

IMPLEMENTAZIONE DI UN ALGORITMO KNN MULTICLASSE SU HARDWARE QUANTISTICO

TESI DI LAUREA SPERIMENTALE IN FISICA

MARIANO MOLLO N85000880

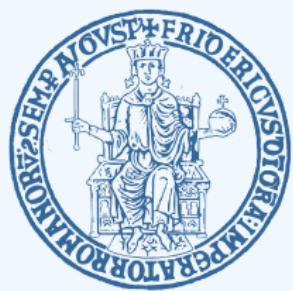
RELATORI:

GIOVANNI ACAMPORA

AUTILIA VITIELLO

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

OTTOBRE 2019



INTRODUZIONE



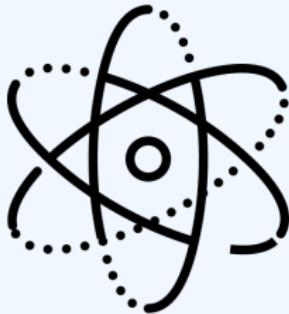
Il machine learning
permette ai
computer di
imparare dai dati

Gli algoritmi di ML prevedono spesso di

- risolvere grandi sistemi di equazioni lineari
- invertire grandi matrici
- calcolare distanze

Effettuare questi calcoli su insiemi dati
grandi e complessi diventa difficile

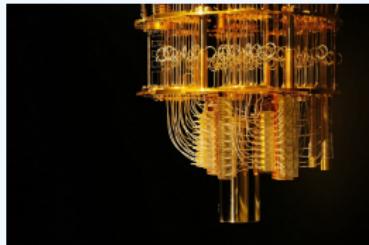
QUANTUM COMPUTING



Il quantum computing studia la costruzione e l'uso di hardware di elaborazione basato sulla meccanica quantistica

- Il quantum computing lavora con vettori in spazi di Hilbert complessi
- I computer quantistici eseguono operazioni lineari sui qubit
- Sistemi a molti qubit sono descritti da grandi vettori che possono essere manipolati in parallelo
- Il machine learning prevede la manipolazione di grandi vettori e matrici

QUANTUM MACHINE LEARNING



L'uso dei computer quantistici per risolvere problemi classici difficili o classi di problemi completamente nuove è chiamato machine learning quantistico, ovvero permettere ai computer quantistici di imparare dai dati più velocemente dei computer classici

DOMANDA DI RICERCA

È possibile implementare su un computer quantistico un algoritmo k-nearest neighbours multiclasse, in modo da migliorare le prestazioni ed il numero di problemi risolvibili?

OBIETTIVI

- Riprodurre l'algoritmo di classificazione binaria KNN quantistico proposto da Schuld et al.
- Implementarne una versione multiclass
- Analizzare le capacità dell'algoritmo usando l'hardware quantistico attualmente disponibile
- Analizzare esperimenti più complessi tramite simulazione

MACHINE LEARNING

Il machine learning è un ramo dell'intelligenza artificiale che permette ai computer di apprendere dai dati. L'apprendimento può formalizzarsi attraverso la definizione di modelli matematici, usati per

- effettuare previsioni (apprendimento supervisionato)
- trovare regolarità in processi complessi (apprendimento non supervisionato)
- effettuare scelte per ottenere un risultato ottimale (apprendimento per rinforzo)

Definizione del problema

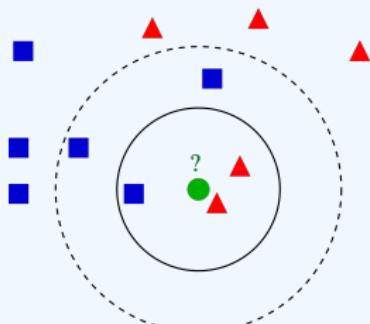
Dato un insieme dati in input con i corrispondenti output, predire l'output di un nuovo input ignoto.

Un tipico esempio di compito di classificazione multiclasse è il riconoscimento di caratteri scritti a mano.

In questo caso si può raccogliere un insieme dati di addestramento che consiste di molteplici caratteri scritti a mano di ciascuna lettera dell'alfabeto.

K-NEAREST NEIGHBOURS CLASSICO

L'algoritmo KNN è uno tra i più semplici del ML ed è un lazy learner



k è un numero naturale

Dato un insieme di apprendimento

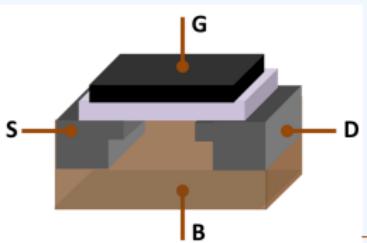
$D = v_0, \dots, v_{M-1}, v_i \in \{\text{classe}_0, \text{classe}_1\}$

Dato un nuovo vettore x :

- considera i k elementi più vicini ad x
- classifica x con un voto a maggioranza

