Quantum Machine Learning

Quantum Communications and Computing - Final report

Boldini Davide, 0001075697
davide.boldini@studio.unibo.it

Lumini Enrico, 0001087025
enrico.lumini@studio.unibo.it

Giugno 2024

Indice

Lista delle figure

1 Introduzione

Con lo sviluppo di questo progetto si richiede l'analisi e l'implementazione di due algoritmi cardine del machine learning quantistico: qPCA (Quantum Principal Component Analysis) e HHL (Harrow–Hassidim–Lloyd). Per l'implementazione di entrambi i circuiti è stato utilizzato il framework Qiskit. Tutti i circuiti progettati sono stati testati in locale utilizzando il simulatore fornito dal framework, ossia *qasm_simulator*.

2 qPCA

Il Quantum Principal Component Analysis (qPCA) è una tecnica che utilizza principi della meccanica quantistica per eseguire l'Analisi delle Componenti Principali (PCA). La PCA è un metodo statistico utilizzato per ridurre la dimensionalità di un dataset, mantenendo il più possibile la varianza presente nei dati originali. Questo processo si ottiene trasformando i dati in un nuovo sistema di coordinate, dove le nuove variabili (chiamate componenti principali) sono ortogonali tra loro e ordinate per importanza in termini di varianza.

2.1 Implementazione

Come illustrato nella Figure 1, l'algoritmo è schematicamente suddiviso in quattro fasi: (1) pre-elaborazione classica, (2) preparazione dello stato, (3) quantificazione della purezza e (4) post-elaborazione classica.

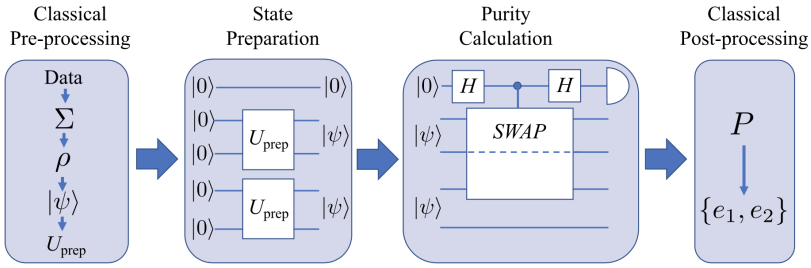


Figure 1: Diagramma schematico dell'algoritmo quantistico per PCA, nel caso speciale di sole due features

Nel primo passaggio, dai vettori di dati grezzi viene calcolata la matrice di covarianza Σ, quest'ultima viene poi normalizzata per formare $\rho = \frac{\Sigma}{Tr(\Sigma)}$ e successivamente purificata per ottenere uno stato puro $|\psi\rangle$. Infine viene calcolato l'operatore unitario U_{prep}, necessario per preparare $|\psi\rangle$ a partire da una coppia di qubit inizialmente entrambi nello stato $|0\rangle$. Lo stato puro $|\psi\rangle$ utilizzato è il seguente:

$$|\psi\rangle = \sqrt{e_1}|\phi_1\rangle \otimes |0\rangle + \sqrt{e_2}|\phi_2\rangle \otimes |1\rangle,$$

considerando che e_1 e e_2 rappresentano gli autovalori mentre ϕ_1 e ϕ_2 i corrispettivi autovettori della matrice di covarianza normalizzata ρ.
Infine l'operatore U_{prep} è stato inizialmente descritto dalla seguente matrice:

$$U = \begin{pmatrix} \sqrt{e_1} * \phi_{11} & 0 & 0 & 0 \\ \sqrt{e_2} * \phi_{21} & 1 & 0 & 0 \\ \sqrt{e_1} * \phi_{12} & 0 & 1 & 0 \\ \sqrt{e_2} * \phi_{22} & 0 & 0 & 1 \end{pmatrix} \quad con \quad |\phi_1\rangle = \begin{pmatrix} \phi_{11} \\ \phi_{12} \end{pmatrix}, \quad |\phi_2\rangle = \begin{pmatrix} \phi_{21} \\ \phi_{22} \end{pmatrix},$$

in modo da generare lo stato puro $|\psi\rangle$ precedentemente descritto partendo da una coppia di qubit inizialmente entrambi nello stato $|0\rangle$. Siccome l'operatore U così definito non risulta unitario abbiamo applicato l'algoritmo di Gram-Schmidt al fine di renderlo tale.

Per implementare il secondo passaggio, abbiamo creato un circuito composto da cinque qubit etichettati, partendo dal meno significativo, come segue: ancilla, A, B, A' e B'. Abbiamo poi utilizziamo il gate U_{prep} due volte per inizializzare lo stato $|\psi\rangle$ rispettivamente sui qubit AB e A'B'.

