

UNIVERSITÀ DI TRIESTE, FISICA
MECCANICA QUANTISTICA AVANZATA

**Identificazione della transizione di fase
nel modello di Ising, tramite SVM**

Enrico Frausin

Mafalda Dal Cin

Enrico Di Salvo

DOCENTI DI RIFERIMENTO: Angelo Bassi, Michele Grossi

Febbraio 2024

Sommario

Nella seguente relazione, viene esposto l'utilizzo di un modello di machine learning classico (Support Vector Machine, SVM) implementato a supporto dell'analisi di una transizione di fase di un sistema quantistico (catena di Ising in campo trasverso). In particolare, la SVM distingue le fasi del sistema valutando in input i valori dei parametri di un circuito variazionale, che rappresenta il ground state del sistema quantistico. Per un apprendimento efficace della SVM, è necessario manipolare i parametri con una cosiddetta funzione kernel. Questa rielabora i dati in uno spazio vettoriale di dimensione maggiore rispetto a quello di partenza, permettendo una separazione lineare delle classi di dati. Come funzioni di kernel sono state considerate tre alternative principali: la Radial Basis Function (RBF), la fidelity e la fidelity per sito. La prima è di carattere classico ed è l'alternativa più comunemente utilizzata nell'analisi dati con una SVM. Le seconde sono funzioni definibili solo sugli stati quantistici. Di conseguenza, per il loro calcolo necessitano di un computer quantistico o di un simulatore.

L'impiego del kernel classico garantisce dei buoni risultati solamente per determinati dataset; la fidelity si dimostra la migliore da impiegare nel caso di interesse, mentre la fidelity per sito si adatta poco bene all'utilizzo su catene di spin corte.

Indice

1	Il modello di Ising e la transizione di fase	2
2	Elaborazione del dataset, VQE e groundstates	3
3	Support Vector Machine	4
3.1	Kernel Classico	5
3.2	Fidelity tradizionale	7
3.3	Fidelity per sito	9
4	Scalabilità	10
5	Simulazione del rumore	11
6	Conclusioni	12

1 Il modello di Ising e la transizione di fase

Nel progetto svolto, è stato considerato il modello di Ising con campo trasverso come sistema quantistico di riferimento. Tale modello è caratterizzato da una catena di N spin con periodic boundary conditions (PBC) soggetta all'azione dell'Hamiltoniana

$$H(J) = \sum_{j=1}^N \sigma_j^z - J \sum_{j=1}^N \sigma_j^x \sigma_{j+1}^x \quad (1)$$

Le σ_j^α sono le matrici di Pauli che agiscono sul sito j -esimo. Tale Hamiltoniana descrive un'interazione tra primi vicini nella catena di spin orientata lungo l'asse x , esplicitata dalle matrici di Pauli σ_j^x , accoppiata con un campo esterno trasverso lungo l'asse z , rappresentato dalle matrici σ_j^z . Il modello in esame presenta una transizione di fase del second'ordine per $J_c = 1(-1)$ nel limite termodinamico di $N \rightarrow \infty$. Per $J_c > 1 (J_c < 1)$ i ground states sono ordinati ferromagneticamente (antiferromagneticamente). La transizione di fase quantistica sulla quale ci siamo concentrati è quella che avviene per $J_c = 1$. Essendo le proiezioni dello spin lungo l'asse z e l'asse x due operatori che non commutano, non è possibile osservare tali quantità contemporaneamente, privando il sistema di una controparte classica.

Introduciamo ora due quantità utili nello svolgimento del progetto: la fidelity F e la fidelity per sito λ . Dati due ground states $|\psi_0(J)\rangle$ dell'Hamiltoniana per diversi J , la **fidelity** è definita come

$$F(J, J') := |\langle \psi_0(J) | \psi_0(J') \rangle| \quad (2)$$

Per N sufficientemente grandi, si presenta la cosiddetta *orthogonality catastrophe*: due ground states con $J \neq J'$ hanno una fidelity esponenzialmente piccola, indipendentemente dal fatto che appartengano o meno alla stessa fase. Questo può comportare dei problemi nell'elaborazione dei dati che verrà esposta in seguito. Si definisce quindi la **fidelity per site** λ da

$$\log[\lambda(J, J')] := \log F(J, J')/N. \quad (3)$$

Nel caso di un reticolo invariante per traslazioni con dimensione locale l , il ground state dell'Hamiltoniana può essere scritto nella sua forma *Matrix Product State (MPS)*, $|\psi_0\rangle = \text{Tr}[A_{i_1} \dots A_{i_N}] |i_1, \dots, i_N\rangle$. Qua le $\{A_{i_j}\}$ sono matrici $D \times D$ che dipendono dal numero locale $i_j = 1, \dots, l$. D è la dimensione di legame, che è legata alla quantità di entanglement dello stato. Con questa forma, la fidelity può essere riscritta come

$$F(J, J') = \sum_{k=1}^{D^2} \lambda_k(J, J')^N \approx \lambda_1(J, J')^N, \quad \text{per } N \gg 1. \quad (4)$$

I λ_k sono gli autovalori della matrice di trasferimento $E(J, J') = \sum_{i=1}^l A_i(J) \otimes A_i(J')$. Gli stati sono normalizzati quindi $|\lambda_k| < 1$. λ_1 indica l'autovalore maggiore in valore assoluto. Esprimendo la fidelity in questa forma è evidente l'*orthogonality catastrophe* e si può notare come, nel limite termodinamico, la fidelity per sito sia una quantità intensiva che non dipende dalla lunghezza della catena di spins.