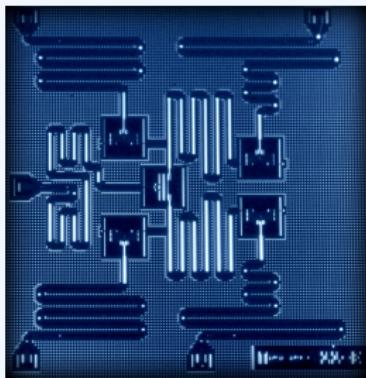
Si assegnano pesi dipendenti da $\frac{1}{\text{distanza}}$ per aumentare l'influenza dei vettori più vicini

QUANTUM COMPUTING



- Solitamente implementati attraverso MOSFET^a
- 2 stati definiti: 0 e 1
- Può trovarsi in uno tra gli stati 0 o 1

^aMOSFET: Metal Oxide Semiconductor Field Effect Transistor



- Implementato in diversi modi: giunzioni superconduttrive, ioni intrappolati, fotoni polarizzati...
- Due stati definiti: $|0\rangle$ e $|1\rangle$
- Può trovarsi in una sovrapposizione degli stati $|0\rangle$ e $|1\rangle$

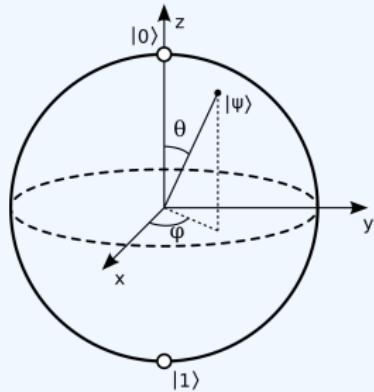
Matematicamente, la sovrapposizione di un qubit è espressa come

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \alpha, \beta \in \mathbb{C},$$

dove α e β sono chimate ampiezze di probabilità.

L'ultima espressione è chiamata vettore di probabilità.

SFERA DI BLOCH



Un qubit si può visualizzare su una 2-sfera parametrizzando α e β in coordinate polari

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

dove $0 \leq \theta \leq \pi$ e $0 \leq \varphi < 2\pi$

REGISTRO DI 2 QUBIT

Un computer quantistico con n qubit ha 2^n ampiezze di probabilità.

Lo stato di un registro a più qubit è rappresentato dal prodotto tensoriale dello stato dei singoli qubit.

$$|00\rangle = |0\rangle \otimes |0\rangle$$

$$|\psi\rangle = c_0 |00\rangle + c_1 |01\rangle + c_2 |10\rangle + c_3 |11\rangle = \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

REGISTRO DI n QUBIT

Un computer quantistico con n qubit ha 2^n ampiezze di probabilità.

Lo stato di un registro a più qubit è rappresentato dal prodotto tensoriale dello stato dei singoli qubit.

$$|00\dots00\rangle = |0\rangle \otimes |0\rangle \otimes \dots \otimes |0\rangle \otimes |0\rangle$$

$$|\psi\rangle = c_0 |00\dots00\rangle + c_1 |00\dots01\rangle + \dots + c_{n-2} |11\dots10\rangle +$$

$$+ c_{n-1} |11\dots11\rangle = \begin{matrix} 00\dots00 \\ 00\dots01 \\ \vdots \\ 11\dots10 \\ 11\dots11 \end{matrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix}$$

PORTE LOGICHE QUANTISTICHE

Per manipolare n qubit esistono apposite porte logiche quantistiche, che sono operatori rappresentabili come matrici $2^n \times 2^n$.



Figure: Hadamard

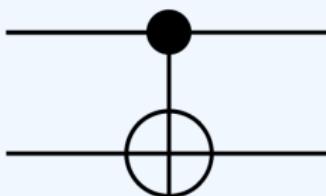


Figure: CNOT

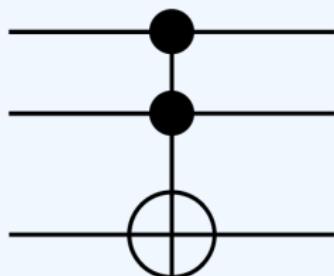


Figure: Toffoli

PUNTI DI FORZA DEI QUANTUM COMPUTER

# di qubit	RAM classica richiesta
5	256 byte
25	2 gigabyte
50	8000 terabyte
275	numero di atomi nell'universo osservabile

Le n ampiezze di probabilità possono essere usate per memorizzare quantità enormi di informazioni. Possiamo inoltre creare e lavorare su più copie in parallelo degli stessi dati.

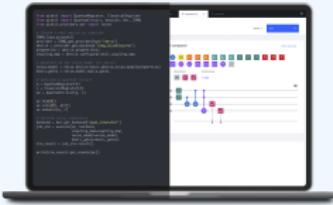
STATO DELL'ARTE



L'IBM Q System One è il primo computer quantistico a circuiti commerciali al mondo, introdotto dall'IBM nel gennaio 2019. L'IBM Q System One possiede 20 qubit.

