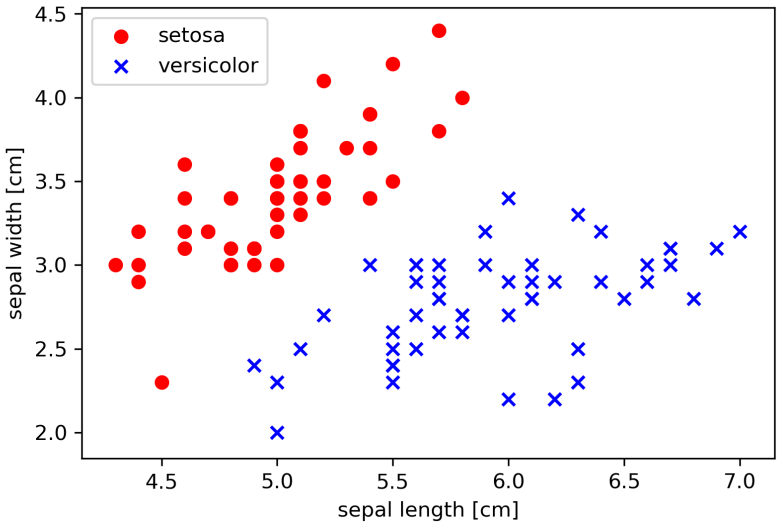


Figura 5.1: Dati grezzi

troviamo che la relazione esistente tra i valori b_l, ovvero le caratteristiche standardizzate e normalizzate, e le ampiezze di probabilità nella QRAM è

$$\theta^{(l)} = \arcsin b_l. \tag{5.2}$$

5.2 SCRITTURA DELL'ALGORITMO

Seguendo le istruzioni si costruisce il seguente circuito quantistico.

Si noti che questo è espresso usando solo le porte quantistiche presenti nell'insieme universale di base messo a disposizione dall'IBM QX. Possiamo infatti notare come le porte $C_n R_y(\theta)$ siano in realtà realizzate attraverso una successione di più porte.

Possiamo seguire i passaggi osservando la figura:

1. i qubit ancilla ed indice vengono posti in sovrapposizione uniforme;
2. il vettore d'input x è posto in entanglement con lo stato fondamentale dell'ancilla;
3. il vettore di training t^0 è posto in entanglement con lo stato eccitato dell'ancilla e con lo stato fondamentale del qubit indice;
4. il vettore di training t^1 è posto in entanglement con lo stato eccitato dell'ancilla e del qubit indice;
5. il qubit classe è invertito condizionato dal fatto che il qubit indice sia $|1\rangle$; questo completa la preparazione iniziale dello stato;

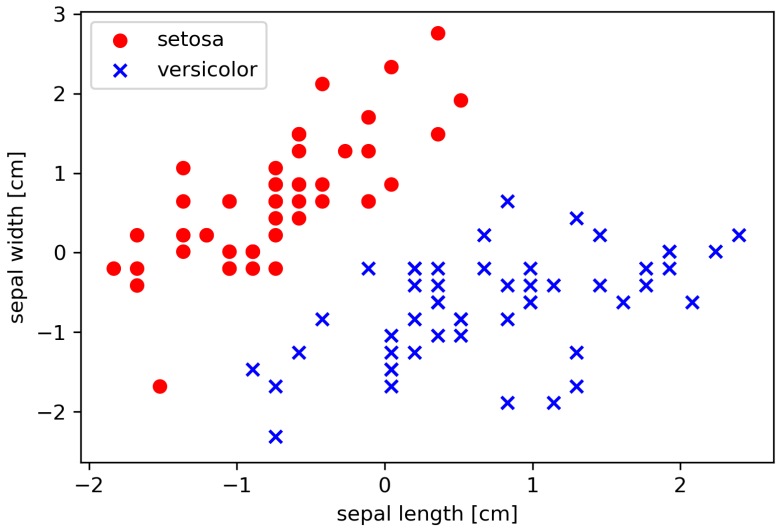


Figura 5.2: Dati standardizzati

6. nell'ultimo passaggio la porta Hadamard fa interferire le copie di x con i vettori d'apprendimento e l'ancilla è misurata, seguita da una misura del qubit classe.