2 Elaborazione del dataset, VQE e groundstates

L'insieme di dati necessario all'apprendimento della SVM, è costituito da una serie di parametri per ogni J campionario. Questi corrispondono ai parametri da legare ad un circuito quantistico, reso nello specifico dall'ansatz **RealAmplitudes** fornito da Qiskit, in modo che contenga l'informazione del groundstate relativo ad $H(J)$ (1). Idealmente:

$$|\psi(J)\rangle = U(J)(|0\rangle_1 \otimes |0\rangle_2 \otimes \dots \otimes |0\rangle_N)$$

Dove $U(J)$ rappresenta l'effetto del circuito sul vettore di zeri, N il numero di qubit, $|\psi(J)\rangle$ il groundstate di $H(J)$.

Nella pratica l'azione del circuito su $|0\rangle$ restituirà un'approssimazione del groundstate di interesse.

La ricerca dei parametri da legare all'ansatz in modo da ottenere $U(J)$ avviene per mezzo di un algoritmo variazionale di ricerca di autostati. La **VQE** fa uso di un backend, di un circuito stampo (l'ansatz) che dipende da certi parametri liberi, e di un ottimizzatore che contribuisce ad identificare i parametri ottimali che minimizzino la misura di $H(J)$ sul circuito.

Nel caso corrente, si è istanziata la VQE proveniente da `qiskit.algorithms.minimum_eigensolvers` con l'ottimizzatore **COBYLA**, **statevector-simulator** da `Aer` come backend e, come anticipato, **RealAmplitudes** come ansatz, impostato su tre ripetizioni di blocchi *rotazioni su Y + entanglement tramite CNOT* per un totale di 40 parametri.

Dataset generato: 600 J campionati uniformemente tra $0.25 \div 1.75$.

L'utilizzo di RealAmplitudes offre svariati vantaggi: l'utilizzo di parametri reali e la possibilità di ottenere funzioni d'onda reali in uscita dal circuito, facilita l'utilizzo di ottimizzatori classici ben rodati; una ridotta profondità, nonché l'impiego esclusivo di operatori di rotazione su Y e gates CNOT, rendono il circuito adattabile e compatibile con svaritati tipi di hardware.

Inoltre, sono stati eseguiti tentativi con l'ottimizzatore SPSA, ottenendo però risultati peggiori.

Oltre ad includere i 40 parametri, i dataset generati riportano il valore dell'energia $H(J)$ misurata sul circuito a fine VQE, e l'autovalore (ritenuto effettivo) estrappolato utilizzando l'eigensolver di numpy sull'hamiltoniana.

Impostando COBYLA per un massimo di 1000 steps, si ottengono i risultati di destra (Fig. 1). Per gran parte dei J , si osserva una perfetta sovrapposizione tra valore reale e quello campionario dalla misurazione.

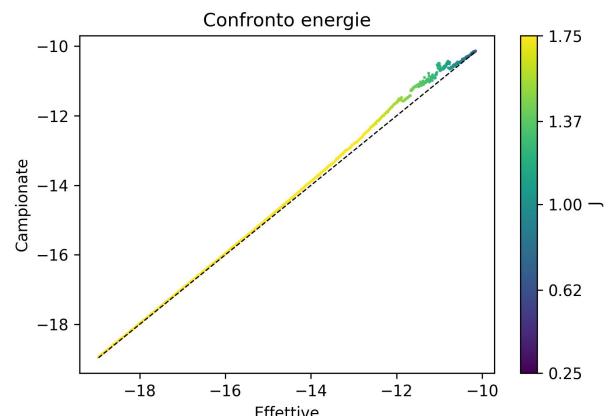


Figura 1: Confronto tra gli autovalori misurati e quelli effettivi

3 Support Vector Machine

Le Support Vector Machine (SVM) è un tipo di algoritmo di apprendimento automatico utilizzato sia per la classificazione che per la regressione. L'obiettivo principale di una SVM è trovare un iperpiano ottimale in uno spazio multidimensionale che separi i punti del dataset appartenenti a classi diverse, in modo tale da massimizzare il margine tra di essi.

Il tipo di separazione è perciò lineare. Per fare in modo che l'algoritmo possa trovare margini di ordine superiore, i dati in pasto alla SVM vengono prima manipolati; ovvero, la macchina si addestra su dei nuovi dati che sono funzione di quelli iniziali. Questa funzione è chiamata **Kernel**. Tramite il kernel si costruisce una matrice $\mathbf{K}_{i,j} = \mathbf{K}(\mathbf{J}_i, \mathbf{J}_j)$ da passare alla SVM. La separazione risulterà lineare nello spazio dei kernel, e non-lineare (in generale) nello spazio del dataset di origine.

Nel presente lavoro, valuteremo le prestazioni della SVM, al variare di tre diversi kernel:

- Kernel **Classico RBF**:

$$\mathbf{K}_{RBF}(\mathbf{J}_i, \mathbf{J}_j, \gamma) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||) \quad (5)$$

x_i, x_j coppia di array di parametri per i rispettivi J .

- **Fidelity**

$$\mathbf{K}^F = \mathbf{F}(\mathbf{J}_i, \mathbf{J}_j) \quad (6)$$

Si veda 2

- **Fidelity per sito**

$$\mathbf{K}^\lambda = \lambda(\mathbf{J}_i, \mathbf{J}_j) \quad (7)$$

Si veda 3

La SVM lavora variando i parametri α_j in maniera tale da minizzare la lagrangiana:

$$L(\beta) = \sum \alpha_j - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (8)$$

le y_i sono i label associati a ciascun J , dove $y = +1$ se $J > 1$, $y = -1$ altrimenti. Tali label classificano i dati in base all'appartenenza alla fase ferromagnetica o paramagnetica rispettivamente.