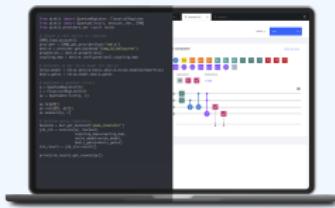
METODI

IBM Q EXPERIENCE



L'IBM Q Experience è un'interfaccia per interagire con le risorse di quantum computing dell'IBM

- accessibile al pubblico
- permette simulazioni con e senza rumore
- fino a 14 qubit superconduttori
- fino a 32 qubit simulati

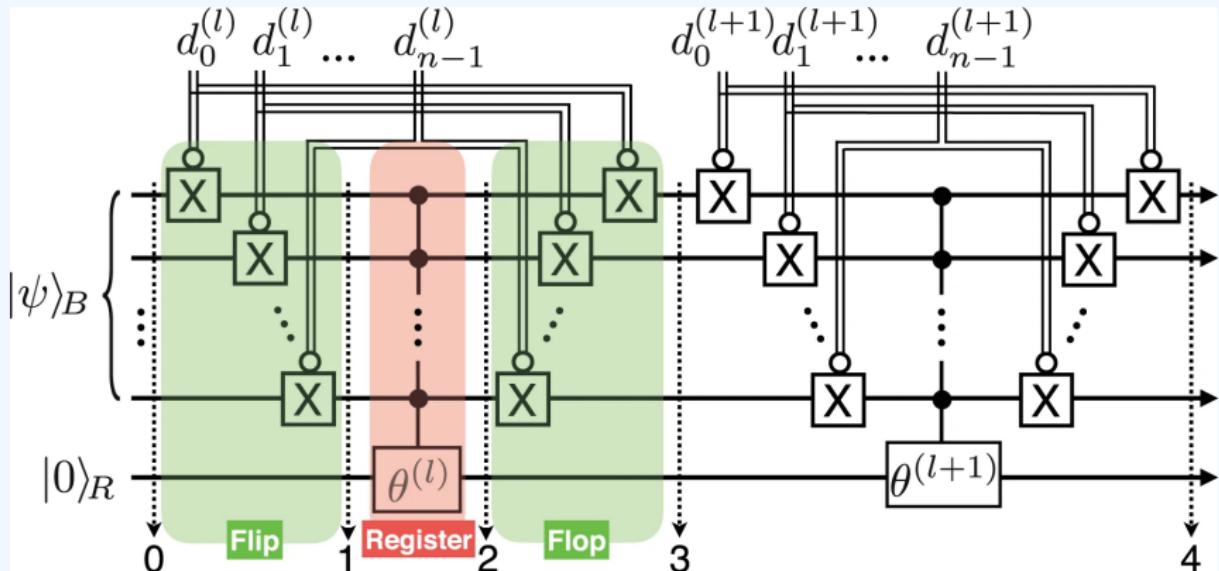


Struttura open source di sviluppo software per quantum computing, permette di

- progettare circuiti quantistici
- simularli sul proprio computer personale
- inviare ordini di esecuzione su hardware quantistico reale
- visualizzare i risultati

QUANTUM MACHINE LEARNING

CODIFICARE DATI CLASSICI NELLE AMPIEZZE



Per codificare dati classici nelle ampiezze di probabilità è stata usata la tecnica di costruzione di stati flip-flop QRAM, come proposto da Petruccione et al.

FF-QRAM

La FF-QRAM è usata per memorizzare un QDB inizializzato in maniera arbitraria.

L'operazione QRAM sui qubit sovrappone un insieme di dati classici $D = \left\{ \left(\vec{d}^{(l)}, b_l \right) \middle| 0 \leq l < M \right\}$ come

$$\text{QRAM}(D) \sum_j \psi_j |j\rangle_B |o\rangle_R \equiv \sum_l \psi_l |\vec{d}^{(l)}\rangle_B |b_l\rangle_R ,$$

dove $\vec{d}^{(l)}$ rappresenta un indirizzo di memoria con n bit di informazione e b_l è l'attributo ad esso associato.

ALGORITMO KNN QUANTISTICO

Stato quantistico iniziale

$$|\psi_0\rangle = \frac{1}{\sqrt{2M}} \sum_{m=1}^M (|0\rangle |\psi_x\rangle + |1\rangle |\psi_{t^m}\rangle) |c^m\rangle |m\rangle$$

Calcolo della distanza con interferenza quantistica

$$|\psi_1\rangle = \frac{1}{2\sqrt{M}} \sum_{m=1}^M \left(|0\rangle (|\psi_x\rangle + |\psi_{t^m}\rangle) + |1\rangle (|\psi_x\rangle - |\psi_{t^m}\rangle) \right) |c^m\rangle |m\rangle$$

Misura condizionale

$$|\psi_2\rangle = \frac{1}{2\sqrt{M}} \sum_{m=1}^M \sum_{i=1}^N (x_i + t_i^m) |0\rangle |i\rangle |c^m\rangle |m\rangle$$

ALGORITMO KNN QUANTISTICO

Probabilità di misurare una data classe

$$P(|c^m\rangle = |s\rangle) = \sum_{m|c^m=s} 1 - \frac{1}{4M} |x - t^m|^2$$

Classificazione

$$c = \begin{cases} 0 & \text{se } P(|c^0\rangle) \text{ maggiore} \\ 1 & \text{se } P(|c^1\rangle) \text{ maggiore} \\ \text{etc...} & \end{cases}$$

