applied sciences

MDPI

Article

Quantum-Inspired Classification Algorithm from DBSCAN–Deutsch–Jozsa Support Vectors and Ising Prediction Model

Kodai Shiba [1,2], Chih-Chieh Chen [2], Masaru Sogabe [2], Katsuyoshi Sakamoto [1,3] and Tomah Sogabe [1,2,3,*]

[1] Engineering Department, The University of Electro-Communications, Tokyo 182-8585, Japan; s1933062@edu.cc.uec.ac.jp (K.S.); katsuyoshi.sakamoto@uec.ac.jp (K.S.)
[2] Grid, Inc., Tokyo 107-0061, Japan; chen.chih.chieh@gridsolar.jp (C.-C.C.); sogabe@gridsolar.jp (M.S.)
[3] i-PERC, The University of Electro-Communications, Tokyo 182-8585, Japan
* Correspondence: sogabe@uec.ac.jp

Abstract: Quantum computing is suggested as a new tool to deal with large data set for machine learning applications. However, many quantum algorithms are too expensive to fit into the small-scale quantum hardware available today and the loading of big classical data into small quantum memory is still an unsolved obstacle. These difficulties lead to the study of quantum-inspired techniques using classical computation. In this work, we propose a new classification method based on support vectors from a DBSCAN–Deutsch–Jozsa ranking and an Ising prediction model. The proposed algorithm has an advantage over standard classical SVM in the scaling with respect to the number of training data at the training phase. The method can be executed in a pure classical computer and can be accelerated in a hybrid quantum–classical computing environment. We demonstrate the applicability of the proposed algorithm with simulations and theory.

Keywords: quantum-inspired algorithm; Deutsch–Jozsa algorithm; Ising model; support-vector machine

Citation: Shiba, K.; Chen, C.-C.; Sogabe, M.; Sakamoto, K.; Sogabe, T. Quantum-Inspired Classification Algorithm from DBSCAN–Deutsch–Jozsa Support Vectors and Ising Prediction Model. *Appl. Sci.* **2021**, *11*, 11386. https://doi.org/10.3390/app112311386

Academic Editors: Andrea Prati, Carlos A. Iglesias, Vincent A. Cicirello and Luis Javier García Villalba

Received: 2 October 2021
Accepted: 9 November 2021
Published: 1 December 2021

1. Introduction

Quantum machine learning [1–3] is an emerging field using quantum information processing for machine learning applications. Various quantum machine learning algorithms have been proposed, such as quantum Boltzmann machines [4], supervised quantum autoencoders [5], reinforcement learning using quantum Boltzmann machines [6], quantum circuit learning [7,8] and some quantum methods for linear algebra [9–11]. Even though the exponentially large Hilbert space is expected to help tackling the "curse of dimensionality" in machine learning, loading large classical data sets into a small quantum memory is a bottleneck [12,13]. Even though this problem can be circumvented in some settings [14], the available quantum information processors are restricted by the small number of qubits and high operational noise in the current noisy intermediate-scale quantum [15] era. On the other hand, these quantum algorithms have inspired various new classical efficient methods [16–19]. The dequantizing of quantum algorithms is not only for practical concerns, but also for theoretical interests in complexity theory [20].

The support-vector machine (SVM) is a widely used classification algorithm [21–24]. There are also newly developed variants of the SVM [25,26] and several quantum versions of the SVM have been proposed [27–29]. One proposal is to use the HHL algorithm [9] to invert the kernel matrix [27]. Schuld et al. [28] proposed a QSVM algorithm in which the distance for the kernel classifier is evaluated by single qubit gate circuits. Recently, Havlicek et al. [29] implemented the quantum circuit learning classification and a quantum kernel estimator for the SVM. However, due to the requirements of quantum coherence, the applications of these algorithms to large-scale data sets has not been demonstrated yet.

In this work, we propose a new quantum-inspired algorithm for classification tasks. The support vectors are selected from the data by a principle inspired by the classical clustering algorithm density-based spatial clustering of applications with noise (DBSCAN) [30] and the quantum algorithm Deutsch–Jozsa (DJ) [31]. DBSCAN is a commonly used classical clustering algorithm that determines clusters by considering numbers of data points within some distance parameter ϵ. DBSCAN is efficient, noise-resilient and does not require the input of the number of clusters. DJ is a deterministic quantum algorithm that determines a specific property of a given Boolean function $f : \{0,1\}^n \to \{0,1\}$ with a single oracle call to the function f. The classical deterministic algorithm to solve the DJ problem requires $O(2^n)$ calls. We combine DBSCAN and DJ to identify support vectors that are close to the classification boundary. The support vectors are then used for prediction based on an Ising model classifier. The prediction labels are identified as spins in an Ising model and the prediction label is aligned with nearby support vectors through spin–spin interactions. The training time complexity of the proposed algorithm scales according to $O(l^2 d)$, where l is the number of training data and d is the feature dimension. This is favorable comparing to $O(l^3 d)$ standard kernel support-vector machines, such as LIBSVM, used in scikit-learn [21–24]. The proposed algorithm can be executed on classical computers, while quantum devices could be used to accelerate several subroutines in the algorithm. The numerical verification is presented. We also provide theoretical evidences for the learnability of the algorithm. We show that the training algorithm is described by an integer linear programming problem. We also derive upper bounds for the VC dimension of the Ising classifier in two special lattices.

2. The Learning Algorithm

2.1. Hypothesis Set

In this subsection, we briefly illustrate the hypothesis set and training and testing algorithms. The flow chart is illustrated in Figure 1. The details are discussed in the following sections. Given the training data set $\tau = \{(\vec{x}_i, y_i)|i = 1, \ldots, l\}$, where $y_i \in \{\pm 1\}$ are the labels for binary classification, the goal of a classification algorithm is to find a hypothesis $f_\theta : (\vec{x}_1, \ldots, \vec{x}_{n_t}) \mapsto \{-1, +1\}^{n_t}$ with small prediction error, where n_t is the number of test data. The classification algorithm consists of two parts, training and prediction. A training algorithm finds a hypothesis by minimizing some loss function $L(f_\theta(\vec{x}_i), y_i)$. Our hypothesis set consists of the ground states of the Ising model. The hypothesis set is

$$f_\theta(\vec{x}_1, \ldots, \vec{x}_{n_t}) = (s_1, \ldots, s_{n_t}),\ \vec{x}_i \in T \text{ for } 1 \le i \le n_t, \tag{1}$$

$$(s_1, \ldots, s_{n_t+n_{sv}}) = \arg\min_{\{s_k\}} [- \sum_{\substack{i<j \\ \vec{x}_i, \vec{x}_j \in S \cup T}} J_{i,j}(\vec{x}_i, \vec{x}_j) s_i s_j - \sum_{\vec{x}_i \in S \cup T} h_i s_i], \tag{2}$$