A classificazione avvenuta, è possibile risalire alla distanza $d(\mathbf{J})$ tra ciascuno dei punti nello spazio dei kernel e l'iperpiano di separazione.

$$d(\mathbf{J}) = \sum \alpha_j y_j K(\mathbf{J}, \mathbf{J}_j) + b \quad (9)$$

Sia i parametri α_j che il parametro di offset b sono forniti dall'algoritmo dopo l'apprendimento.

\mathbf{J}_C punto critico del sistema, corrisponde ai punti sull'iperpiano di separazione, ed è quindi tale da rendere: $d(\mathbf{J}_C) = 0$

3.1 Kernel Classico

Si riportano di seguito i risultati ottenuti impiegando un Kernel Classico (Sec. 3). Eq. 5 non fa uso esplicito dell'autostato, cioè non richiede l'utilizzo di un circuito quantistico per il calcolo del Kernel (per questa ragione viene chiamato "classico"). L'informazione riguardo gli autostati è contenuta implicitamente negli array di 40 parametri, provenienti dalla VQE (Sec. 2), su cui si valutano le distanze.

La procedura è stata ripetuta per due diversi intervalli di train: 100 J campionati su tutto $\delta_0 = [0.25, 1.75]$; 80 J campionati su $\delta_1 = [0.8, 0.9] \cup [1.6, 1.7]$.

Nota: l'utilizzo della RBF manifesta l'idea di pesare le distanze tra i parametri secondo una distribuzione normale; kernel diversi (tipo polinomiali) hanno mostrato risultati peggiori.

Intervallo δ_0

Vengono mostrati di seguito kernel e distanze per diversi valori di γ .

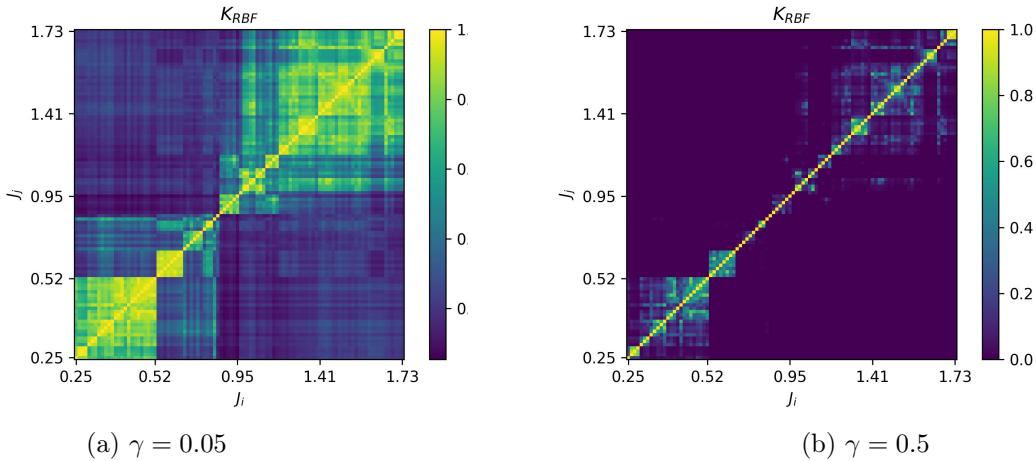


Figura 2: Matrice del kernel classico per $\gamma = 0.05$ a sinistra, ed $\gamma = 0.5$ a destra. Aumentare γ vuol dire rendere il valore di K in eq. 5 più piccolo. γ maggiori bruciano una parte dell'informazione della matrice.

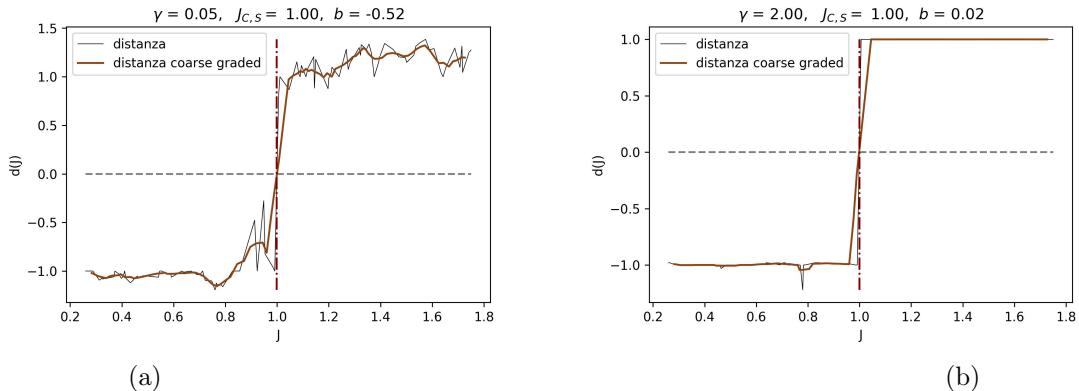


Figura 3: Calcolo di $d(J)$ per $\gamma = 0.05$ e $\gamma = 2$. Il punto critico trovato $\mathbf{J}_C = 1.00$, ottenuto dall'intersazione tra la $d(J) = 0$ e la funzione distanza resa più liscia tramite coarse graining (perciò $J_{C,S}$ nel titolo), non cambia.