IMPLEMENTAZIONE

PREPARAZIONE DEI DATI

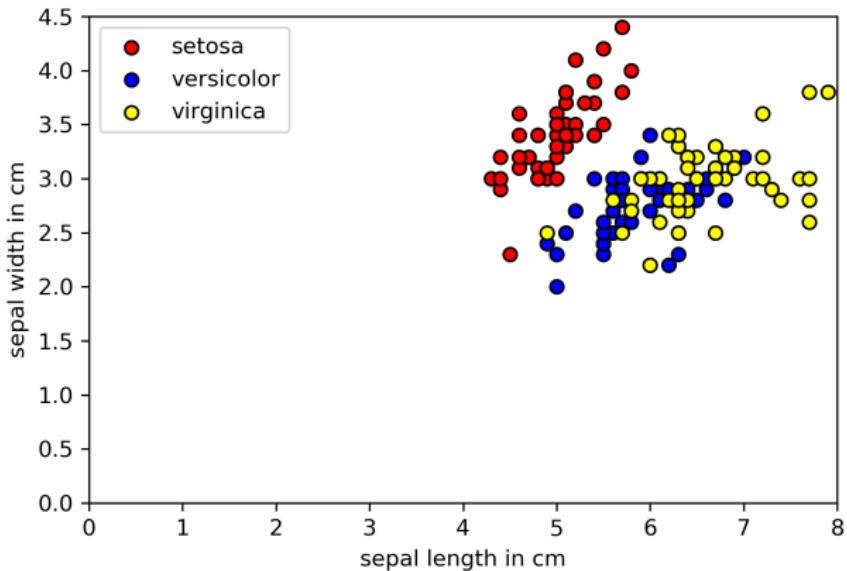


Figure: Data set Iris

PREPARAZIONE DEI DATI

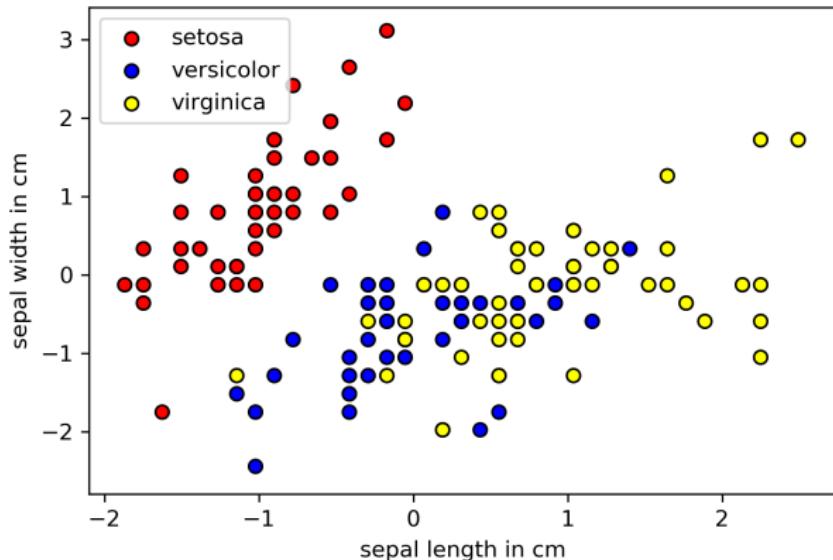


Figure: Data set Iris standardizzato

PREPARAZIONE DEI DATI

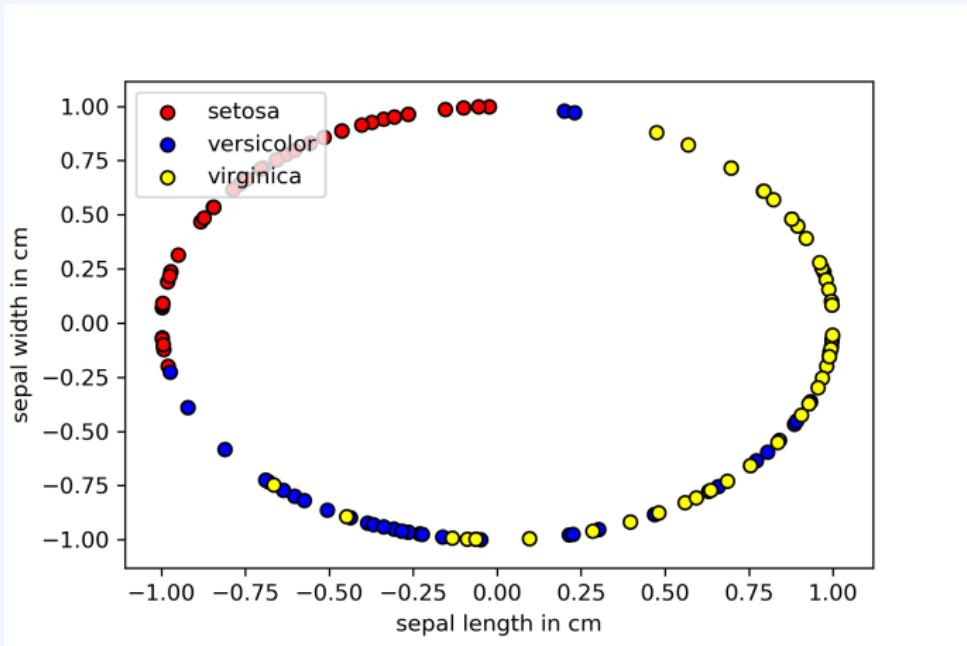


Figure: Data set Iris normalizzato (2 caratteristiche)