where $S = \{\text{support vectors}\}$ and $T = \{\text{test data}\}$, $n_{sv} = |S|$, $n_t = |T|$, $s_i \in \{-1, 1\}$. $h_i = y_i$ for $\vec{x}_i \in S$ and $h_i = 0$ for $\vec{x}_i \in T$. The free parameters of the model are, in general, $\theta = (J_{ij})$. We use a physics-motivated ansatz to efficiently determine the support vectors and θ from three free parameters, $\theta' = (\alpha, \beta, \epsilon)$, as explained in the following subsections. Finally, the training is performed by minimizing the loss function with respect to the remaining three free parameters, $\theta = (\alpha, \beta, \epsilon)$, with particle swarm optimization (PSO). The loss function is the cross-entropy loss function

$$L(f_\theta(\vec{x}_i), y_i) = -(1 - q(\vec{x}_i)) \ln(p_{-1}(\vec{x}_i)) - q(\vec{x}_i) \ln(p_{+1}(\vec{x}_i)), \tag{3}$$

where $q(\vec{x}_i) = (1 + y_i)/2$; $p_{\pm 1}(\vec{x}_i)$ is the probability that $s_i = \pm 1$ for the Ising ground state.

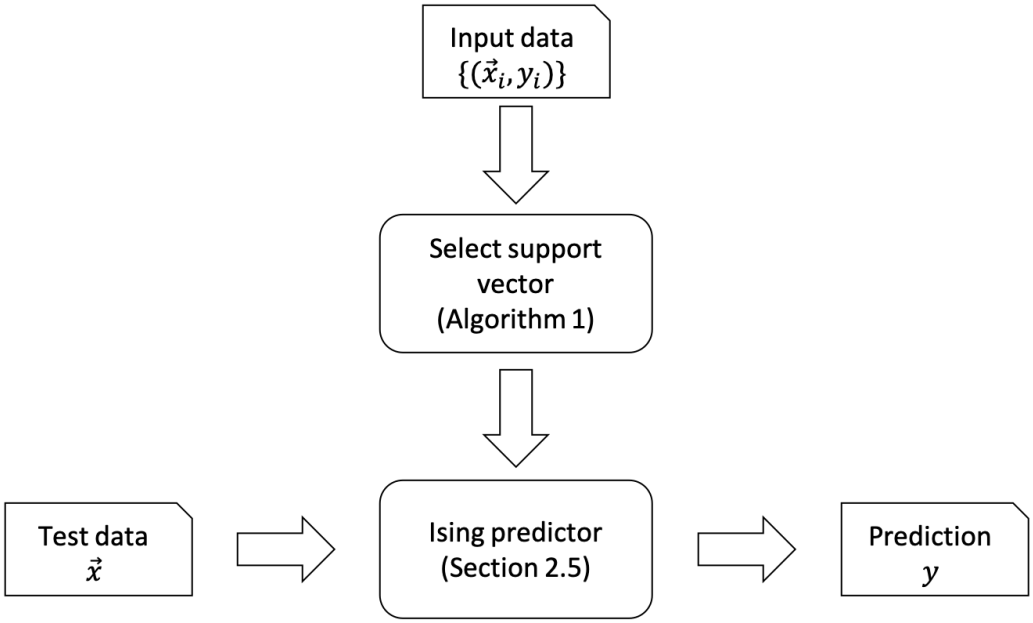

Figure 1. Outline of the proposed algorithm. The training data set is fed into the support vector selection algorithm. The support vectors are then used in the Ising predictor for prediction. The support vector selection algorithm is described in Algorithm 1 and the Ising predictor is described in Section 2.5.

2.2. DBSCAN Algorithm for Determining the Support Vector

The training algorithm is designed by applying the DBSCAN clustering algorithm to a Deutsch–Jozsa decision problem on the training data. The idea is depicted in Figure 2. Let us consider the hypersphere $B_\epsilon(\vec{x}_i)$ of radius ϵ centered in the data point $\vec{x}_i$. Near the classification boundary, it is more likely that the data in the sphere have two different classification labels. On the other hand, for the data in the interior of each cluster, it is more likely that all the data points in the sphere have the same label. Hence, the classification boundary can be identified by a DBSCAN scanning through all the data points. For each data point, we query a decision problem, namely, whether the data inside the sphere is constant or not. If the number of non-constant spheres around is large for a given data point, we then select this point as a support vector for the prediction algorithm. This decision problem takes the form of a Deutsch–Jozsa problem and the potential use of the quantum DJ algorithm to enhance the training is discussed in Section 2.4.

After selecting the support vectors from the training data, we compute a kernel matrix $(J_{sv})_{i,j} = J(\vec{x}_i, \vec{x}_j)$ for $\vec{x}_i, \vec{x}_j \in \{\text{support vectors}\}$. This kernel matrix will be used in the prediction algorithm. The matrix elements are, in general, some functions of data positions. We choose J to be the inverse of some power-law function of distances between data points:

$$J(\vec{x}_i, \vec{x}_j) = \frac{1}{||\vec{x}_i - \vec{x}_j||^\beta}. \tag{4}$$

The training algorithm can be described as the pseudo-code Algorithm 1.