Nel terzo passaggio abbiamo utilizzato l'ancilla per implementare l'algoritmo che determina la purezza $P := Tr(\rho^2)$ di ρ. Quest'ultimo prevede l'utilizzo di due gate hadamard sul qubit di ancilla intervallati da un gate di SWAP controllato dall'ancilla stessa con target i qubit A e A'. Abbiamo infine aggiunto una misura sul qubit di ancilla.

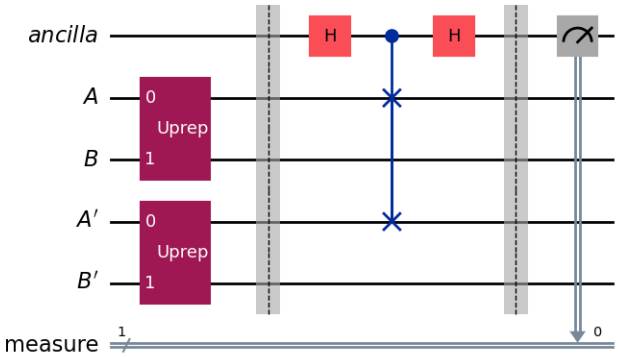


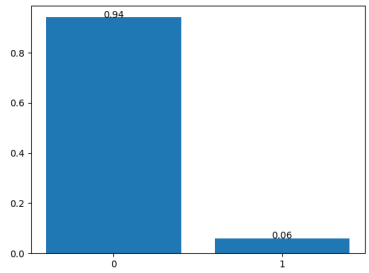
Figure 2: Circuito dell'algoritmo quantistico per PCA implementato in Qiskit

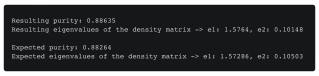
A seguito dell'esecuzione del circuito e della misura sull'ancilla abbiamo determinato il valore di P utilizzando la relazione $p_0 - p_1 = Tr(\rho^2) = P$, dove p_0 e p_1 sono rispettivamente le probabilità che l'ancilla risulti 0 e 1. Infine abbiamo calcolato i due autovalori utilizzando le seguenti equazioni:

$$e_1 = Tr(\Sigma) * \frac{1 + \sqrt{1 - 2(1 - P)}}{2}, \quad e_2 = Tr(\Sigma) * \frac{1 - \sqrt{1 - 2(1 - P)}}{2}$$

2.2 Risultati

Abbiamo testato l'algoritmo utilizzando la seguente $\Sigma = \begin{pmatrix} 0.38 & 0.57 \\ 0.57 & 1.30 \end{pmatrix}$, con relativi autovalori $e_1 = 1.57$ e $e_2 = 0.10$ ed abbiamo ottenuto i seguenti risultati:





(a) Output della misura del qubit di ancilla

(b) Confronto tra P e autovalori attesi e calcolati

Figure 3: Risultati del test qPCA

Come si può osservare gli autovalori e la purezza ottenuti con all'algoritmo quantistico sono molto simili a quelli attesi, nello specifico si hanno i seguenti errori quadratici medi:

$$MSE_P = 1.38 \times 10^{-5}, \quad MSE_{e_1} = 1.25 \times 10^{-5}, \quad MSE_{e_2} = 1.26 \times 10^{-5}$$

3 HHL

L'algoritmo HHL (Harrow-Hassidim-Lloyd) è un algoritmo quantistico sviluppato nel 2009 da Aram Harrow, Avinatan Hassidim e Seth Lloyd. È progettato per risolvere sistemi lineari di equazioni della forma $Ax = b$, dove A è una matrice quadrata, x è il vettore incognito da determinare e b è un vettore noto. L'algoritmo HHL è significativo perché può, in teoria, risolvere sistemi di equazioni lineari con un guadagno esponenziale rispetto ai migliori algoritmi classici noti. Tuttavia, la sua implementazione pratica richiede un numero elevato di qubit e operazioni quantistiche precise, rendendolo attualmente una sfida per la tecnologia quantistica contemporanea. L'implementazione che segue è specifica per sistemi 2x2.

3.1 Implementazione

Come illustrato nella Figure 4, l'algoritmo è suddiviso in 5 componenti principali: (1) preparazione dello stato, (2) quantum phase estimation (QPE), (3) ancilla quantum encoding (AQE), (4) inverse quantum phase estimation (IQPE) e (5) misurazione.

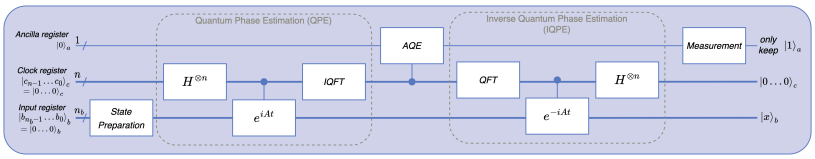


Figure 4: Schema dell'algoritmo HHL

Nella preparazione dello stato, l'input register $|0\dots0\rangle_b$ deve essere ruotato affinché le ampiezze corrispondano ai coefficienti di $\vec{b}$. Nel nostro caso avremo $\vec{b} = \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}$ pertanto l'input register sarà formato da un singolo qubit, $n_b = 1$, nello stato $|b\rangle = \beta_0|0\rangle + \beta_1|1\rangle$.