Come si nota dalle figure precedenti, per quanto la scelta di una γ maggiore rovini l'informazione della matrice di Kernel la soglia critica ricavata non ne risulta compromessa, e si mostra pienamente compatibile con quanto atteso. Tuttavia, notiamo che la distanza, nel caso di $\gamma = 2$ più alto, mostra uno stacco più netto tra le due fasi. È ipotizzabile che riducendo l'esponenziale in 5 ne segua un abbattimento del ruolo di K nel determinare la distanza 9. L'andamento provoca un'identificazione di J ottimale per via esclusivamente degli y_i , e del fatto che l'intervallo (e quindi i label) esplorino tutto δ_0 , compresa la zona critica. Ci aspettiamo comportamenti diversi variando δ .

Intervallo δ_1

Rendendo l'intervallo asimettrico rispetto a $J_C = 1.0$ aspettato, la SVM con kernel classico comincia a mostrarsi inadeguata.

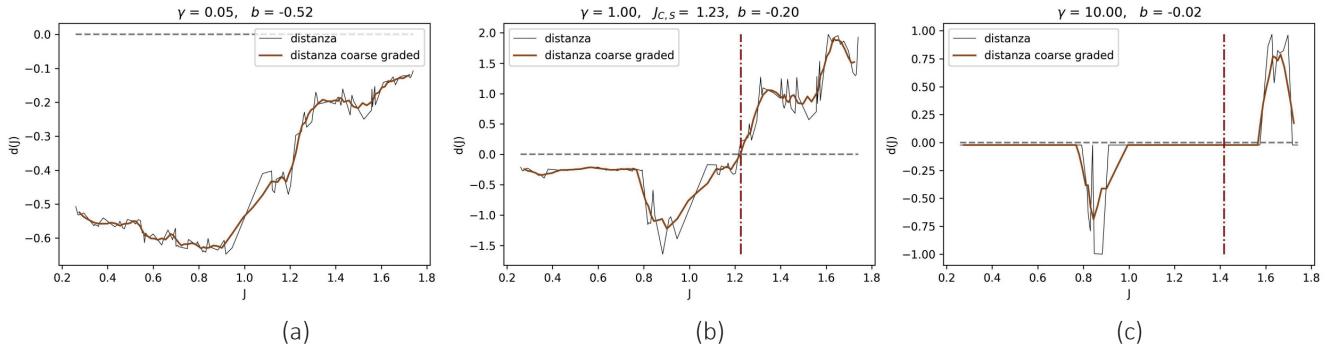


Figura 4: Caption

Figura 5: Calcolo di $d(J)$ per $\gamma = 0.05$ (a), $\gamma = 1$ (b) e $\gamma = 10$ (c), nel caso di intervallo δ_1 asimmetrico rispetto al punto critico aspettato. Nella figura di sinistra, la distanza si mantiene negativa e J_C non viene trovato. In quella di centrale si individua $J_C = 1.23$. In quella di destra, nuovamente non viene trovato J_C .

In questo caso, valori bassi ($\gamma = 0.05$) non portano risultati. L'asimmetria del kernel rende difficile alla SVM delineare un piano di separazione coerente coi label, mentre $\gamma = 1$ rende manifesta l'azione degli y_i che non basta tuttavia a far assestare J_C su 1.0. γ ancora maggiori comportano un appiattimento del modello.

3.2 Fidelity tradizionale

Per quanto riguarda il kernel relativo alla fidelity, abbiamo considerato due tipi di fidelity diversi. Abbiamo definito prima una funzione "fidelity_SV(x1, x2)": x1 e x2 (array di parametri) vengono legati a due circuiti, eseguiti successivamente tramite *StateVector-Simulator*; dei vettori di stato risultanti, viene calcolata la sovrapposizione.

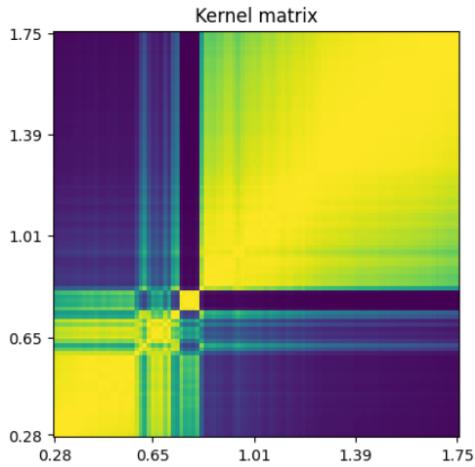
Successivamente, abbiamo considerato anche una funzione "fidelity(x1, x2, shots)": lega x1, x2 ai circuiti; circuito(x2) viene trasposto e connesso al primo circuito(x1); il circuito risultante viene eseguito su *QasmSimulator* per un numero di shots generalmente pari a 1000; la sovrapposizione si calcola infine come la radice della frequenza di outcome $|0\rangle$.

Idealmente, i risultati provenienti da "fidelity_SV" rappresentano il limite di quelli della seconda funzione per un elevato numero di shots.

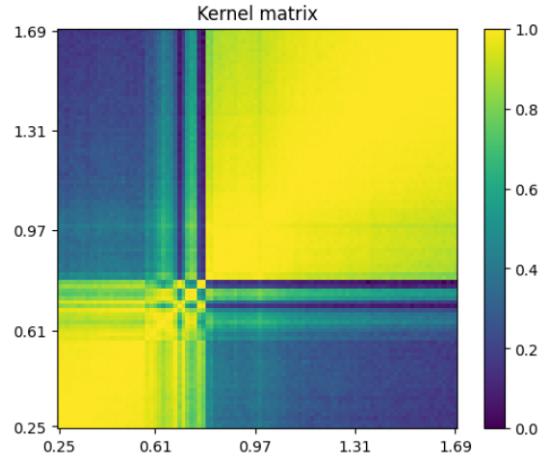
Come nel caso del kernel classico (3.1), abbiamo testato le fidelity negli stessi due diversi intervalli di train indicati nella sezione precedente, ovvero 100 J campionati su tutto $\delta_0 = [0.25, 1.75]$ e 80 J campionati su $\delta_1 = [0.8, 0.9] \cup [1.6, 1.7]$.