PREPARAZIONE DEI DATI

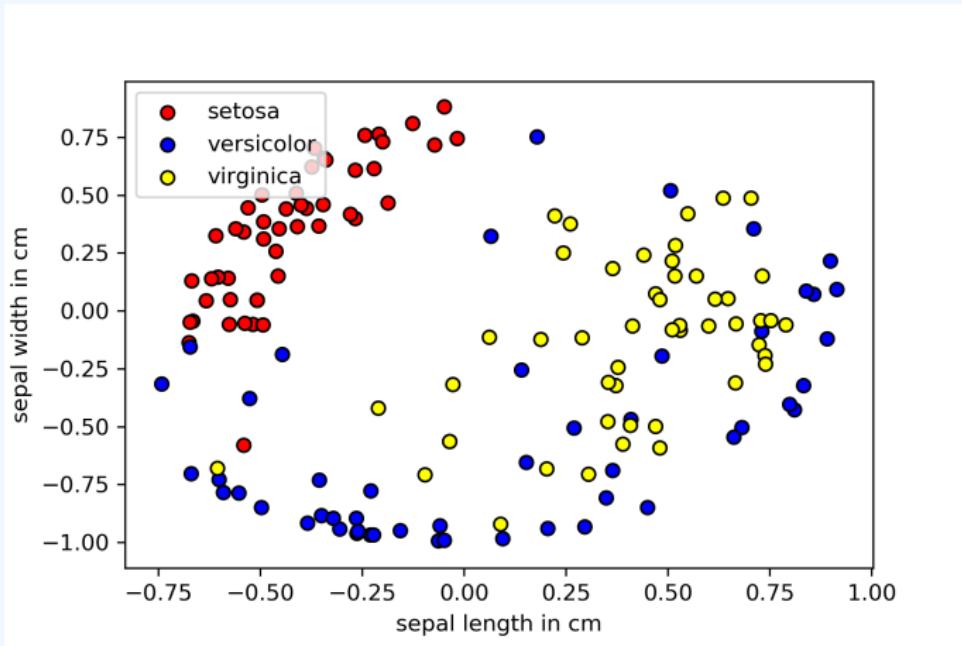
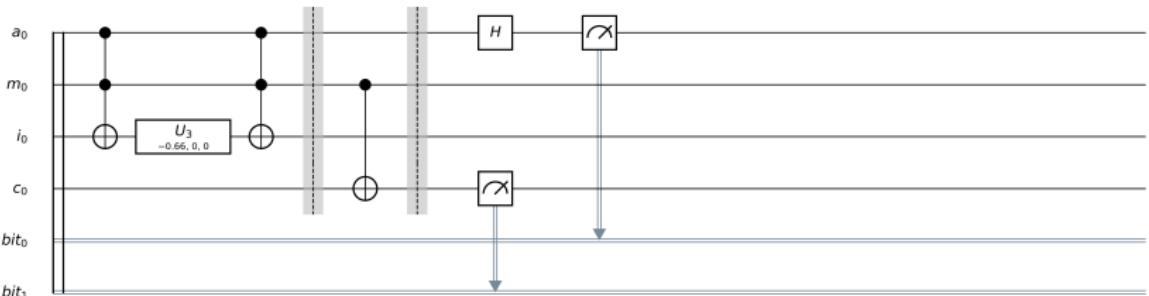
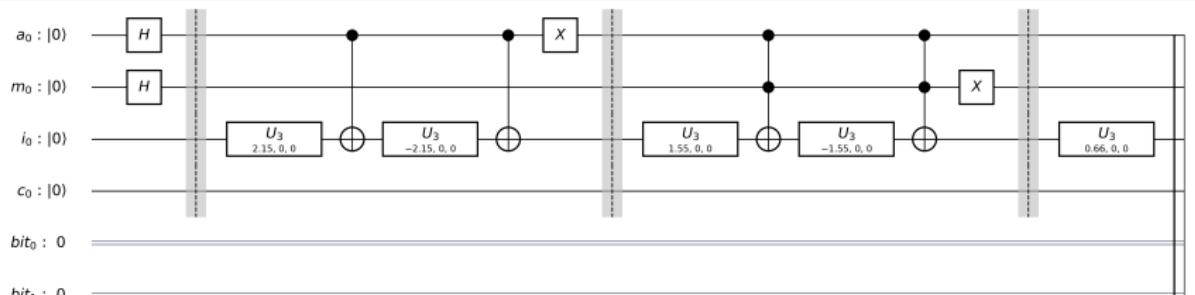


Figure: Data set Iris normalizzato (4 caratteristiche)

IL CIRCUITO QUANTISTICO

Il circuito quantistico per un classificatore binario è il seguente



Per il classificatore multiclasse il disegno è più complicato.

