Quantum Information and Computation, Vol. 19, No. 7&8 (2019) 0609–0625

QUANTUM SPEEDUP OF TRAINING RADIAL BASIS FUNCTION NETWORKS

CHANGPENG SHAO[a]
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Received April 5, 2019
Revised June 12, 2019

Radial basis function (RBF) network is a simple but useful neural network model that contains wide applications in machine learning. The training of an RBF network reduces to solve a linear system, which is time consuming classically. Based on HHL algorithm, we propose two quantum algorithms to train RBF networks. To apply the HHL algorithm, we choose using the Hamiltonian simulation algorithm proposed in [P. Rebentrost, A. Steffens, I. Marvian and S. Lloyd, Phys. Rev. A 97, 012327, 2018]. However, to use this result, an oracle to query the entries of the matrix of the network should be constructed. We apply the amplitude estimation technique to build this oracle. The final results indicate that if the centers of the RBF network are the training samples, then the quantum computer achieves exponential speedup at the number and the dimension of training samples over the classical computer; if the centers are determined by the K-means algorithm, then the quantum computer achieves quadratic speedup at the number of samples and exponential speedup at the dimension of samples.

Keywords: quantum algorithm, quantum machine learning, radial basis function network

Communicated by: R Jozsa & M Mosca

1 Introduction

1.1 Backgrounds

One goal of machine learning algorithms is to find and study general types of relations (for example clusters, principal components, correlations, classifications) of data. In many cases, it is not straightforward to find these relations in the input space. Kernel method is a commonly used technique to simplify the analysis of data by applying some nonlinear function (called feature map) to map the data into a higher dimensional space (called feature space). For instance, by Cover's theorem [1], a complex pattern-classification problem is more likely to be linearly separable when casted into a high-dimensional space nonlinearly.

Kernel method has been widely used in many machine learning algorithms, such as support vector machines (SVM) [2], principal components analysis (PCA) [3], radial basis function networks (RBF networks) [4], Gaussian process [5], kernel perceptron [6], and so on. As a result, the performance and application scope of these machine learning algorithms are highly increased. Recently, the relationship between feature map, kernel method and quantum computing has been investigated in [7]. It shows that quantum computing and kernel method share certain similarities. Therefore, we believe that quantum computer can further improve

[a]cpshao@amss.ac.cn

the efficiency of the above machine learning algorithms. In this paper, we will focus on the speedup of training RBF networks.

Before that, we discuss a little more about the idea of kernel method. Denote the input data as $\mathbf{x}$ and the feature map as φ, then by representer theorem [8], the output of the algorithm studied behind the idea of kernel method often appears in the form $\mathbf{w} \cdot \varphi(\mathbf{x})$ for some weight vector $\mathbf{w}$ that need to be learned. Sometimes, a bias b is introduced so that the output becomes $b + \mathbf{w} \cdot \varphi(\mathbf{x})$.

Training in machine learning is often accomplished in a supervised approach by providing a lot of samples $\mathbf{x}^{(t)} \in \mathbf{R}^N$, where $t = 1, \ldots, M$. Each sample $\mathbf{x}^{(t)}$ has a desired output $r^{(t)} \in \mathbf{R}$, which is given in advance. When receiving enough samples, the weight $\mathbf{w}$ is learned by minimizing a cost function. A commonly used one is the mean square error $\frac{1}{M}\sum_{t=1}^{M}(\mathbf{w} \cdot \varphi(\mathbf{x}^{(t)}) - r^{(t)})^2$. This corresponds to a least square problem. It is equivalent to solve the following linear system

$$A^\dagger A\mathbf{w} = A^\dagger \mathbf{r}, \tag{1.1}$$

where the t-th column of $A^\dagger$ is $\varphi(\mathbf{x}^{(t)})$, and the column vector $\mathbf{r}$ is composed of $r^{(t)}$.

In quantum computing, people have already discovered many efficient quantum linear solvers, such as HHL [9] and its generalizations [10–12], SVE [13,14], blocking encoding [15,16], etc. One natural question is that are these quantum linear solvers still efficient in solving the linear system (1.1)? It is generally not easy to answer this question since the structure of the linear system can be complicate so that the assumptions of quantum linear solvers may not hold. However, for the RBF networks, we will provide an affirmative answer.