Intervallo δ_0

Nell'intervallo più ampio, che include il valore critico atteso, le matrici del kernel rispetto alla "fidelity_SV" e alla "fidelity" sono le seguenti:



(a) "fidelity_SV"

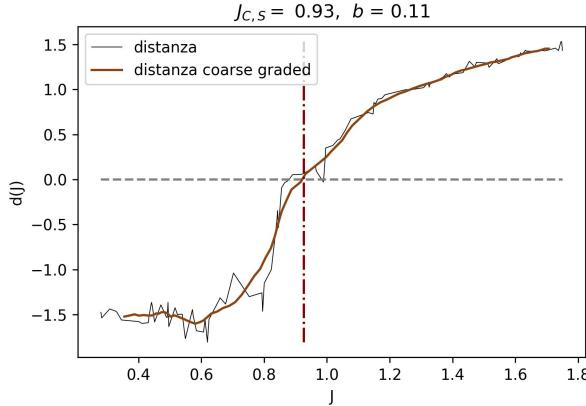


(b) "fidelity" con 1000 shots

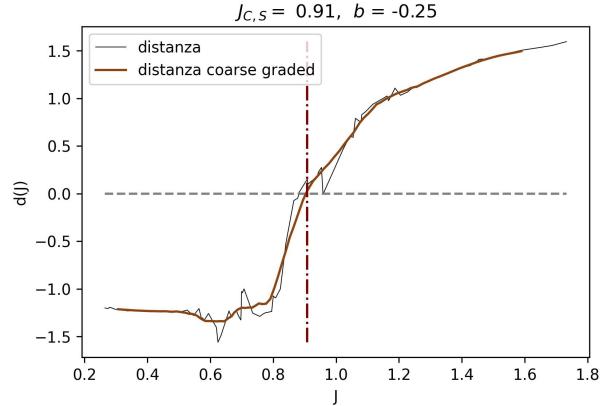
Figura 6: Possiamo notare che le differenze tra le due matrici 6a e 6b sono minime, infatti risulta poco più definita la separazione nella prima, e tali differenze non condizionano significativamente il risultato.

Inoltre le matrici risultano essere leggermente asimmetriche rispetto al valore critico $J_C = 1.0$, ma ciò non comporta grandi conseguenze nei risultati successivi riguardanti la distanza, in quanto mitigato dalla presenza dei coefficienti y_j .

Se ora consideriamo le distanze corrispondenti, possiamo ottenere i grafici seguenti (Fig. 7). Analizzando tali curve, si può notare come i relativi J_C vengono trovati nei punti 0.91 e 0.93. Si può ipotizzare che tali risultati siano dovuti alle dimensioni del sistema, e che gli effetti di taglia tendano ad abbassare il valore di soglia di J_c .



(a) La "fidelity_SV" identifica $J_C = 0.93$.

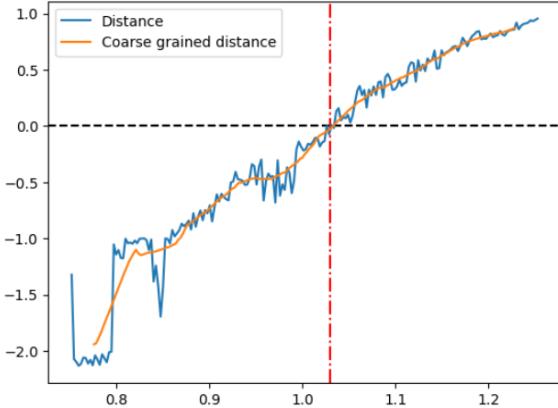


(b) La "fidelity" con 1000 shots identifica $J_C = 0.91$.

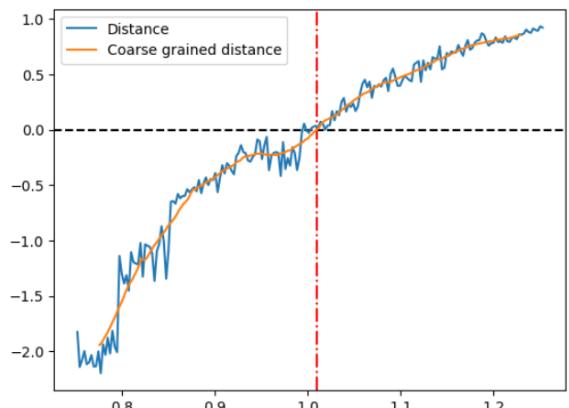
Figura 7: Valori della distanza $d(J)$.

Intervallo δ_1

Consideriamo ora il train nell'intervallo $\delta_1 = [0.8, 0.9] \cup [1.6, 1.7]$, scelto appositamente asimmetrico rispetto al punto critico atteso $J_C = 1.0$. Le distanze ottenute sono illustrate nei seguenti grafici:



(a) La "fidelity_SV" identifica $J_C = 1.02$ e $b = -0.429$.



(b) La "fidelity" con 1000 shots identifica $J_C = 1.01$ e $b = 0.067$.

In questo caso, invece, si può notare una differenza rispetto all'uso del kernel classico: i punti critici sono identificati in corrispondenza di 1.02 e 1.01 rispettivamente. La vicinanza ad uno della J_c trovata però non deve trarre in inganno. Se l'ipotesi fatta precedentemente è vera, ossia che gli effetti di taglia abbassino J_c , l'utilizzo di un intervallo asimmetrico falsa il valore soglia trovato, alzandolo sotto queste circostanze.