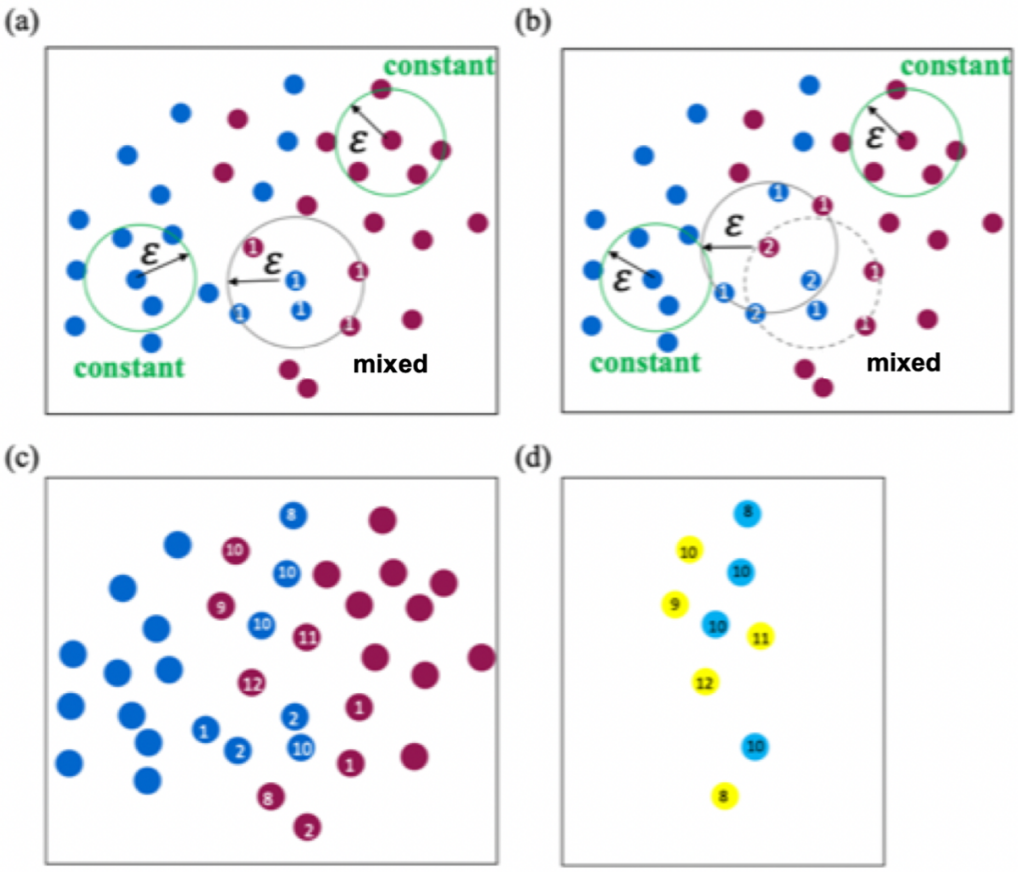

Figure 2. The training process. The red dots and blue dots are training data with two different classification labels. (**a**) For each data point $\vec{x}_i$, we consider the hypersphere $B_\epsilon(\vec{x}_i)$ of radius ϵ centered in the data point $\vec{x}_i$. We ask the question of whether all the data points in the sphere have the same label or not. (**b**) If the data within the sphere are mixed (not constant), then all the data within the sphere receive one count by $c[j] = c[j] + 1$. (**c**) After scanning through all the data, we rank the data according to $c[i]$. (**d**) The data points with the highest $c[i]$ are selected as support vectors. The blue and yellow dots are support vectors for two different classes.

Algorithm 1: The training algorithm to determine support vectors.

Input: Training dataset $\{(\vec{x}_i, y_i)|i = 1, \ldots, l\}$, ϵ, α, β
Output: Support vectors $\{(\vec{x}_i, y_i)|i = 1, \ldots, n_{sv}\}$, $(J_{sv})_{i,j}$

Allocate an integer array $c[i]$, $i \in \{1, \ldots, l\}$ where l is the number of training data
for *each data point* $\vec{x}_i$ *in the training dataset :* **do**
    Making a query:
    Examine the class label y_j of all the training data points $\vec{x}_j \in B_\epsilon(\vec{x}_i)$
    Determine whether the training data in the sphere including the central data point are all the same class (constant) or mixed with other classes (mixed)
    if *mixed* **then**
        $c[j] \leftarrow c[j] + 1$ for all $\vec{x}_j \in B_\epsilon(\vec{x}_i)$
    end
end
Rank $c[i]$
The top $1/\alpha$ data points in the ranking are the support vectors
Compute matrix $(J_{sv})_{i,j}$ for $\vec{x}_i, \vec{x}_j \in \{\text{support vectors}\}$ according to Equation (4)

2.3. Integer Linear Programming Formulation

We show that the training algorithm is equivalent to solving an integer linear programming problem. We use the following definitions: $B_\epsilon(\vec{x}_i)$ is the hypersphere of radius ϵ centered in $\vec{x}_i$; $Y_{ij} = \frac{1}{2}(1 + y_i y_j)$ is the indicator y-correlation matrix of training labels; $K^\epsilon_{ij} = \kappa_\epsilon(\vec{x}_i, \vec{x}_j) = \lim_{\gamma\to\infty} \frac{1}{2}[1 + \tanh(\gamma(\epsilon - ||\vec{x}_i - \vec{x}_j||))] = H(\epsilon - ||\vec{x}_i - \vec{x}_j||)$ is the indicator kernel matrix, where $H(x)$ is the Heaviside step function. The indicator kernel matrix satisfies

$$K^\epsilon_{ij} = \begin{cases} 1, & \text{if } \vec{x}_j \in B_\epsilon(\vec{x}_i) \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

The indicator y-correlation matrix satisfies

$$Y_{ij} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{if } y_i = -y_j \end{cases}. \tag{6}$$

From the definitions, we have

$$\prod_{p=1}^{l} Y_{jp}^{K_{jp}} = \begin{cases} 1, & \text{if } \{y_k | \vec{x}_k \in B_\epsilon(\vec{x}_j)\} \text{ is constant} \\ 0, & \text{if } \{y_k | \vec{x}_k \in B_\epsilon(\vec{x}_j)\} \text{ is mixed} \end{cases}. \tag{7}$$

Then, the counting number returned by the DBSCAN-DJ training procedure is $c_i = \sum_{j=1}^{l} K_{ij}(1 - \prod_{p=1}^{l} Y_{jp}^{K_{jp}})$. Since all $c_i > 0$, the ranking algorithm selects support vectors that satisfy the solution for the constrained integer linear programming problem:

$$\min_{\alpha_i} \sum_{i}^{l} c_i \alpha_i \tag{8}$$

$$\alpha_i \in \{0, 1\} \tag{9}$$

$$\sum_{i}^{l} \alpha_i = \lfloor \frac{1}{\alpha} \rfloor, \tag{10}$$

where $\alpha_i = 1$ if and only if $\vec{x}_i$ is selected as a support vector. Notice that, if the number of training data is l, the sorting only takes $O(l \log l)$ time, while general integer linear programming is NP-complete.

2.4. Quantum-Enhanced Algorithm with DJ

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, that is guaranteed to be constant or balanced, and its reversible circuit $U_f : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$, the DJ algorithm determines whether f is constant or balanced with one call to U_f. A balanced function means $f(x) = 1$ for half of the possible input string x. The DJ algorithm is conducted using operator $(H^{\otimes n} \otimes 1)U_f H^{\otimes(n+1)}$ on the initial state $|0\rangle^{\otimes n}|1\rangle$. The probability to measure an all-zero string $x = 00\ldots0$ is $P_0 = |\frac{1}{2^n}\sum_x (-1)^{f(x)}|^2$. Hence, if f is constant, the probability to obtain a measurement result $00\ldots0$ is $P_0 = 1$. If f is balanced, then $P_0 = 0$. If f is not constant nor balanced, then $1 > P_0 > 0$.