Considereremo validi per i nostri scopi solo le misure del qubit classe ottenute quando il qubit ancilla è trovato nello stato $|0\rangle$.

Si mostra un esempio di risultato di classificazione usando le prime due caratteristiche del data set e gli elementi appartenenti alle classi setosa e versicolor. Si scelgono come vettori di training un elemento per ciascuna classe, nel nostro caso il vettore 34 ed il vettore 86 dal data set, rispettivamente setosa e versicolor. Si assegna alla classe setosa lo stato $|0\rangle$ del qubit classe ed alla classe versicolor lo stato $|1\rangle$. Si sottopongono poi a classificazione, in due esperimenti separati, due vettori sconosciuti: il vettore 29 (setosa) ed il vettore 57 (versicolor).

Il primo passo è simulare l'esperimento sul computer in uso, dato che il problema in esame è facilmente eseguibile su un comune portatile. I risultati non filtrati per i due esperimenti sono visibili in figura. Nella didascalia sono scritti i conteggi corrispondenti ad un determinato esito di misura sui due qubit ancilla e classe. La cifra sulla destra contiene la misura del qubit ancilla, quella sulla sinistra del qubit classe. Tali valori, normalizzati ad uno, sono rappresentati in un istogramma, che approssima la distribuzione di probabilità degli esiti di misura per grandi numeri di esecuzioni dell'algoritmo. Per questo motivo si userà preferibilmente un numero di esecuzioni pari al massimo permesso sui computer quantistici dell'IBM, ovvero 8192.

Selezionando i risultati laddove il bit di destra è 0, abbiamo praticamente effettuato la misura condizionale necessaria al funzionamento dell'algoritmo.

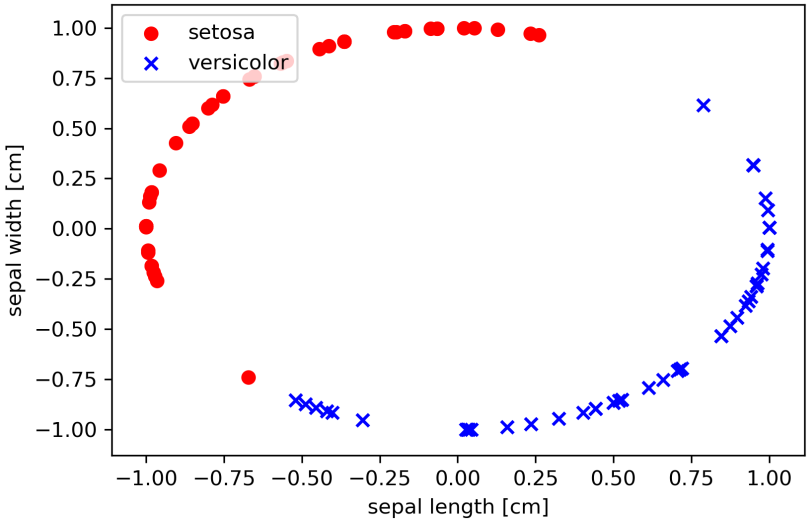


Figura 5.3: Dati normalizzati

Il passo successivo è eseguire questi stessi circuiti quantistici su un vero computer quantistico. È usata scelta la macchina a 14 qubit ibmq_16_melbourne per ottenere le seguenti misure per il vettore d'input setosa.

Gli inevitabili fenomeni di rumore rendono i risultati meno piccati ma comunque distinguibili in questo caso.

Si noti che fino ad ora non è stato fatto uso delle tecniche di costruzione QRAM introdotte nella sezione 4.1.1.

L'obiettivo principale di questa tesi è cercare di ottenere un algoritmo multiclasse generico a partire da quello appena discusso. Si può dire che le seguenti sottosezioni segnano il vero e proprio punto di inizio del lavoro originale in questa tesi[1].