When φ is simple, such as $\varphi(\mathbf{x}^{(t)}) = \mathbf{x}^{(t)}$, then under certain assumptions about the input samples (e.g., qRAM), the linear system (1.1) can be solved efficiently [17–19]. An important choice is $\varphi(\mathbf{x}^{(t)}) = (\mathbf{x}^{(1)} \cdot \mathbf{x}^{(t)}, \ldots, \mathbf{x}^{(M)} \cdot \mathbf{x}^{(t)})$, which has been used in studying least-squares SVM. By applying the Hamiltonian simulation technique of density operators [20], least-squares SVM for big data classification can be implemented efficiently in a quantum computer [21]. More complicated cases with the j-th entry of $\varphi(\mathbf{x}^{(t)})$ equals $(\mathbf{x}^{(j)} \cdot \mathbf{x}^{(t)})^l$ can also be implemented efficiently in a quantum computer by the same technique, where $l \in \mathbf{N}$ is fixed.

Note that quantum linear solvers are mainly based on quantum singular value decomposition, which means that they actually find the least square solution of linear systems. Therefore, to solve (1.1) in a quantum computer, it is equivalent to apply quantum linear solvers to solve

$$A\mathbf{w} = \mathbf{r}. \tag{1.2}$$

1.2 Radial basis function networks

In machine learning, one most commonly used feature map is *radial basis function* (RBF) [22]. A function ψ is called a RBF if there is a real valued function ϕ and a vector $\mathbf{c}$ such that $\psi(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$, where $\|\cdot\|$ is the Euclidian norm. This means a RBF only depends on the distance from a fixed point. If the feature map is a RBF, then the j-th entry of $\varphi(\mathbf{x}^{(t)})$ is $\varphi_j(\mathbf{x}^{(t)}) = \phi_j(\|\mathbf{x}^{(t)} - \mathbf{m}^{(j)}\|)$, where $\{\phi_j\}_j$ are real valued functions. The vectors $\{\mathbf{m}^{(j)} \in \mathbf{R}^N\}_j$ are called the *centers* of φ. To discover the features of the data coherently, for any j we often choose $\phi_j = \phi$ for some fixed real valued function ϕ.

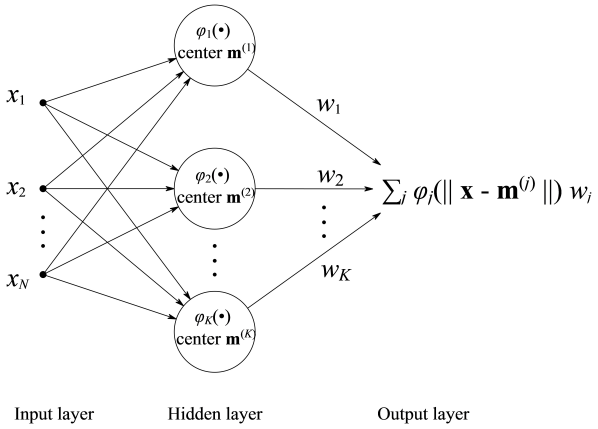


Fig. 1. The structure of RBF network

RBF network is a special type of neural network with only one hidden layer (see figure 1). It was first formulated in 1988 by Broomhead and Lowe [23]. The basic neural computation rule of RBF networks builds on the idea of feature map: Fix a real-valued function ϕ and K centers $\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(K)}$. For any data $\mathbf{x}$ in the input layer, it is mapped as a K-dimensional vector $(\phi(\|\mathbf{x} - \mathbf{m}^{(1)}\|), \cdots, \phi(\|\mathbf{x} - \mathbf{m}^{(K)}\|))$ in the hidden layer by ϕ. Finally, the output of the RBF network reads in the form $\sum_j w_j \phi(\|\mathbf{x} - \mathbf{m}^{(j)}\|)$, where $w_1, \ldots, w_K$ are the weights need to be learned.

In many classification problems, RBF network exhibits nearly identical performance as SVM [24]. RBF networks are continuously increasing their popularity due to a number of advantages compared to other types of neural networks. This includes better approximation capabilities and simple network structures.