Se incrementassimo N si dovrebbe osservare il fenomeno dell'"orthogonality catastrophe", che compromette i risultati ottenuti sui grafici delle $d(J)$.

Nel nostro caso non è stato possibile sviluppare tale casistica e visualizzarne quindi le conseguenze per limiti dovuti alla potenza computazionale, ma nonostante ciò, abbiamo sviluppato e implementato le fidelity per sito, come illustrato nella prossima sezione (3.3).

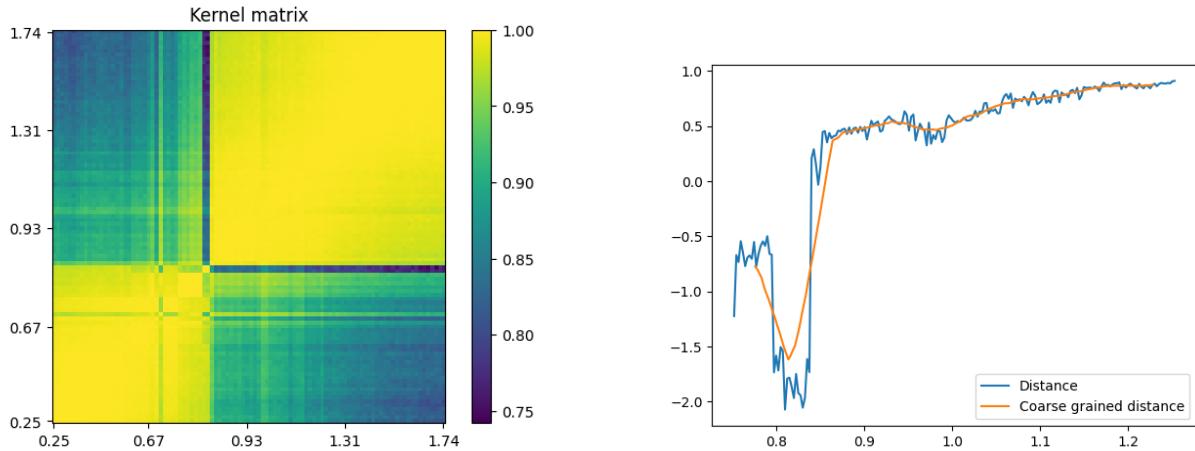
3.3 Fidelity per sito

L'esistenza dell' "orthogonality catastrophe" implica la necessità di sviluppare un kernel quantistico diverso dalla fidelity. Come indicato in 1, una buona alternativa è rappresentata dalla fidelity per site. Rimane però il problema di come implementare tale quantità in un algoritmo. Una prima opzione sarebbe quella di considerare la radice N-esima di ogni elemento $K_{ij}^\lambda = [K_{ij}^F]^{1/N}$. Tale metodo, però, non funziona. Difatti, la fidelity prossima allo zero per N sufficientemente grandi (N>1000 circa) rende difficile il calcolo di K_{ij}^F stesso.

La scelta effettuata è stata quella di inferire tramite un fit la probabilità $p_{0^n} = K_{ij}^F(n)$ di ottenere la stringa completa di zeri dal circuito per N, nel seguente modo:

- si è inizializzato n tale da $1 < n < N$.
- Per 4 iterazioni, nel caso di 10 spin, si è calcolata la fidelity $K_{ij}^F(n)$ come in 3.2, aumentando ciascuna volta n di un'unità, $n = n + 1$.
- Tramite una funzione di fit esponenziale, si è inferito il valore di $K_{ij}^F(N)$ per N.
- A questo punto, si è calcolata la radice N-esima del valore ottenuto al punto precedente per ottenere la fidelity per sito, $K_{ij}^\lambda = [K_{ij}^F(N)]^{1/N}$.

Nelle simulazioni effettuate si è ben distanti dai valori di N per i quali accade l'"orthogonality catastrophe". Una sua versione è stata implementata comunque nel codice per verificarne la validità. Di seguito, vengono esposti i risultati di un'esecuzione dell'algoritmo per la catena da 10 spin, con 2000 iterazioni per ogni misura.



(a) Kernel calcolato con la fidelity per sito

(b) Distanza calcolata con la fidelity per sito

Figura 9: Kernel calcolato con la fidelity per sito (a) e la corrispettiva distanza calcolata con tale kernel (b). Il valore di J critico trovato è di $J_c = 0.85$, con un'accuratezza della SVM di 0.85 ed un parametro di offset di $b = 0.14$.

Si può osservare così che i risultati ottenuti sono confrontabili con quelli ottenuti con la fidelity classica. Il kernel risulta leggermente più rumoroso. Ciò potrebbe essere causato dalla struttura dell'algoritmo, che prevedere il passaggio aggiuntivo di fit.

4 Scalabilità

Gli stessi procedimenti seguiti in Sec. 3.2 sono stati ripetuti per un numero di qubit minore: $N' = 5$. Se ne traggono conclusioni circa la scalabilità del modello e, in modo particolare, come varia il calcolo della fidelity classica a seconda della taglia del sistema. Non è stato possibile, per ragioni di potenza computazionale, provare numeri maggiori di qubit.

Viene mostrato di seguito il confronto tra la matrice dei kernel, nel caso di fidelity standart, per N ed N' .