A seguito della QPE, gli autovalori della matrice A verranno codificati nel clock register, tramite la codifica di base. Pertanto il clock register non memorizza gli autovalori esatti ma una loro approssimazione. Al fine di effettuare tale codifica gli autovalori devono essere interi, non essendo questo sempre possibile risulta necessario determinare un valore t tale che $\tilde{\lambda_j} = \frac{N\lambda_j t}{2\pi}$ risulti intero, dove $\tilde{\lambda_j}$ rappresenta l'autovalore codificato nel clock register, λ_j l'autovalore non intero e $N = 2^n$ con n il numero di qubit del clock register. Inoltre per effettuare la codifica in maniera corretta il rapporto tra $\frac{\lambda_0}{\lambda_1}$ e $\frac{\tilde{\lambda_0}}{\tilde{\lambda_1}}$ deve essere preservato.

Siccome per risolvere il sistema lineare abbiamo bisogno della matrice inversa A, nella prima fase della AQE, calcoliamo i reciproci degli autovalori, in quanto l'inversa della matrice $A = \sum_j \lambda_i |u_j\rangle\langle u_j|$ è esprimibile come $A^{-1} = \sum_j \frac{1}{\lambda_j}|u_j\rangle\langle u_j|$. Per valutare i reciproci, senza misurare i qubit, abbiamo effettuato uno swap tra due coppie di qubit del clock register dipendenti dalla dimensione di quest'ultimo. Ad esempio, nel caso di n=4, lo swap viene effettuato tra il bit 1 e 3 del clock register.

A questo punto ruotiamo il qubit ancilla, $|0\rangle_a$, in base ai reciproci degli autovalori valutati al punto precedente in modo da codificarli nelle ampiezze di quest'ultima. Quando l'ancilla viene misurata collassa nello stato $|0\rangle$ o $|1\rangle$, il risultato viene considerato solo nel caso in cui si misuri $|1\rangle$.

Al fine di mantenere l'integrità e l'accuratezza del calcolo finale viene eseguita la fase di uncomputation, che comprende le inverse degli step precedenti, escludendo la rotazione del qubit ancilla. Pertanto lo stato finale risulterà nella forma $|x\rangle_b|0\rangle_c^{\otimes n}|1\rangle_a$.

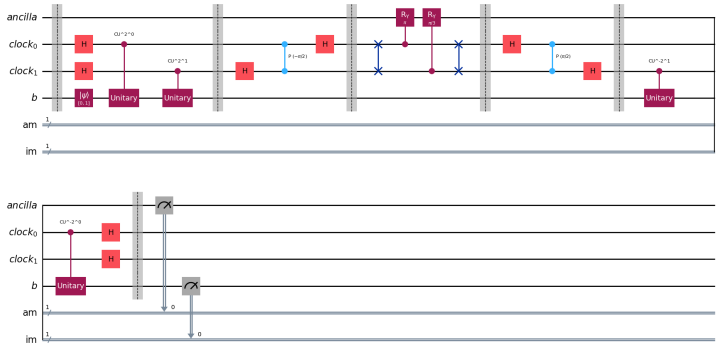


Figure 5: Circuito dell'algoritmo HHL implementato in Qiskit

3.2 Risultati

Abbiamo verificato il circuito utilizzando la seguente matrice $A = \frac{1}{2}\begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$ e il seguente vettore noto $b = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Siccome non possiamo accedere direttamente allo stato finale abbiamo verificato la veridicità del risultato valutando che il rapporto tra i risultati calcolati in maniera classica risulti comparabile a quello delle probabilità ottenute dall'algoritmo quantistico implementato.

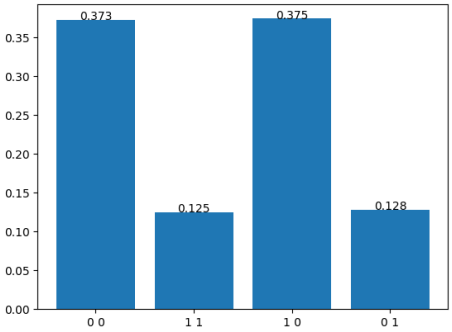


Figure 6: Risultato della misura sul circuito implementato

Risolvendo in maniera classica il sistema sopra descritto si ottiene $x_1 = x_2$ pertanto il rapporto risulta pari a 1. Come si può osservare il rapporto delle probabilità risultate dalla misura sul circuito quantistico, notando che il bit ancilla è il meno significativo, risulta $\frac{0.128}{0.125} = 1.024$. Come si può osservare i due valori risultano molto simili, con un MSE pari a 5.3×10^{-4}.