Given the data $\{(\vec{x}_i, y_i)\}$ in a sphere, we construct the reversible circuit for the function $f : \vec{x}_i \to (1 + y_i)/2$. This serves as the oracle function in the DJ algorithm. A four-data point example is shown in Table 1 and Figure 3. For n data points, the reading of the data and the construction of the oracle U_f take $O(n)$ time. However, once given the oracle, the DJ algorithm can determine whether the function is constant or balanced with a single call. In the case where the data are not constant, there is some probability to obtain measurement results other than $0\ldots0$. We could repeat the algorithm several times (the number of repetitions is another hyper-parameter that determines the precision) to obtain a reasonable estimation of the probability of how likely that is a constant function. Then, this probability provides an estimation for c_i. For an efficient implementation of

the U_f circuit, one could utilize the single-step n-Toffoli gate [32] in a quantum computer. While this scheme works in principle, it is difficult to implement and the overhead is large. In our simulation, we take an simplified oracle consisting of one n-Toffoli gate and one n-Toffoli gate with negative control, as shown in Figure 4. Each qubit is encoded in a classical indicator state $(1+y_i)/2 \in \{0,1\}$. Then, a single query to the oracle gives the measurement result $\prod_i (1+y_i)/2$, which indicates whether the data are constant or not. This approach is simpler to implement, but requires more encoding qubits. In the original DJ, it takes $O(\log n)$ for n data points, while, in this simplified approach, it takes $O(n)$ qubits. However, it has a potential acceleration similar to that of DJ, where a single call to the oracle gives the desired result.

Table 1. An example of four training data's look-up table for function $f(x)$.

x_i	y_i	$(1+y_i)/2$
00	-1	0
01	1	1
10	1	1
11	-1	0

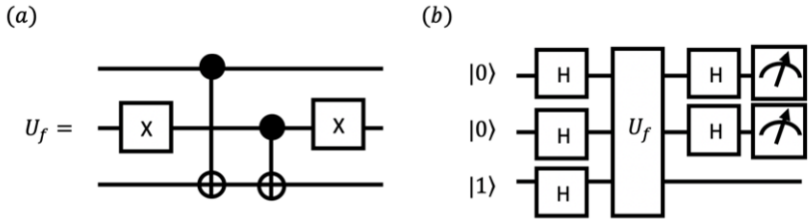

Figure 3. The quantum circuit for the 4-data point example in Table 1. (**a**) The oracle circuit U_f. (**b**) The circuit for the DJ algorithm.

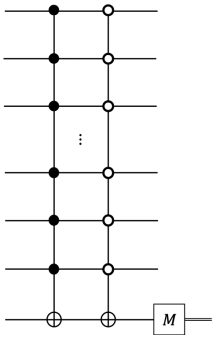

Figure 4. The quantum circuit implementation for the DJ query. The circuit consists of one n-Toffoli gate and one n-Toffoli gate with negative control.

2.5. Annealing Algorithm for Data Prediction

The classical SVM algorithm constructs $\vec{w}$ and b from the training data and then predicts each test data by computing $\vec{w} \cdot \vec{x} + b$. We propose an algorithm such that it is possible to predict multiple test data points with a single run of the prediction algorithm. The prediction is made by finding the ground state of the Ising model:

$$H = -\sum_{i<j} J_{i,j} s_i s_j - \sum_i h_i s_i. \tag{11}$$

The number of spins is $N = n_{sv} + n_t$, where n_{sv} is the number of support vectors and n_t is the number of test data. The coupling constants are functions of the data positions $J_{ij}(\vec{x}_i, \vec{x}_j)$ defined in Equation (4). We use the notation

$$J_{ij} = \begin{cases} (J_{sv})_{ij}, & \text{if } \vec{x}_i, \vec{x}_j \in \{\text{support vectors}\} \\ (J_t)_{ij}, & \text{if } \vec{x}_i, \vec{x}_j \in \{\text{testing data}\} \\ (J_{int})_{ij}, & \text{if } \vec{x}_i \in \{\text{testing data}\}, \vec{x}_j \in \{\text{support vectors}\} \end{cases}. \tag{12}$$

J_{sv} is an $n_{sv} \times n_{sv}$ matrix, J_t is an $n_t \times n_t$ matrix and J_{int} is an $n_{sv} \times n_t$ matrix. Notice that the J_{sv} matrix is given by the training algorithm and does not need to be computed again. The magnetic field is defined as

$$h_i = \begin{cases} y_i, & \text{if } \vec{x}_i \in \{\text{support vectors}\} \\ 0, & \text{if } \vec{x}_i \in \{\text{testing data}\} \end{cases}. \tag{13}$$

The prediction result is given by the ground state of the Ising model:

$$y_i = s_i \text{ , for } \vec{x}_i \in \{\text{testing data}\}. \tag{14}$$

The configurations s_i of the support vectors are dummy variables and not used for prediction. Intuitively, the model favors parallel spins, since $J_{ij} > 0$. Hence, we expect the prediction to have the same spin direction as nearby training data. In principle, the Ising ground state can be calculated by any Ising solver. However, the problem of finding the general Ising model ground state is equivalent to quadratic unconstrained binary optimization (QUBO), which is NP-hard. In this work, we use adiabatic quantum computing implemented on a quantum circuit simulator to find the ground state [33]. The prediction labels are determined by

$$s_i = \begin{cases} -1, & \text{if } p_{-1}(\vec{x}_i) > p_{+1}(\vec{x}_i) \\ +1, & \text{if } p_{-1}(\vec{x}_i) < p_{+1}(\vec{x}_i) \end{cases}, \tag{15}$$

where $p_\pm$ is the probability that the point label is predicted to be ± 1.

2.6. The VC Dimension

We further show that the Ising classifier has a finite VC dimension upper bound in two simplified toy models in Figure 5. These simplifications can be regarded as regularization methods. The intuition is that, for $J_{ij} > 0$, anti-parallel spin configurations have higher energy and it is difficult that they are a ground state. This difficulty sets a limitation on how many points can be shattered and leads to an upper bound for the VC dimension. The detail derivation is in Appendix A. We use the following definitions: n_{sv} is the number of support vectors; n_t is the number of prediction data points; $N = n_{sv} + n_t$ is the total number of spins in the Ising model; d is the feature space dimension. We assume that $n_{sv} \leq n_t$. For the first toy model, we consider a simplification by introducing equal distance constraints to the support vectors and test data. This means $J_{ij} = \frac{1}{||\vec{x}_i - \vec{x}_j||} = J$ for all (i, j). The Ising prediction model, in this case, is the fully connected infinite-dimensional mean field Ising model:

$$H = -J \sum_{i<j} s_i s_j. \tag{16}$$

Notice that this is a strong constraint which requires $N \leq d + 1$, but there is still one continuous parameter J, so the hypothesis set is still infinite. This model is exactly solvable. The eigenenergy is $E_m = J(-\frac{1}{2}N(N-1) + 2m(N-m))$ for eigenstates with m spin up and $N - m$ spin down. If there are two test data, the anti-parallel state $\uparrow\downarrow$ is always an excited state; hence, it is impossible it is part of a ground state. We obtain the result $d_{VC} = 1$ in this case.