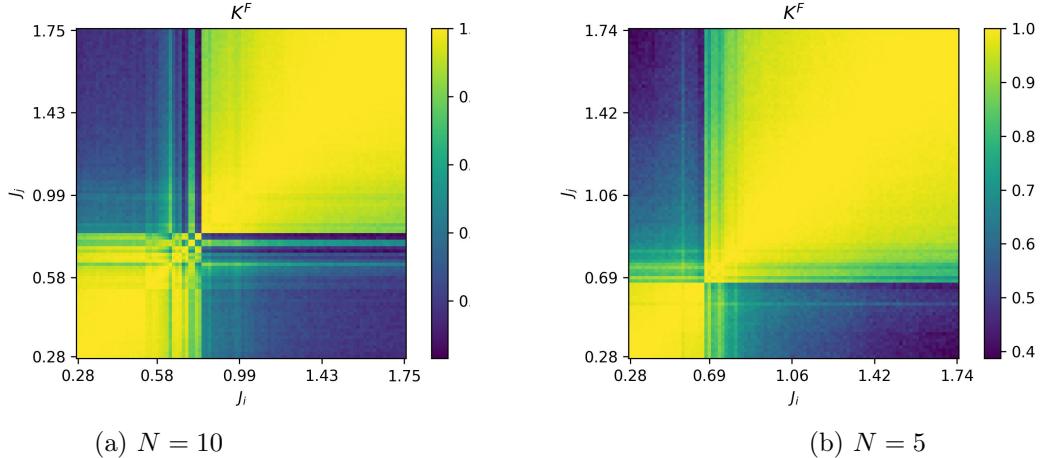


Figura 10: Confronto tra le matrici dei kernel $K = K^F$, nel caso di $N=10$ (a) e $N=5$ (b)

Notiamo come già la matrice per $N = 10$ si mostri più asimmetrica di quanto previsto in teoria. Infatti, la separazione tra le fasi dovrebbe centrarsi attorno a $J_{i,C} = J_{j,C} = 1.0$. Questo errore non grava troppo sulla performance della SVM, per via dei label y_j . A questo punto, si può affermare con più confidenza che lo spostamento della linea critica visibile sia dovuto ad effetti di taglia. Ci aspetteremmo che per N maggiori, la matrice riproduca più fedelmente la separazione teorica.

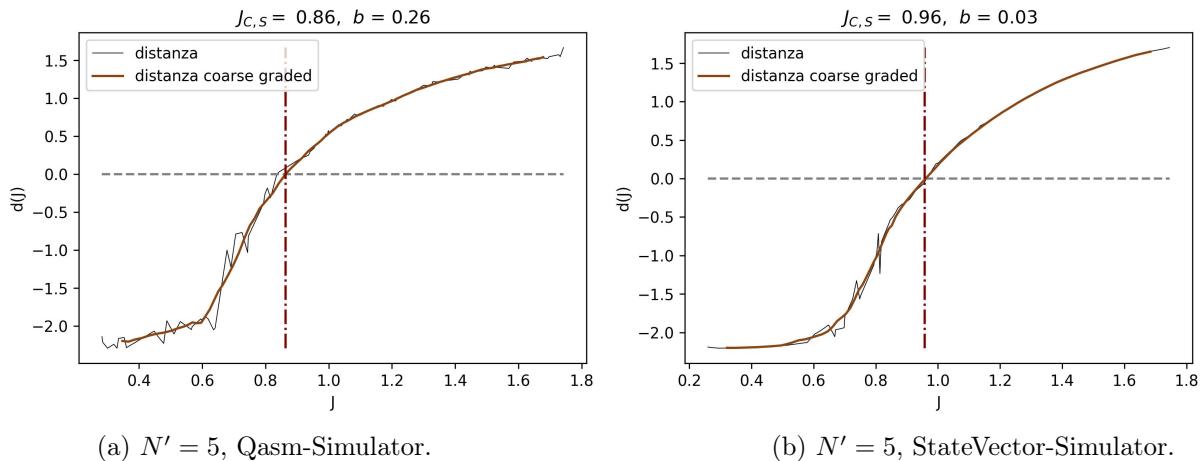


Figura 11: funzione distanza $d(J)$ calcolata nel caso di $N' = 5$ tramite Qasm-Simulator con 1000 shots (a) e StateVector-Simulator (b). Ci aspettiamo che (b) rappresenti il limite di (a) per un numero di shots più grande.

Il risultato limite di $J_C = 0.96$ si dimostra comunque ragionevole, per quanto leggermente più distante dal valore atteso di quanto non sia quello in sezione 3.2 per $N = 10$.

Riguardo all'impiego del kernel K^λ per diverse taglie, non ci è stato possibile provarlo per $N > 10$; per di più, applicarlo al caso di $N' = 5$ non risulterebbe di particolare interesse, visto come è definito (Sec. 1). Ci limitiamo a rimandare alle considerazioni teoriche già esposte riguardo la catastrofe di ortogonalità, in sezione 1.

5 Simulazione del rumore

Come ultima verifica si è implementato un Sampler, da usare in alternativa ai simulatore delle sezioni precedenti, che possa essere impostato in modo da prevedere del rumore generato sul circuito. Si è fatto uso del Sampler di default di Qiskit, unito al modello di rumore **depolarizing-error** da *qiskit.aer-noise*. Questo tipo di generatore di rumore introduce, con una certa probabilità p , degli effetti di decoerenza all'altezza dei gate CNOT presenti nel circuito.

Vengono mostrati di seguito i risultati ottenuti per $N' = 5$ qubits, al variare di p , $K = K^F$.