RISULTATI

CLASSIFICAZIONE BINARIA

Classificazione setosa vs. versicolor dal data set Iris

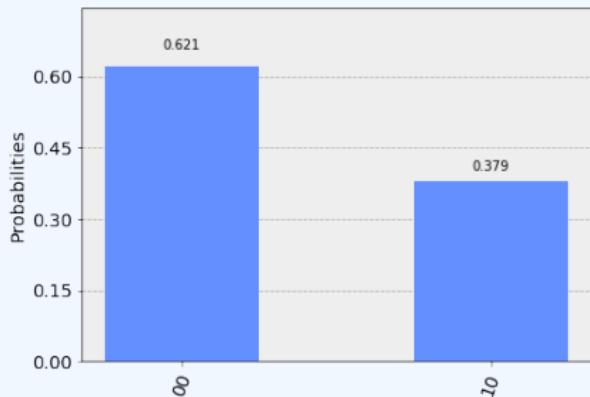


Figure: Simulazione su setosa

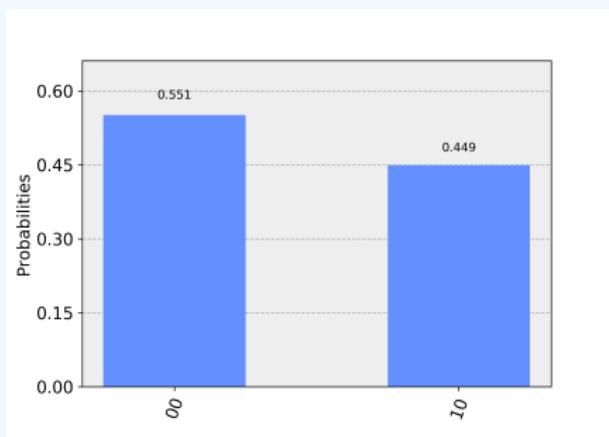


Figure: Esecuzione reale su setosa

CLASSIFICAZIONE MULTICLASSE

Classificazione setosa vs. versicolor vs. virginica dal data set Iris

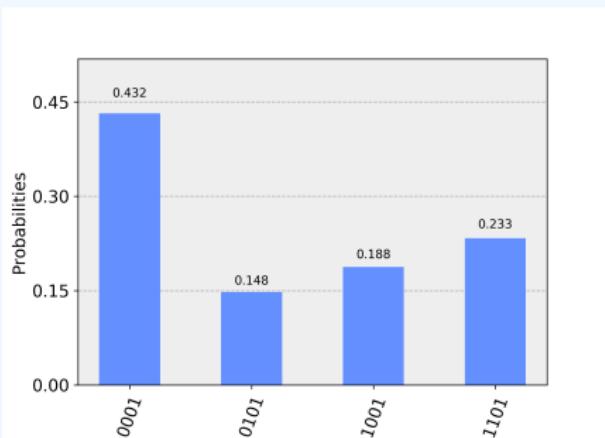


Figure: Simulazione su setosa

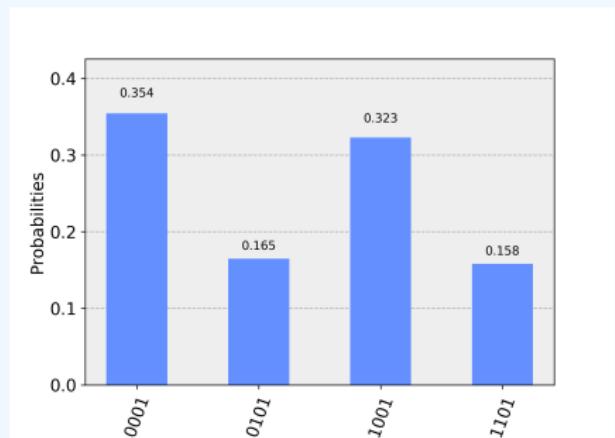


Figure: Esecuzione reale su setosa

CLASSIFICAZIONE MULTICLASSE

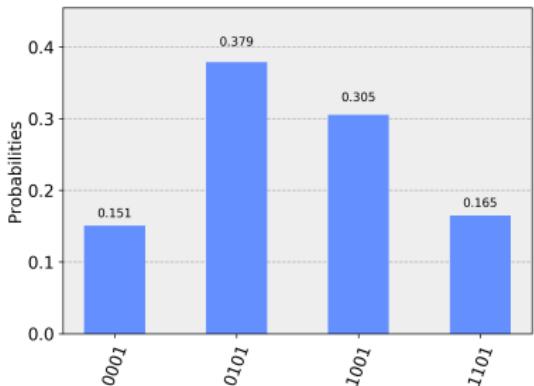


Figure: Simulazione su versicolor

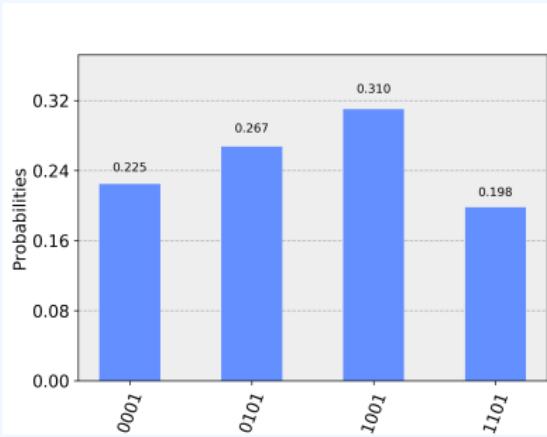


Figure: Simulazione su virginica

Nota: i risultati per la classe versicolor sono molto variabili.