Sebbene si potrebbe mirare subito ad aumentare il numero di classi riconosciute, dobbiamo tenere conto di un problema che si deve affrontare quando si ha a che fare con data set simili ad Iris: le classi versicolor e virginica non sono ben distinguibili; infatti se avessimo effettuato una misura considerando solo due feature, come abbiamo appena fatto, avremmo ottenuto risultati vaghi (probabilità intorno al 50% per entrambi i risultati) o addirittura erronei.

Un modo di aggirare il problema consiste nell'uso di feature map[10], ovvero funzioni polinomiali che, applicate al data set, ci restituiscono una nuova configurazione dei vettori di apprendimento da usare per il training e per la classificazione. Usando una feature map appropriata, si possono separare in maniera sufficiente vettori appartenenti a classi diverse e migliorare l'efficienza del riconoscimento.

1 È possibile trovare lavori analoghi a questo in rete, si veda a titolo di esempio gli algoritmi sviluppati da Carsten Blank su https://github.com/carstenblank/dc-qiskit-qml

Questa tesi non adotta tale tecnica, in quanto il suo scopo è verificare come scala in grandezza l'algoritmo KNN quantistico appena illustrato. Si tenta dunque di migliorare l'efficienza del riconoscimento aumentando il numero di caratteristiche, dunque la dimensionalità, dell'insieme dei vettori di training.

5.2.1 Aumento delle numero di caratteristiche

La maggior parte dei vettori versicolor e virginica non vengono classificati correttamente, qui abbiamo selezionato un caso in cui la classificazione è corretta fin dal caso più semplice.

Per verificare il miglioramento apportato da un aumento del numero di caratteristiche considerate, si prende in esame la classificazione del vettore d'input 54 (versicolor), con i vettori di training numero 51 (versicolor) e 146 (virginica) del data set. Il vettore d'input viene classificato correttamente durante la simulazione con probabilità vicina al 51.4%. Effettuando la stessa misura, ma tenendo conto di tutte le quattro feature, arriviamo ad una probabilità di classificazione corretta del 58.3% nel migliore dei casi. Questo è un segno che l'implementazione ha un margine di miglioramento apprezzabile.

5.2.2 Aumento del numero di vettori d'apprendimento

Per poter implementare più di due classi, è necessario essere capaci di memorizzare almeno tre vettori di apprendimento, per poi associare ad ognuno di essi la rispettiva classe. L'aggiunta di vettori di training, riprendendo in esame il confronto tra classi versicolor e virginica, non ha effetti ben prevedibili, almeno per piccoli insiemi. Aggiungendo altri due vettori d'apprendimento si possono migliorare o peggiorare i risultati, a seconda della disposizione dei vettori nello spazio delle caratteristiche; si rende necessario un criterio di selezione dei vettori più significativi per ottenere un classificatore il più efficace possibile, non potendo inserire tutti i vettori del data set, e tantomeno potendo scegliere a caso i vettori finché non si riscontra il risultato aspettato (il presupposto di un algoritmo maturo è che una volta ultimato non debba aver bisogno di aggiustamenti significativi da parte dell'operatore umano). Oltretutto una scelta manuale degli specifici vettori di training potrebbe andare bene per i vettori d'input su cui si sta concentrando in un dato momento, ma non si può conoscere l'efficacia rispetto a eventuali nuovi vettori sconosciuti.

5.2.3 Implementazione multiclasse

L'aggiunta della capacità di riconoscere tutte le tre classi in una sola esecuzione non è di natura differente dall'aggiungere qubit per avere maggiori vettori di addestramento. Passiamo dall'avere un solo qubit classe, che ha associato il proprio stato $|0\rangle$ alla prima classe e lo stato $|1\rangle$ alla seconda, ad avere due qubit ed associare lo stato $|00\rangle$ alla prima

classe, lo stato $|01\rangle$ alla seconda classe e lo stato $|10\rangle$ alla terza classe. Lo stato $|11\rangle$ resta inutilizzato.