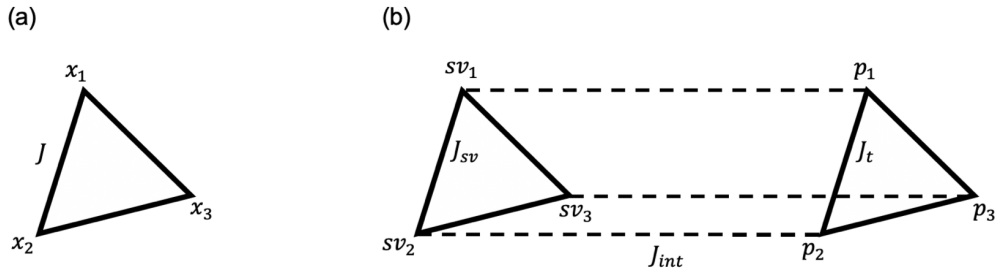

Figure 5. (**a**) Fully-connected mean field Ising model. (**b**) A relaxed model where there are only one-to-one couplings between support vectors and test data.

This upper bound can be relaxed by considering a model with less constraints. Consider a second toy model where all the support vectors form a fully connected equal-distance lattice defined by parameter J_{sv}. All the test data form another fully connected equal-distance lattice defined by another parameter, J_t. The distance between support vectors and prediction data is parametrized by a different variable J_{int}. Hence, there are three continuous variables in the model. The interaction between support vectors and prediction data points is restricted to one-to-one coupling. Let

$$\begin{cases} \{\text{support vectors}\} & = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{n_{sv}}\} \\ \{\text{test data}\} & = \{\vec{x}_{n_{sv}+1}, \vec{x}_{n_{sv}+2}, \ldots, \vec{x}_{n_{sv}+n_t}\} \end{cases}. \tag{17}$$

Then, the coupling for this model is

$$J_{ij} = \begin{cases} J_{sv}, & \text{if } 1 \leq i < j \leq n_{sv} \\ J_t, & \text{if } (n_{sv}+1) \leq i < j \leq (n_{sv}+n_p) \\ J_{int}, & \text{if } 1 \leq i \leq n_{sv} \text{ and } j = i + n_{sv} \\ 0, & \text{otherwise} \end{cases}. \tag{18}$$

For this model, we obtain the result $d_{vc} \leq \sqrt{\frac{2J_{int}}{J_t} n_{sv}}$. The intuition is that, if there are more support vectors (large n_{sv}), the model has more degrees of freedom. If the support vectors are closer to the test data (large J_{int}), the model has stronger influences on the test data. Hence, the Ising model can shatter more test data for larger J_{int} and n_{sv}. On the other hand, if the distance among test data is small (large J_t), then the strong interactions among test data make it difficult to be shattered by the model.

3. Simulation Results

We demonstrate the classification task for two types of teacher data (linear classification boundary and non-linear classification boundary) using the proposed method. The learning of the optimal $(\alpha, \beta, \epsilon)$ parameters was performed by minimizing the cost function. Since the $(\alpha, \beta, \epsilon)$ parameters are associated with discrete ranking, we chose gradient-free optimization instead of gradient-based optimization. We implemented the Nelder–Mead method and particle swarm optimization (PSO) algorithms. We observed that PSO generally gives better results; hence, we focused on the results obtained by PSO. The hyperparameter N is the number of particles for PSO. The learned support vectors, classification boundary and learning curves for $N = 10, 50, 100$ are depicted in Figure 6. The learned parameters are shown in Table 2. In all the cases, the total number of support vectors was fixed at $n_{sv} = 10$, where five of them belonged to $+1$ and the other five belonged to -1. The α and the ϵ parameters were fixed to $\alpha = 3$ and $\epsilon = 0.4$. The optimization was performed by minimizing the loss function Equation (3) for the optimal values of β. We observed that the classification could be performed in all experiments. We also observed that, for larger values of N, the optimized loss function had a lower value, but the

test accuracy was also lower. This suggests possible overfitting for large values of N in PSO optimization.