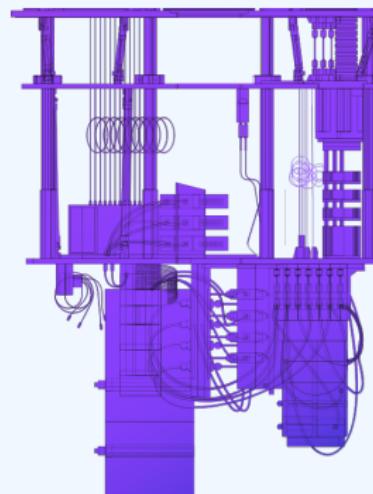
CONCLUSIONE

RIASSUNTO

- L'elaborazione quantistica è nella frontiera dei supercomputer e ha il potenziale di accelerare gli algoritmi di machine learning classico
- È stata riprodotta un'implementazione di algoritmo KNN quantistico di classificazione binaria su hardware di piccola scala
- Se ne è esteso il funzionamento in modo da renderlo multiclass
- Si sono effettuati test su hardware quantistico di media scala

PROSPETTIVE

- Far girare gli algoritmi su computer con maggiori risorse, sia in termini di numero di qubit che di tempi di decoerenza
- A tal proposito, sarebbe interessante l'esecuzione sul computer a 20 qubit annunciato quest'anno
- Si attende lo sviluppo di corrispettivi quantistici per algoritmi di IA più complessi



DOMANDE?

ALGORITMO PER CLASSIFICATORE BINARIO

Inizializzazione dei registri quantistici

```
a = QuantumRegister(1, 'a')
m = QuantumRegister(1, 'm')
i = QuantumRegister(1, 'i')
c = QuantumRegister(1, 'c')
b = ClassicalRegister(2, 'bit')
circuit = QuantumCircuit(a, m, i, c, b)
```

Sovrapposizione degli stati

```
circuit.h(a)
circuit.h(m)
```

Codifica del vettore d'input

```
circuit.cry(xo, a[o], i[o])
circuit.x(a) # entanglement dell'ancilla con o
```

ALGORITMO PER CLASSIFICATORE BINARIO

Codifica dei vettori di training

```
circuit.mcry(to, a[:] + m[:], i[o], None)
circuit.x(m) # entanglement di m con o
circuit.mcry(t1, a[:] + m[:], i[o], None)
circuit.cx(m, c) # entanglement della classe 1 con m 1
```

Interferenza degli stati

```
circuit.h(a)
```

Operazione di misura

```
circuit.measure(a, b[o])
circuit.measure(c, b[1])
```

```
# circuit.draw(output='mpl')
```

ALGORITMO PER CLASSIFICATORE MULTICLASSE

Inizializzazione dei registri quantistici

```
a = QuantumRegister(1, 'a') # knn ancilla  
m = QuantumRegister(2, 'm') # training vector index  
i = QuantumRegister(2, 'i') # feature index  
r = QuantumRegister(1, 'r') # rotation qubit  
q = QuantumRegister(5, 'q') # qram ancilla  
c = QuantumRegister(2, 'c') # class  
b = ClassicalRegister(4, 'bit')  
circuit = QuantumCircuit(a, m, i, r, q, c, b)
```

Sovrapposizione degli stati

```
circuit.h(a)  
circuit.h(m)  
circuit.h(i)  
circuit.h(c)
```

ALGORITMO PER CLASSIFICATORE MULTICLASSE

Codifica dei vettori

```
# circuit.cry(theta, control, target)
# circuit.mcry(theta, controls, target, ancillae)
# » Encode the input vector »
encodeVector(circuit, inputVirginica, i, a[:] + i[:, r[o], q])
circuit.x(a) # entanglement dell'ancilla con o
# » Encode the training vectors »
buildTrainingState(trainingArray)
```

Interferenza e misura

```
circuit.measure(r, b[o])
circuit.h(a)
circuit.measure(a, b[1])
circuit.measure(c[o], b[2])
circuit.measure(c[1], b[3])
```