Similar to many other machine learning algorithms, RBF networks have wide applications in function approximation, classification and system control [4,25–27]. One simple application of RBF network is the XOR problem. In the XOR problem, we need to find a boundary that can separate two patterns: {(1, 1), (0, 0)} and {(1, 0), (0, 1)}. The two patterns are not linear separable in the input space (see figure 2 (left)). If we choose $K = 2$ and

$$\begin{aligned} \varphi_1(\mathbf{x}) &= \exp(-\|\mathbf{x} - (1,1)\|^2), \\ \varphi_2(\mathbf{x}) &= \exp(-\|\mathbf{x}\|^2). \end{aligned} \tag{1.3}$$

Then the images of {(1, 1), (0, 0)} and {(1, 0), (0, 1)} in the feature space are {(1, 0.135), (0.135, 1)} and {(0.368, 0.368), (0.368, 0.368)} respectively. The pattern in the feature space is linear separable (see figure 2 (right)). In this example, there is no increase in the dimension of the feature space compared with the input space.

The training of a RBF network has two steps:

1. Compute the centers of the RBFs. There are two choices of the centers:
   Type I. The training samples.
   Type II. The centers of the clusters of the training samples found by the K-means.
2. Compute the weights by solving the linear system (1.1).

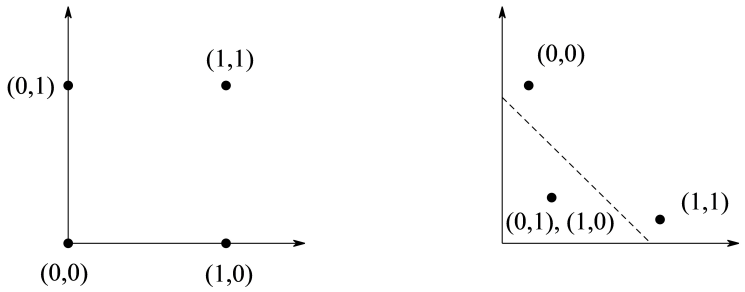
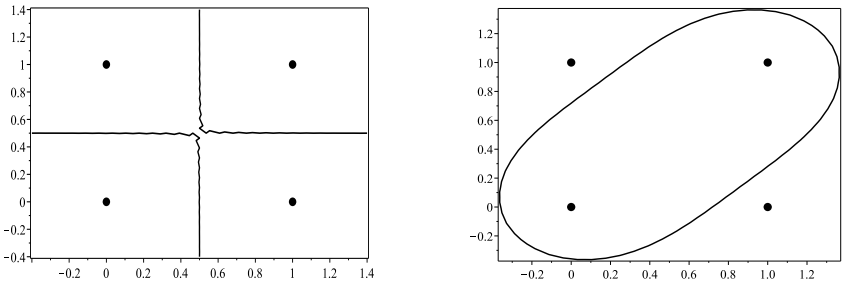


Fig. 2. Apply RBF network to solve XOR problem.

When we obtain $\mathbf{w} = (w_1, \ldots, w_K)$ by solving the linear system (1.1), we can apply RBF network to do classification on new data $\mathbf{x}$ by computing the value $\sum_i w_i \phi(\|\mathbf{x} - \mathbf{m}^{(i)}\|)$. For the classification problem with two patterns, $\sum_i w_i \phi(\|\mathbf{x} - \mathbf{m}^{(i)}\|) = 0$ defines the decision boundary. For instance, in the XOR problem, we can set the labels of $\{(1, 1), (0, 0)\}$ and $\{(1, 0), (0, 1)\}$ are 1 and -1, respectively. By taking the centers as the samples themselves and $K = 4$, then the weight after training is $\mathbf{w} = 2.503(1, -1, -1, 1)^T$. The decision boundary is shown in figure 3(a). In comparison, if we still use the choice (1.3), but add a bias b, then the trained weight is $\mathbf{w} = (5.012, 5.012)$ and the bias is $b = -4.689$. The decision boundary in the input space is shown in figure 3(b).



(a) (b)

Fig. 3. Decision boundary of XOR problem.

Theoretical investigations and practical results [28] suggest that the choice of the nonlinear function ϕ is not crucial to the performance of the RBF network. Therefore, in this paper, we will focus on the commonly used one: the Gaussian type of RBF function

$$\phi(r) = \exp(-r^2/2\sigma^2), \tag{1.4}$$

where $\sigma > 0$ is the width of ϕ.