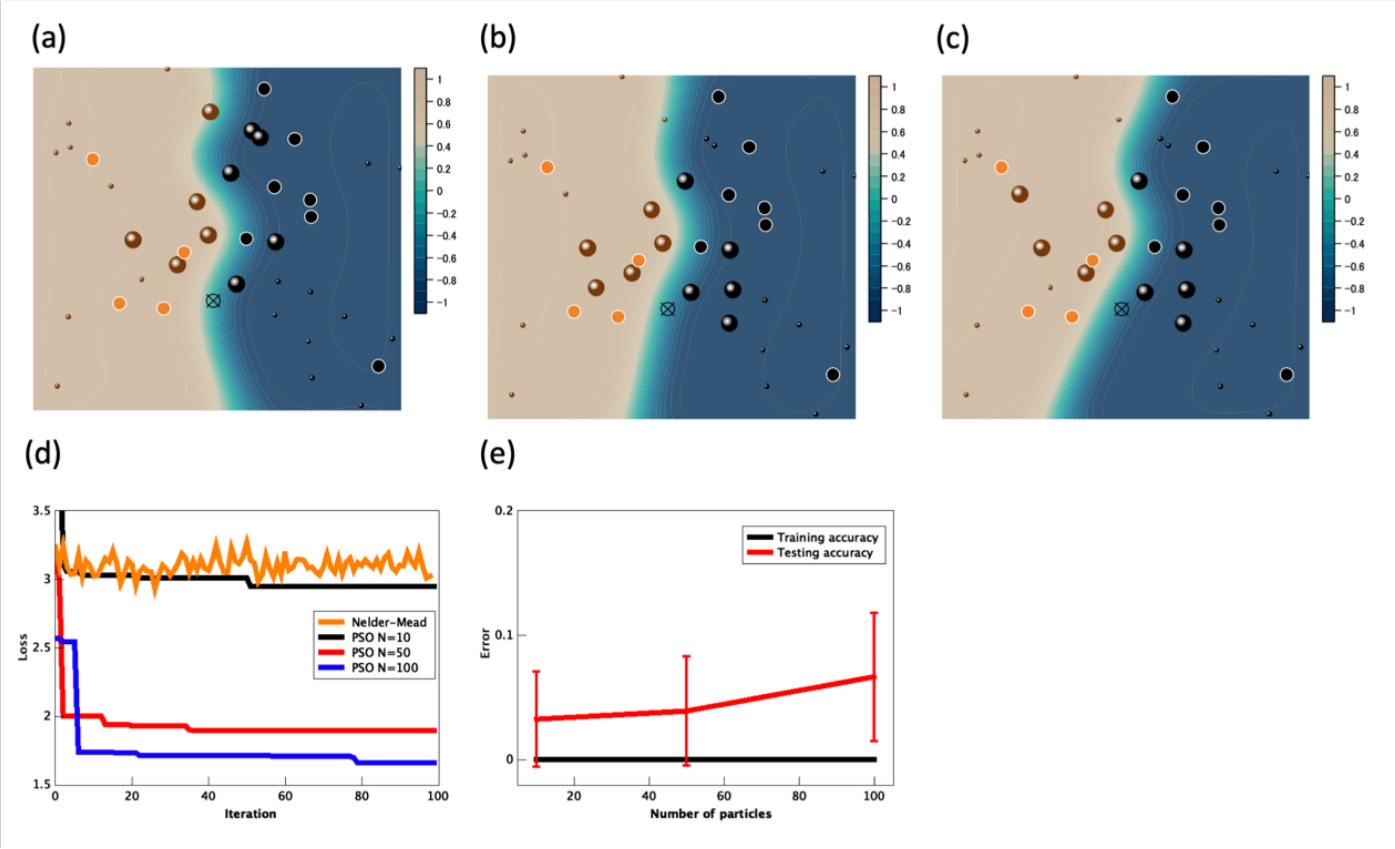

Figure 6. Support vector distribution, prediction probability and classification boundary from particle swarm optimization (PSO) results. The experimental parameters and optimal values are listed in Table 2. (**a**) $N = 10$. (**b**) $N = 50$. (**c**) $N = 100$. The small dots are the training data. The symbols bordered by white circles are the test data that were classified correctly. The crossed circles are the test data that were classified incorrectly. The large sphere symbols are the support vectors. (**d**) The learning curves for four experiments. The orange line is the Nelder–Mead experiment which does not converge. The other three lines are PSO results. The black line is $N = 10$, the red line is $N = 50$ and the blue line is $N = 100$, respectively. (**e**) The training and testing accuracy suggest overfitting for larger values of N.

Table 2. Particle swarm optimization results. N is the number of particles and β is the hyperparameter being optimized. The optimization is conducted by minimizing the cross-entropy loss function Equation (3). The optimized results are visualized in Figure 6.

N	β	Test Accuracy
10	1.128	0.9675
50	0.6356	0.9610
100	0.6157	0.9335

We now compare the proposed method with scikit-learn SVM. The verification results are shown in Figure 7. We used 28 teacher data (small dots) for each verification. The detail numbers of data used are listed in Table 3. The large sphere symbols are the support vectors. In the classification using the proposed algorithm, we set the radius of the scan circle to $\epsilon = 0.5$ (linear data) and $\epsilon = 0.6$ (non-linear data). The power of the reciprocal of the distance among data was $\beta = 1$ and the criterion for determining the support vectors from the ranking was $\alpha = 3$. For the traditional SVM, we used scikit-learn.svm.SVC with default values. From Figure 7, we found that both the linear classification and the non-linear classification performed by the proposed method could give a classification

boundary similar to that of scikit-learn. However, if we changed only the positions of teacher data without changing the number of teacher data, there were cases where it could not be classified well. A possible cause is that the number of teacher data was too small. As a result, the number of support vectors was insufficient and the classification accuracy might have been reduced. Therefore, we expected a better classification result by using a larger number of training data. However, we noticed that quantum adiabatic calculations required a number of qubits of $n_{sv} + n_t$, where n_{sv} is the number of support vectors and n_t is the number of test data. Our experiment was limited by this restriction.

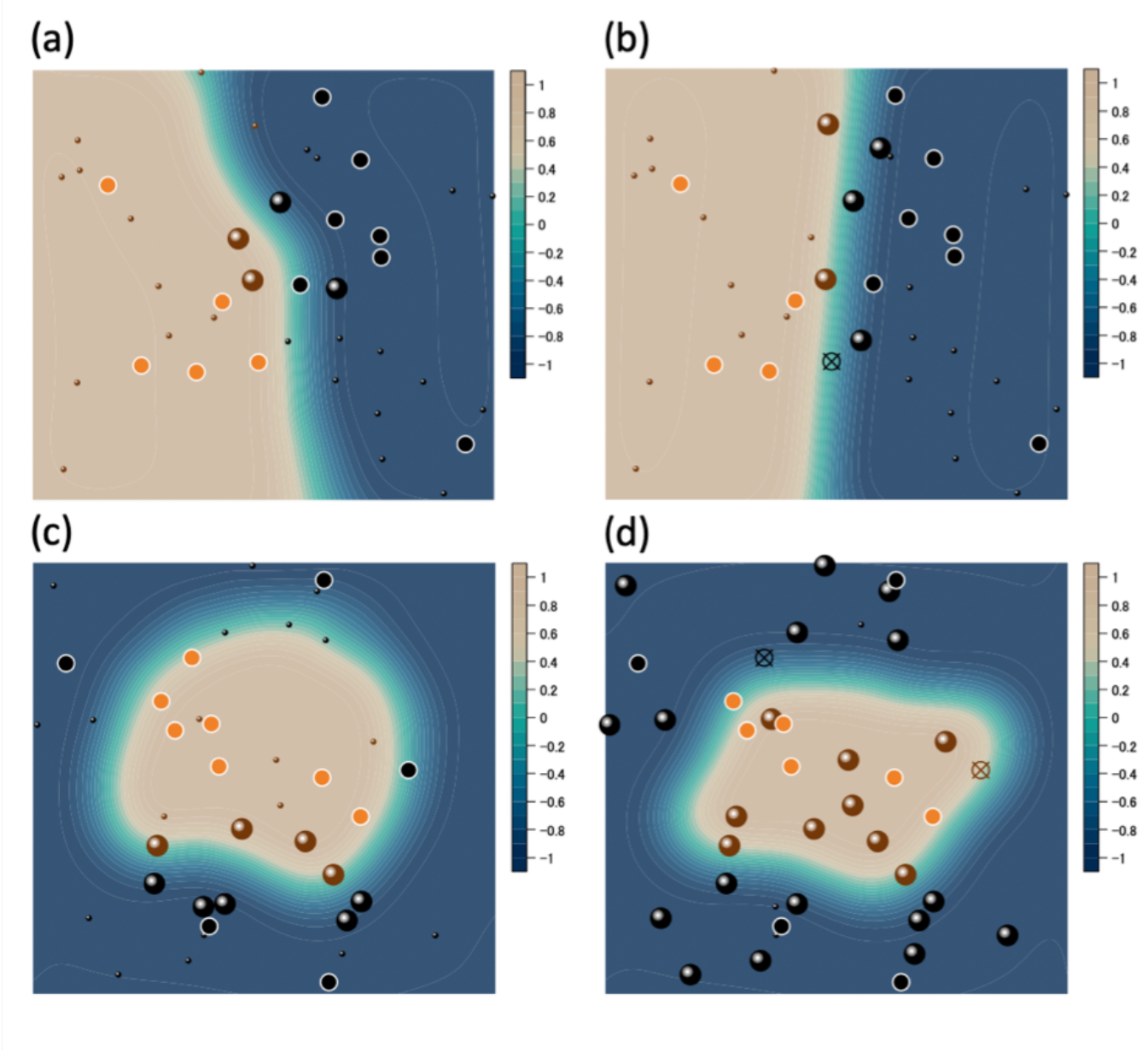

Figure 7. Classification results. The small dots are the training data. The symbols bordered by white circles are the test data that were classified correctly. The crossed circles are the test data that were classified incorrectly by scikit-learn SVM, but correctly predicted by the proposed algorithm. The large sphere symbols are the support vectors. (**a**) Linear data classification by the proposed algorithm. (**b**) Linear data classification by scikit-learn SVM. (**c**) Non-linear data classification by the proposed algorithm. (**d**) Non-linear data classification by scikit-learn SVM.