Nel processo di costruzione della QRAM contenente lo stato iniziale, oltre che ai qubit indice, i vettori di addestramento saranno in entanglement anche con i due qubit classe, nella maniera appropriata per ogni vettore.

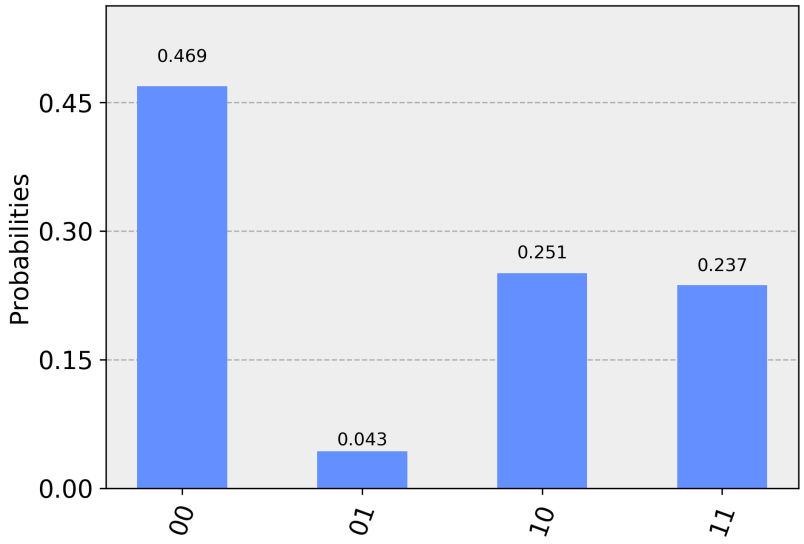
5.3 ESECUZIONE COMPLETA

Per studiare l'efficienza dell'algoritmo a diversi stadi di miglioramento, si divide il data set in un insieme dedicato all'addestramento ed un insieme di vettori da classificare. Al fine di avere risultati statisticamente rilevanti, per ogni esecuzione i vettori di training ed il vettore d'input sono scelti casualmente a partire dal data set completo. Si contano le classificazioni di successo rispetto al totale dei tentativi, al variare dei parametri come il numero di features o il numero di vettori di training usati.

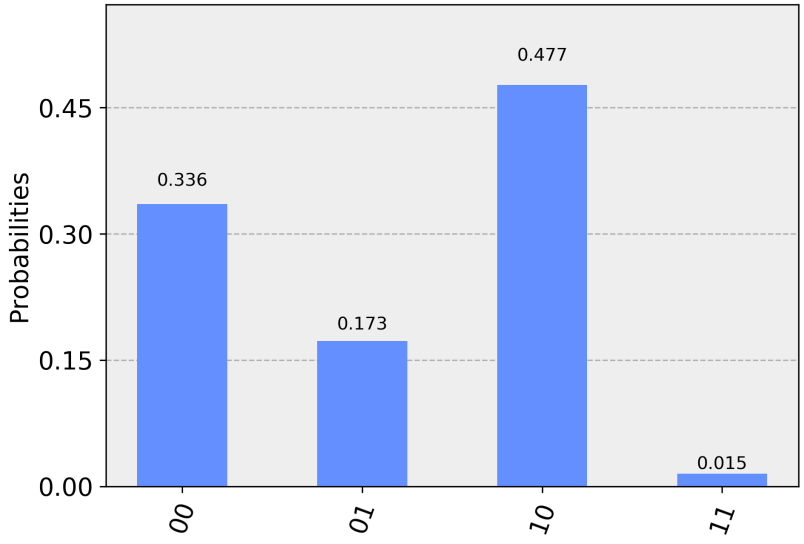
Provando ad effettuare una classificazione a tre classi con 32 vettori di training otteniamo i seguenti esiti:

- i vettori della classe setosa sono correttamente classificati 10 volte su 10;
- i vettori della classe versicolor sono correttamente classificati 5 volte su 10;
- i vettori della classe virginica sono correttamente clasificati 9 volte su 10.

I risultati vengono necessariamente da simulazioni, in quanto sono necessari 19 qubit sotto queste condizioni.