Although the RBF network has a simple structure, solving linear system to train it is still a time consuming process for a classical computer. In this paper, we will show how to apply quantum computer to speed up the training of RBF networks. For a RBF network, usually the number M of samples is large. Hence, to train a RBF network in a quantum computer, it is important to achieve speedup at M. This will be the main concern of this paper. Because

of this, in this paper we will only focus on using HHL algorithm to solve the linear system (1.2). Other quantum linear solvers [10–16] may also be used to improve our algorithms on other parameters, such as the condition number and the precision. However, the dependence on M cannot be improved (see remark 3).

In this following, when we say HHL algorithm, it not only indicates the original one to solve linear systems, but also indicates its generalizations to do basic linear algebraic operations, such as matrix-vector multiplication.

1.3 Basic idea of the quantum algorithms

For type I, the centers are simple. In this case, we will propose two quantum algorithms to solve the linear system (1.2) by HHL algorithm. Besides the influence of qRAM, the other difficulty to use HHL algorithm is the Hamiltonian simulation. Thus, Hamiltonian simulation will the main focus of our algorithms to speed up the training of RBF network.

The first algorithm is based on the Hamiltonian simulation technique of density matrix [20]. To apply this technique, we need to represent A as a density matrix. This is achieved by considering the coherent states of samples. Coherent states are known in the field of quantum optics as a description of light modes. Formally, they are superpositions of *Fock states*, which are basis states from an infinite-dimensional discrete basis $\{|0\rangle, |1\rangle, \ldots\}$. The final result shows that the quantum computer can achieve an exponential speedup at M, but no speedup at N (see theorem 1) over the classical computer. The other one (see theorem 3) achieves exponential speedup both at M and N by applying the Hamiltonian simulation technique proposed in [29]. However, to apply the result of [29], we need to build an oracle to query the entries of A. We will construct this oracle by amplitude estimation technique.

For type II, we need to apply quantum K-means algorithm to find the centers first. K-means is a unsupervised algorithm to achieve data clustering. In [30], Lloyd et al. apply adiabatic method to give a quantum K-means algorithm. This algorithm generates a quantum state, in which the first register stores the centers of clusters and the second register stores the samples belong to their cluster. The complexity to get this quantum state to accuracy ϵ is no greater than $O(\epsilon^{-1}K\log(MN))$, and is $O(\epsilon^{-1}\log(KMN))$ when the centers are relatively well separated.

When getting the centers, the next task is to solve the linear system (1.2) by HHL algorithm. The first algorithm used in type I does not work for type II since the matrix A is generally not Hermitian. Thus, we cannot represent it as a density matrix. However, the second proposal in type I still works. Similarly, we will use amplitude estimation to construct the required oracle to implement Hamiltonian simulation. Different from type I, the oracle here is much harder to build in that the result of quantum K-means algorithm is just a quantum state, not the classical information of the centers. The final result (see theorem 2) indicates that the quantum computer achieves quadratic speedup at M and exponential speedup at N to train RBF network over the classical computer.

2 Training RBF networks with type I choice of the centers

2.1 Training the network

In type I, the centers are the training sample, thus the matrix $A = (a_{ij})_{M\times M}$ satisfies

$$a_{ij} = \exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma^2). \tag{2.1}$$

In this paper, we assume that the information $\|\mathbf{x}^{(i)}\|$ and $|\mathbf{x}^{(i)}\rangle$ are given in advance, such as by qRAM [31] or binary tree data structure [32].

Let σ be a fixed real number, then any real number r defines a *coherent state* in the *Fock basis* [33, 34]

$$|\psi_r\rangle := e^{-r^2/2\sigma^2} \sum_{n=0}^{\infty} \frac{(r/\sigma)^n}{\sqrt{n!}} |n\rangle. \tag{2.2}$$

Assume that $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbf{R}^N$, then we can similarly define the coherent state of $\mathbf{x}$ as

$$|\psi_{\mathbf{x}}\rangle := |\psi_{x_1}\rangle \otimes \cdots \otimes |\psi_{x_N}\rangle. \tag{2.3}$$