Table 3. The number of training and testing data used in the experiment in Figure 7.

Data	Total	+1 Label	−1 Label
Linear Training	28	13	15
Linear Testing	12	5	7
Non-linear Training	28	9	19
Non-linear Testing	12	7	5

4. Computational Complexity

We investigate the efficiency of the proposed algorithm. Table 4 shows the time complexity of the kernel SVM and the proposed algorithm. The training algorithm is DBSCAN; hence, the worst-case complexity is essentially the same as DBSCAN which takes $O(l^2)$ time for computing distances among l training data points. This is better than the standard kernel SVM, which takes the worst-case time of $O(l^3)$ for kernel matrix inversion (empirical scaling is $O(l^{2+\delta})$ for some $1 > \delta > 0$) [22,23]. The training algorithm has also the feature of Deutsch–Jozsa; hence, it can be quantum-enhanced by using an n-Toffoli gate to speed it up. The speed-up does not change the worst-case scaling of the training algorithm's complexity. The cost of the prediction part is $O((n_{sv} + n_t)n_t d)$ for computing the J_t and J_{int} matrices. This scaling seems to be worse than that of the kernel SVM, but, in practice, this is not an issue. We could divide the test data into n_b batches and run the prediction algorithm one time for each batch. The scaling becomes $O((n_{sv} + \frac{n_t}{n_b})n_t d)$ in this case. For example, when $n_b = n_t$ (i.e., we predict one datum at the time, as in the kernel SVM), we recover the same scaling as the kernel SVM. The cost of the prediction part depends on finding the ground state of the Ising model, which is equivalent to solving a QUBO problem. This, in general, could take an exponentially large amount of time, depending on the solver algorithm. However, to make a reasonable prediction, it is not necessary to find the optimal solution. The Ising prediction part can be solved by classical simulated annealing, or other QUBO solver and can be potentially accelerated by using adiabatic quantum computation or quantum annealing [34]. One might question about the possibility to combine the DBSCAN–DJ training algorithm with the kernel SVM prediction. However, since our training algorithm does not produce $\vec{w}$ and $\vec{b}$ vectors, it cannot be used for kernel SVM prediction.

Table 4. Worst-case time complexity for the proposed algorithm comparing to the standard kernel SVM [22–24]. d is the feature space dimension, l is the number of training data, n_{sv} is the number of support vectors, n_t is the number of test data and n_b is the number of batches for the test data. T_a is the annealing time to find the Ising ground state, which depends on the choice of Ising solver algorithm.

	Training	Prediction
Kernel SVM	$O(l^3 d)$	$O(n_{sv} n_t d)$
This work	$O(l^2 d)$	$O((n_{sv} + \frac{n_t}{n_b})n_t d) + T_a$

5. Conclusions

In this work, we propose a quantum-inspired algorithm for classification. The algorithm has an advantage over the traditional kernel SVM in the training stage. The algorithm can be performed on a pure classical computer and can be enhanced in a hybrid quantum–classical computing environment. We experimentally verify the algorithm and provide theoretical background for the algorithm. The applicability to more general classification problems and possible improvements will be studied in future works. For example, multi-class classifications could be treated by one-versus-rest or one-versus-one approaches [35]. The optimal value for ϵ could be determined by other methods [36].

Author Contributions: K.S. (Kodai Shiba) carried out simulation and experiment. C.-C.C. wrote the manuscript and carried out the theoretical simulation. M.S. and K.S. (Katsuyoshi Sakamoto) contributed to research design. T.S. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and scripts that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: We thank Naoki Yamamoto for valuable discussions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. VC Dimension of Ising Predictor

In this appendix, we provide the derivation for the VC dimension of the Ising classifier toy models. First, we show a simple result that one test datum can always be shattered by one support vector; hence, there is a general lower bound $d_{VC} \geq 1$. If there is only one support vector s_1 and one test datum s_2, the Hamiltonian is $H = -Js_1s_2 - h_1s_1 = -s_1(Js_2 + h_1)$. Given any s_2, we can choose $s_1 = s_2$ to produce the desired ground state. Hence, one point can always be shattered by one support vector. Hence, $d_{VC} \geq 1$.

Then, we consider the fully connected Ising model $H = -J\sum_{i<j} s_is_j$. The eigenenergy $E_m = J(-\frac{1}{2}N(N-1) + 2m(N-m))$ for eigenstates with m spin-up and $(N-m)$ spin-down. The spectrum is a parabolic curve concave downward as a function of m. The doubly degenerated ground states are all spin-up and all spin-down states with energy $E_0 = -\frac{J}{2}N(N-1)$. The highest energy state occurs at $m = \lfloor \frac{N}{2} \rfloor$ with half spin-up and half spin-down and the energy is $E_{max} = J\lfloor \frac{N}{2} \rfloor$.

Now, we can consider the first prediction model defined in Equation (16). Let us consider two configurations of prediction data: (A) half spin-up and half spin-down; (B) all spins aligned in the same direction. To shatter the prediction data, the model has to be able to achieve a ground state when the given test data are in the highest energy state $E_t^A = J\lfloor \frac{n_t}{2} \rfloor$. The best possible choice of the support vectors is that all spins are aligned in the same direction with some energy E_{sv}. The interaction between support vectors and test data gives the energy $E_{int}^A = -\frac{J}{2}(n_t \bmod 2)$. The resulting total energy is $E_{total}^A = E_t^A + E_{int}^A + E_{sv} = J\lfloor \frac{n_t}{2} \rfloor - \frac{J}{2}(n_t \bmod 2) + E_{sv}$. To predict configuration (A), E_{total}^A must be lower than the state E_{total}^B, where all prediction spins are aligned with all support vectors. The total energy in the second case is $E_{total}^B = -\frac{J}{2}n_t(n_t-1) - Jn_tn_{sv} + E_{sv}$. The requirement $E_{total}^A \leq E_{total}^B$ means $J\lfloor \frac{n_t}{2} \rfloor - \frac{J}{2}(n_t \bmod 2) + \frac{J}{2}n_t(n_t-1) + Jn_tn_{sv} \leq 0$. Since only the second term is negative, this condition can never be satisfied for any $n_t \geq 2$. Hence, the model can never shatter two points, so $d_{VC} \leq 1$. Since we also have $d_{VC} \geq 1$, we conclude that $d_{VC} = 1$.