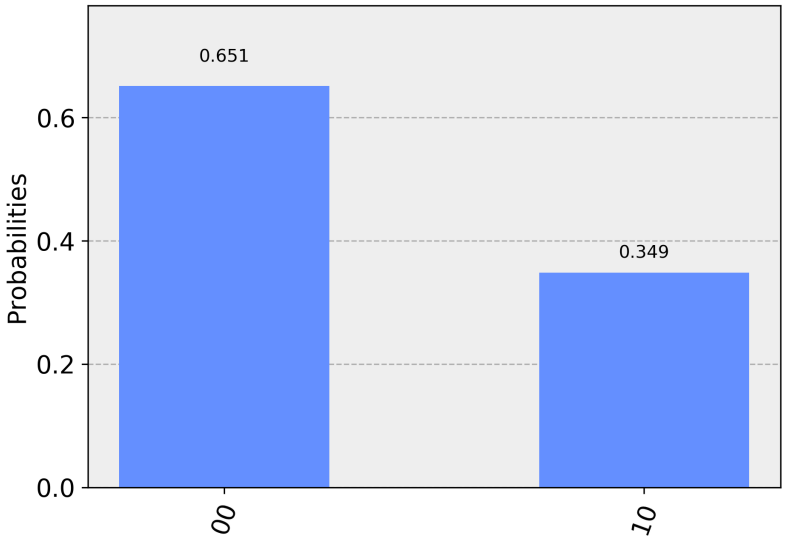


(a) Iris setosa

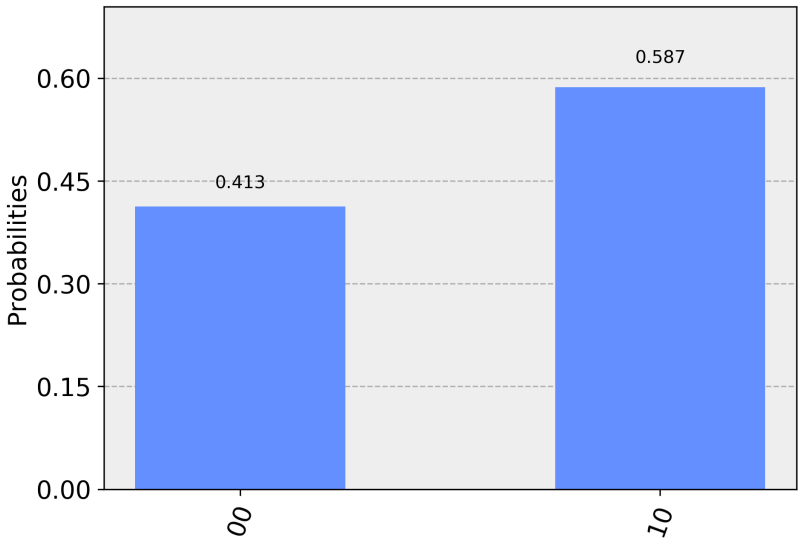


(b) Iris versicolor

Figura 5.4: Simulazione del circuito. I conteggi totali sono per setosa: {'00': 3843, '10': 2056, '01': 352, '11': 1941}; per versicolor: {'00': 2749, '10': 3908, '01': 1414, '11': 121}.