ALGORITMO PER CLASSIFICATORE MULTICLASSE

Definizione dei costruttori

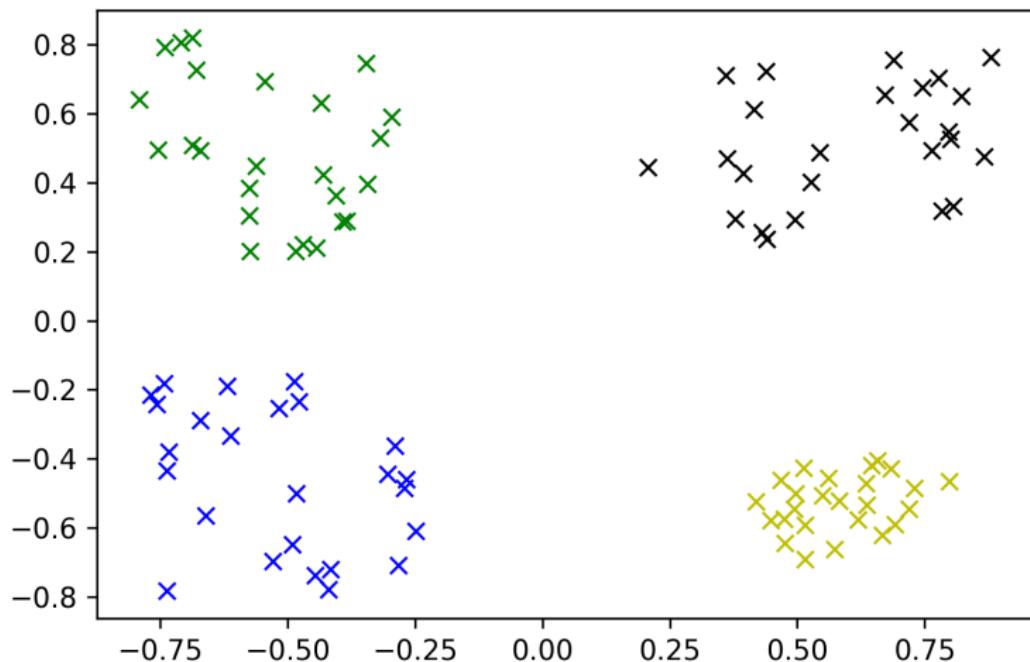
```
def encodeTraining(circuit, data, i, controls, rotationQ, ancillaQ,  
c, m):  
    # Header  
    encodeClass(circuit, c)  
    encodeIndex(circuit,m)  
    # Encoder  
    encodeVector(circuit, data, i, controls, rotationQ, ancillaQ)  
    # Footer  
    encodeClass(circuit, c)  
    encodeIndex(circuit, m)
```

ALGORITMO PER CLASSIFICATORE MULTICLASSE

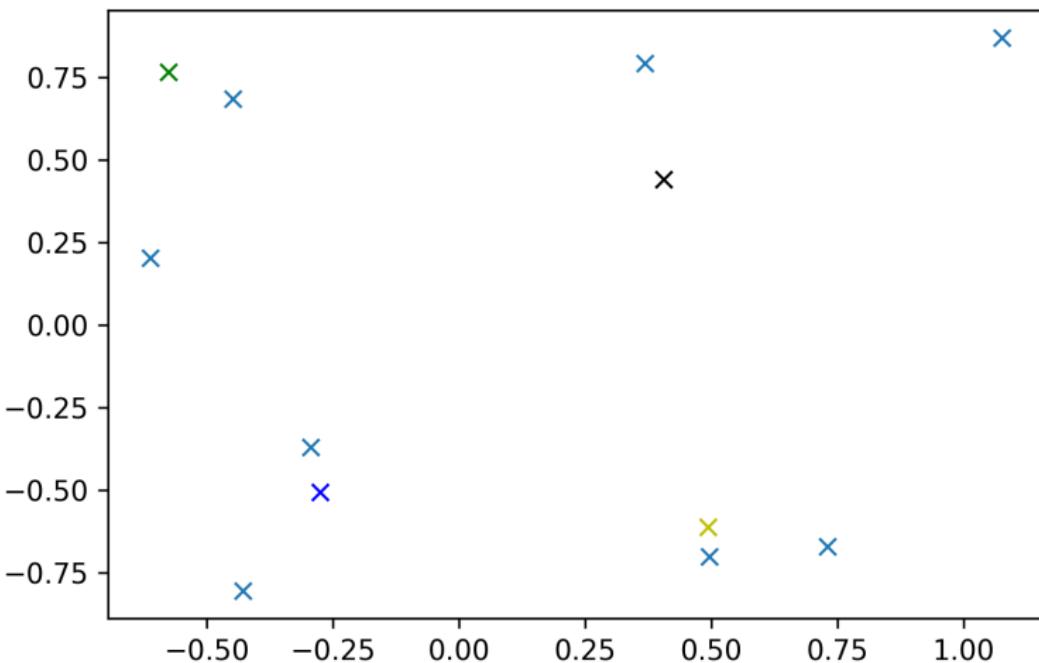
Definizione del codificatore

```
def encodeVector(circ, data, i, controls, rotationQ, ancillaQ):
# |00>
    circuit.x(i)
    circuit.mcry(data[0], controls, rotationQ, ancillaQ)
    circuit.x(i)
# |01>
    circuit.x(i[1])
    circuit.mcry(data[1], controls, rotationQ, ancillaQ)
    circuit.x(i[1])
# |10>
    circuit.x(i[0])
    circuit.mcry(data[2], controls, rotationQ, ancillaQ)
    circuit.x(i[0])
# |11>
    circuit.mcry(data[3], controls, rotationQ, ancillaQ)
```

CLASSIFICAZIONE DI CLUSTER



CLASSIFICAZIONE DI CLUSTER



CLASSIFICAZIONE DI CLUSTER