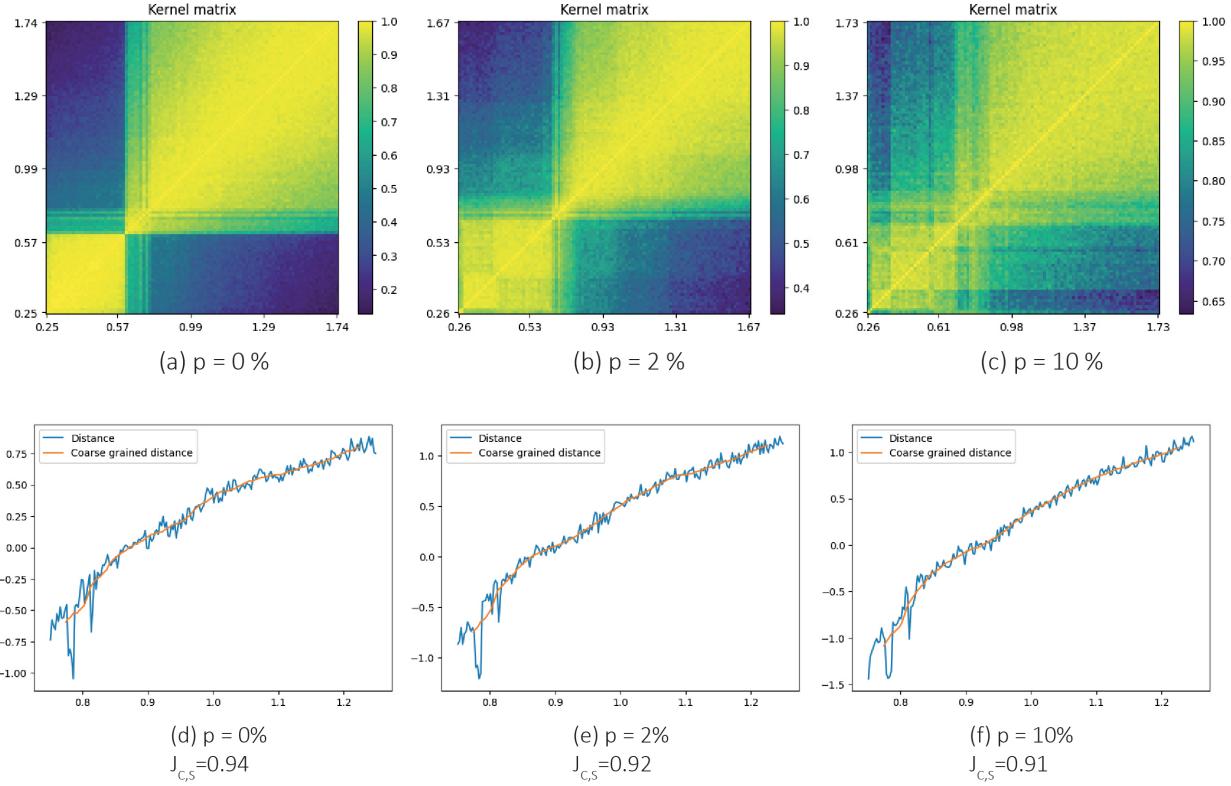


Figura 12: Effetto del rumore sia su K^F (a)-(b)-(c), che su $d(J)$ (d)-(e)-(f), al variare di p .

Come si vede di sopra, l'aggiunta di rumore si riflette sul kernel rendendo l'immagine più granulare e la separazione tra le fasi meno netta. In più, la distanza manifesta di meno la transizione di fase come appariva invece nelle sezioni precedenti (sotto forma di un salto della funzione). La stima di J_c varia all'aumentare del rumore. Si ottengono risultati comunque confrontabili tra di loro.

6 Conclusioni

I vantaggi legati all'utilizzo della fidelity per sito sarebbero evidenti in un modello con un numero elevato di qubit, anche in relazione alla definizione stessa dell'algoritmo corrispondente. Infatti l'algoritmo della fidelity per sito richiederebbe di partire da un numero di qubit n molto maggiore di 1 e molto minore di N , cosa che risulta difficile nella nostra implementazione. Nel nostro caso, quindi limitandoci per motivi computazionali ad un numero di qubit pari a $N = 10$, possiamo affermare che i risultati ottenuti con la fidelity classica individuano il valore critico in un punto più vicino a quello atteso.

Come si è discusso nella sezione riguardante la scalabilità, inoltre, utilizzare un numero maggiore di qubit dovrebbe comportare la creazione di una matrice della fidelity più definita e con una simmetria più evidente attorno a $J = 1$.

Viene riportato di seguito un breve reseconto delle soglie critiche ottenute per ciascun kernel:

Kernel	RBF con $\gamma = 0.05$	RBF con $\gamma = 1$	Fidelity (SV)	Fidelity	Fidelity per sito
J_c	1.0	1.0	0.93	0.91	0.85

Tabella 1: Valori critici trovati con intervallo di apprendimento $\delta_0 = [0.25, 1.75]$

Kernel	RBF con $\gamma = 0.05$	RBF con $\gamma = 1$	Fidelity (SV)	Fidelity	Fidelity per sito
J_c	-	1.23	1.02	1.01	0.86

Tabella 2: Valori critici trovati con intervallo di apprendimento $\delta_1 = [0.8, 0.9] \cup [1.6, 1.7]$

Confrontando le due tabelle, si può concludere che la scelta dell'intervallo di training svolge un ruolo fondamentale nella capacità della SVM di determinare l'effettivo valore di J alla quale avviene la transizione di fase. Gli effetti di scala risultano essere importanti, traslando il valore di J_c mediamente verso valori più piccoli di uno. Ciò dipende dalla scelta del label $y = 1$ per la fase ferromagnetica e $y = -1$ per la fase paramagnetica. La fidelity per sito risulta essere molto simile nei due casi. Ciò può essere implicato dalla presenza del fit nell'algoritmo, che con la sua presenza, impone un vincolo in più sulla forma del kernel.