(a) Iris setosa



(b) Iris versicolor

Figura 5.5: Simulazione del circuito, risultati filtrati

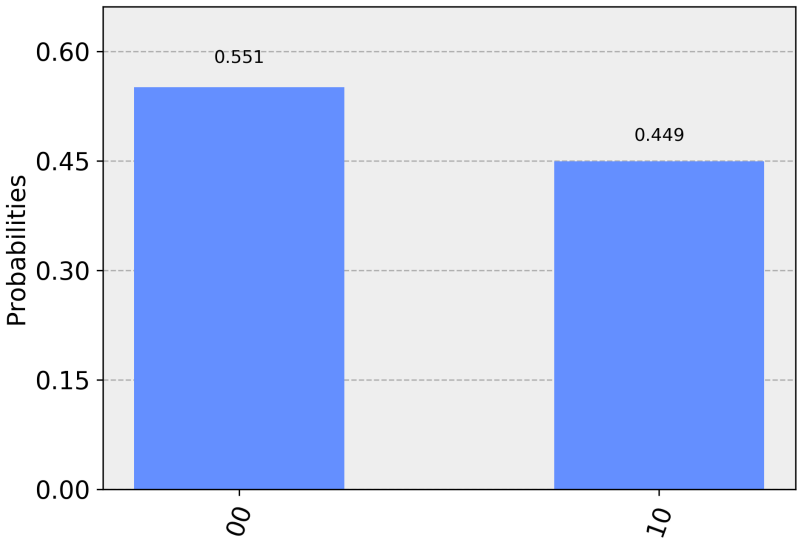


Figura 5.6: Esecuzione su hardware reale (setosa)

6 CONCLUSIONE

Parte III

APPENDICE

A APPENDICE

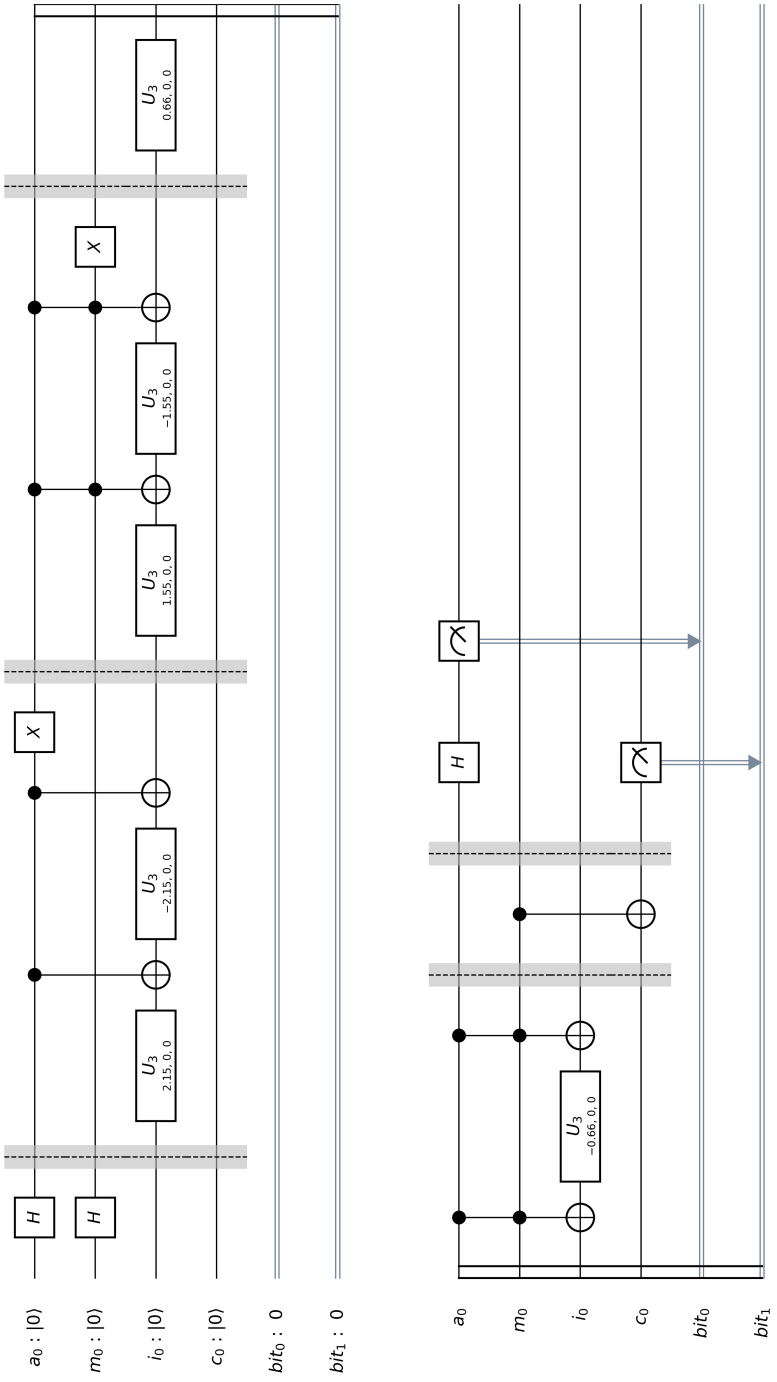


Figura A.1: Il circuito quantistico.