It is easy to verify that for any two real vectors $\mathbf{x}, \mathbf{y}$, the following identity holds

$$\langle \psi_{\mathbf{x}} | \psi_{\mathbf{y}} \rangle = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2). \tag{2.4}$$

To solve the linear system (1.2) by HHL algorithm, we need efficient quantum algorithms to exponentiate A. In [20], Lloyd et al. proposed such a quantum algorithm if we can represent A in the density matrix form. The basic idea is as follows: Let ρ be the density matrix we need to exponentiate. Let $S = \sum_{i,j} |i,j\rangle\langle j,i|$ be the swap operator and τ be another density matrix, then simple computation shows that $\mathrm{Tr}_1\{e^{-\mathbf{i}S\Delta t} \rho \otimes \tau\, e^{\mathbf{i}S\Delta t}\} = \tau - \mathbf{i}[\rho, \tau]\Delta t + O((\Delta t)^2) = e^{-\mathbf{i}\rho\Delta t} \tau e^{\mathbf{i}\rho\Delta t} + O((\Delta t)^2)$. Repeating this procedure with m copies of ρ allows one to approximate $e^{-\mathbf{i}\rho n\Delta t} \tau e^{\mathbf{i}\rho n\Delta t}$. If the error to approximate $e^{-\mathbf{i}\rho\Delta t}$ is ϵ_0, which equals $(\Delta t)^2$, then the error to approximate $e^{-\mathbf{i}\rho t}$ is $m\epsilon_0 = t^2/m$. To make this error small in size ϵ, we can choose $m = O(t^2/\epsilon)$. Therefore, $O(t^2/\epsilon)$ copies of ρ will produce an ϵ-approximation of $e^{-\mathbf{i}\rho t}$. For simulating density operators, this result is already optimal [35].

To represent A as a density matrix, we first prepare the following quantum state

$$|\Phi\rangle = \frac{1}{\sqrt{M}} \sum_{i=1}^{M} |i\rangle |\psi_{\mathbf{x}^{(i)}}\rangle. \tag{2.5}$$

The corresponding density matrix equals

$$|\Phi\rangle\langle\Phi| = \frac{1}{M} \sum_{i,j=1}^{M} |i\rangle |\psi_{\mathbf{x}^{(i)}}\rangle\langle j| \langle \psi_{\mathbf{x}^{(j)}}|. \tag{2.6}$$

Computing the partial trace with respect to the coherent state, by equation (2.4), we obtain A as a quantum density matrix

$$\rho = \mathrm{Tr}_2 |\Phi\rangle\langle\Phi| = \frac{1}{M} \sum_{i,j=1}^{M} \langle \psi_{\mathbf{x}^{(i)}} | \psi_{\mathbf{x}^{(j)}} \rangle |i\rangle\langle j| = \frac{1}{M} \sum_{i,j=1}^{M} a_{ij} |i\rangle\langle j| = \frac{A}{M}. \tag{2.7}$$

Now we have represented A as a density operator, thus we can use the Hamiltonian simulation technique [20] to simulate $e^{-\mathbf{i}At/M}$. To apply HHL algorithm to solve the linear system (1.2), we assume that the eigenvalue decomposition of $A = \sum_i \sigma_i |u_i\rangle\langle u_i|$. We can formally rewrite $|\mathbf{r}\rangle = \sum_i \beta_i |u_i\rangle$, then by quantum phase estimation, we obtain $\sum_i \beta_i |u_i\rangle |\sigma_i / M\rangle$. Finally, apply control rotation and undo the quantum phase estimation, we get $\sum_i \beta_i \sigma_i^{-1} |u_i\rangle |0\rangle +$

$\sum_i \beta_i \sqrt{1-\sigma_i^{-2}}|u_i\rangle|1\rangle$. Measuring the second register, if we get $|0\rangle$, then we will get the quantum state of the solution of the linear system (1.2). The above is the basic procedure of HHL algorithm.