Then, we consider the second relaxed model defined in Equation (18). To simplify the derivation, we assume that n_{sv} and n_t are both even numbers. The support vectors can be in the configuration of half spin-up and half spin-down. The spin-up support vectors are one-to-one-coupled to the spin-up test data. The interaction energy is $E_{int}^A = -J_{int}n_{sv}$ and the total energy is $E_{total}^A = E_t^A + E_{int}^A + E_{sv} = J_t\frac{n_t}{2} - J_{int}n_{sv} + E_{sv}$. We also have $E_{total}^B = E_t^B + E_{int}^B + E_{sv} = \frac{-1}{2}J_tn_t(n_t-1) + 0 + E_{sv}$. The requirement $E_{total}^A \leq E_{total}^B$ then implies $n_p \leq \sqrt{\frac{2J_{int}}{J_t}n_{sv}}$. Hence, $d_{VC} \leq \sqrt{\frac{2J_{int}}{J_t}n_{sv}}$. We may also consider that all support vectors are aligned in the same direction. In this case, we have $E_{total}^A = E_t^A + E_{int}^A + E_{sv} = J_t\frac{n_t}{2} + 0 + E_{sv}$ and $E_{total}^B = E_t^B + E_{int}^B + E_{sv} = \frac{-1}{2}J_tn_t(n_t-1) + J_{int}n_{sv} + E_{sv}$; therefore, the upper bound remains the same.

References

1. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **2015**, *56*, 172–185. [CrossRef]
2. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef] [PubMed]
3. Aaronson, S. The learnability of quantum states. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2007**, *463*, 3089–3114. [CrossRef]
4. Dumoulin, V.; Goodfellow, I.J.; Courville, A.; Bengio, Y. On the Challenges of Physical Implementations of RBMs. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14), Quebec City, QC, Canada, 27–31 July 2014; pp. 1199–1205.
5. Romero, J.; Olson, J.P.; Aspuru-Guzik, A. Quantum autoencoders for efficient compression of quantum data. *Quantum Sci. Technol.* **2017**, *2*, 045001. [CrossRef]
6. Crawford, D.; Levit, A.; Ghadermarzy, N.; Oberoi, J.S.; Ronagh, P. Reinforcement Learning Using Quantum Boltzmann Machines. *Quantum Inf. Comput.* **2018**, *18*, 51–74. [CrossRef]
7. Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K. Quantum circuit learning. *Phys. Rev. A* **2018**, *98*, 032309. [CrossRef]
8. Chen, C.C.; Watabe, M.; Shiba, K.; Sogabe, M.; Sakamoto, K.; Sogabe, T. On the expressibility and overfitting of quantum circuit learning. *ACM Trans. Quantum Comput.* **2021**, *2*, 1–24. [CrossRef]
9. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef]
10. Borle, A.; Elfving, V.E.; Lomonaco, S.J. Quantum Approximate Optimization for Hard Problems in Linear Algebra. *arXiv* **2020**, arXiv:2006.15438.
11. Lin, L.; Tong, Y. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum* **2020**, *4*, 361. [CrossRef]
12. Harrow, A.W. Small quantum computers and large classical data sets. *arXiv* **2020**, arXiv:2004.00026.
13. Aaronson, S. Read the fine print. *Nat. Phys.* **2015**, *11*, 291–293. [CrossRef]
14. Nakaji, K.; Yamamoto, N. Quantum semi-supervised generative adversarial network for enhanced data classification. *arXiv* **2020**, arXiv:2010.13727.
15. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
16. Tang, E. A Quantum-Inspired Classical Algorithm for Recommendation Systems. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019), Phoenix, AZ, USA, 23–26 June 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 217–228. [CrossRef]
17. Chia, N.H.; Li, T.; Lin, H.H.; Wang, C. Quantum-inspired sublinear algorithm for solving low-rank semidefinite programming. *arXiv* **2019**, arXiv:1901.03254.
18. Arrazola, J.M.; Delgado, A.; Bardhan, B.R.; Lloyd, S. Quantum-inspired algorithms in practice. *Quantum* **2020**, *4*, 307. [CrossRef]
19. Bravyi, S.; Kliesch, A.; Koenig, R.; Tang, E. Hybrid quantum-classical algorithms for approximate graph coloring. *arXiv* **2020**, arXiv:2011.13420.
20. Chia, N.H.; Gilyén, A.; Li, T.; Lin, H.H.; Tang, E.; Wang, C. Sampling-Based Sublinear Low-Rank Matrix Arithmetic Framework for Dequantizing Quantum Machine Learning. In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020), Chicago, IL, USA, 22–26 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 387–400. [CrossRef]
21. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
22. Chang, C.C.; Lin, C.J. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–27. [CrossRef]
23. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
24. Bordes, A.; Ertekin, S.; Weston, J.; Bottou, L. Fast Kernel Classifiers with Online and Active Learning. *J. Mach. Learn. Res.* **2005**, *6*, 1579–1619.
25. Wu, X.; Zuo, W.; Lin, L.; Jia, W.; Zhang, D. F-SVM: Combination of Feature Transformation and SVM Learning via Convex Relaxation. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5185–5199. [CrossRef]
26. Rizwan, A.; Iqbal, N.; Ahmad, R.; Kim, D.H. WR-SVM Model Based on the Margin Radius Approach for Solving the Minimum Enclosing Ball Problem in Support Vector Machine Classification. *Appl. Sci.* **2021**, *11*, 4657. [CrossRef]
27. Rebentrost, P.; Mohseni, M.; Lloyd, S. Quantum Support Vector Machine for Big Data Classification. *Phys. Rev. Lett.* **2014**, *113*, 130503. [CrossRef]
28. Schuld, M.; Fingerhuth, M.; Petruccione, F. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhys. Lett.)* **2017**, *119*, 60002. [CrossRef]
29. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [CrossRef]
30. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, USA, 2–4 August 1996; pp. 226–231.

31. Deutsch, D.; Jozsa, R. Rapid solution of problems by quantum computation. *Proc. R. Soc. Lond. Ser. A Math. Phys. Sci.* **1992**, *439*, 553–558. [CrossRef]
32. Rasmussen, S.E.; Groenland, K.; Gerritsma, R.; Schoutens, K.; Zinner, N.T. Single-step implementation of high-fidelity *n*-bit Toffoli gates. *Phys. Rev. A* **2020**, *101*, 022308. [CrossRef]
33. Farhi, E.; Goldstone, J.; Gutmann, S.; Sipser, M. Quantum Computation by Adiabatic Evolution. *arXiv* **2000**, arXiv:quant-ph/quant-ph/0001106.
34. Kadowaki, T.; Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **1998**, *58*, 5355–5363. [CrossRef]
35. Bishop, C.M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*; Springer: Berlin/Heidelberg, Germany, 2006.
36. Giri, K.; Biswas, T.K.; Sarkar, P. ECR-DBSCAN: An improved DBSCAN based on computational geometry. *Mach. Learn. Appl.* **2021**, *6*, 100148. [CrossRef]