BIBLIOGRAFIA

[1] *Android 10*. https://www.android.com/android-10/. [Online; recuperato il 7 settembre 2019]. 2019.

[2] Mark Fingerhuth. «Quantum-enhanced machine learning: Implementing a quantum k-nearest neighbour algorithm». In: *Bachelor Thesis, Maastricht University* (2017).

[3] Ralph Jacobson. «2.5 quintillion bytes of data created every day. How does CPG & Retail manage it?» In: *IBM Consumer Products Industry Blog* (2013). Recuperato il 6 settembre 2019. URL: https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it/.

[4] Will Knight. *IBM Raises the Bar with a 50-Qubit Quantum Computer*. https://www.technologyreview.com/s/609451/ibm-raises-the-bar-with-a-50-qubit-quantum-computer/. [Online; recuperato il 7 settembre 2019].

[5] Donald E. Knuth. «Computer Programming as an Art». In: *Communications of the ACM* 17.12 (1974), pp. 667–673.

[6] U. Las Heras, U. Alvarez-Rodriguez, E. Solano e M. Sanz. «Genetic Algorithms for Digital Quantum Simulations». In: *Phys. Rev. Lett.* 116 (23 2016), p. 230504. DOI: 10.1103/PhysRevLett.116.230504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.116.230504.

[7] Daniel K. Park, Francesco Petruccione e June-Koo Kevin Rhee. «Circuit-Based Quantum Random Access Memory for Classical Data». In: *Scientific Reports* 9.3949 (2019). DOI: 10.1038/s41598-019-40439-3.

[8] *Quantum Computing at IBM*. https://www.ibm.com/quantum-computing/learn/what-is-ibm-q. [Online; recuperato il 7 settembre 2019].

[9] Sebastian Raschka e Vahid Mirjalili. *Python Machine Learning*. 2ª ed. Packt, 2017. ISBN: 978-1-78712-593-3.

[10] M. Schuld, M. Fingerhuth e F. Petruccione. «Implementing a distance-based classifier with a quantum interference circuit». In: *EPL (Europhysics Letters)* 119.6 (2017). DOI: 10.1209/0295-5075/119/60002.

[11] Maria Schuld, Ilya Sinayskiy e Francesco Petruccione. «Quantum Computing for Pattern Classification». In: *PRICAI 2014: Trends in Artificial Intelligence*. A cura di Duc-Nghia Pham e Seong-Bae Park. Cham: Springer International Publishing, 2014, pp. 208–220. ISBN: 978-3-319-13560-1.

[12] P. W. Shor. «Algorithms for quantum computation: Discrete logarithms and factoring». In: *Proc. 35nd Annual Symposium on Foundations of Computer Science (Shafi Goldwasser, ed.), IEEE Computer Society Press* (1994), pp. 124–134.

[13] Francesco Tacchino, Chiara Macchiavello, Dario Gerace e Daniele Bajoni. «An artificial neuron implemented on an actual quantum processor». In: *npj Quantum Information* (2019). URL: https://doi.org/10.1038/s41534-019-0140-4.

[14] Kristan Temme e Jay Gambetta. *Researchers Put Machine Learning on Path to Quantum Advantage*. https://www.ibm.com/blogs/research/2019/03/machine-learning-quantum-advantage/. [Online; recuperato il 7 settembre 2019]. 2019.

[15] Noson S. Yanofsky e Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2013.

DECLARATION

Put your declaration here.

Napoli, Ottobre 2019

Mariano Mollo

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Thank you very much for your feedback and contribution.