The complexity analysis is similar to that of HHL algorithm. Let κ be the condition number of A. If the Hamiltonian simulation is efficient, then the complexity of the original HHL algorithm is $O(\kappa^2(\log M)/\epsilon)$. It is quadratic at the condition number κ, one comes from the error to estimate the inverse of eigenvalues, which further relates to the complexity of Hamiltonian simulation. The other one comes from the success probability. For sparse matrices, the complexity of Hamiltonian simulation linearly depends on t. As analyzed in HHL algorithm, to make the error of estimating the inverse of eigenvalues small in size ϵ, the evolution time t in Hamiltonian simulation should be chosen as κ/ϵ. However, here the Hamiltonian simulation has complexity $O(t^2/\epsilon)$. This will lead to $O(\kappa^2/\epsilon^3)$ in the complexity if we choose $t = \kappa/\epsilon$. The success probability is $\sum_i |\beta_i \sigma_i^{-1}|^2 \geq 1/\kappa^2$, here we make the same assumption that $1/\kappa \leq |\sigma_i| \leq 1$ as HHL algorithm. It is achieved by performing a suitable scaling on the original linear system. Thus, by amplitude amplification, the algorithm should be repeated $O(\kappa)$ times. Therefore, the complexity to get the quantum state of the solution to accuracy ϵ of the linear system (1.2) is $O(\kappa^3 N(\log M)/\epsilon^3)$, where the parameter N comes from the preparation of N coherent states in equation (2.3). Concluding the above analysis, we have

Theorem 1 *Let $\{\mathbf{x}^{(t)} \in \mathbf{R}^N : t = 1, \ldots, M\}$ be the training samples of the RBF network. If the centers are the training samples, then the quantum state of the weight of the RBF network can be obtained in time*

$$O(\kappa^3 N(\log M)/\epsilon^3), \tag{2.8}$$

where ϵ is the precision and κ is the condition number of $A = (\exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma^2))_{M\times M}$.

Classically, if we apply conjugate gradient method to solve the linear system (1.2), then the complexity is $O(MN^2 + sM\sqrt{\kappa}\log(1/\epsilon))$, where s is the sparsity of A and $O(MN^2)$ is used to compute all the entries of A. Therefore, if the condition number of A is not large, then quantum algorithm is exponentially faster at M over the classical training algorithm. Although there are some other classical algorithms, such as least mean square or recursive least-squares algorithm [4], to solve the least square problem, the complexity is still at least linear at M and N.

The complexity of the quantum training algorithm obtained in theorem 1 is linear at the dimension N of the samples. In the end of section 3, we will show that exponential speedup at N is also achievable.

2.2 Prediction on the new data

In this subsection, we show how to do data prediction with the quantum state of the solution of the linear system (1.2). The prediction on a new data $\mathbf{x}$ can be obtained by applying swap test [36] to estimate the inner product between $|\mathbf{w}\rangle$ and $|\varphi(\mathbf{x})\rangle$. This requires efficient preparation of the state $|\varphi(\mathbf{x})\rangle$. One naive idea is to evaluate all the M entries of $\varphi(\mathbf{x})$, then prepare $|\varphi(\mathbf{x})\rangle$. However, this costs at least $O(M)$ operations, which means the exponential speedup at M obtained in theorem 1 will disappear.

Because of quantum parallelism, there actually exists an efficient approach to prepare the state $|\varphi(\mathbf{x})\rangle$. The basic idea is as follows: For simplicity, denote

$$|\phi_t\rangle = \frac{1}{\sqrt{\|\mathbf{x}\|^2 + \|\mathbf{x}^{(t)}\|^2}}\Big(\|\mathbf{x}\|\,|+\rangle|\mathbf{x}\rangle - \|\mathbf{x}^{(t)}\|\,|-\rangle|\mathbf{x}^{(t)}\rangle\Big). \tag{2.9}$$

Step 1. Set the initial state as $\frac{1}{\sqrt{M}}\sum_{t=1}^{M}|t\rangle$.

Step 2. We can generate $\frac{1}{\sqrt{M}}\sum_{t=1}^{M}|t\rangle|\phi_t\rangle$ by viewing $|t\rangle$ as a control qubit in the initial state. The amplitude of $|0\rangle$ of the first register of $|\phi_t\rangle$ equals $\|\mathbf{x}-\mathbf{x}^{(t)}\|/\sqrt{2(\|\mathbf{x}\|^2+\|\mathbf{x}^{(t)}\|^2)}$. By amplitude estimation [37] (see appendix 1 for a brief introduction), we will obtain an approximation of $\|\mathbf{x}-\mathbf{x}^{(t)}\|$. Store this approximation into an ancilla register, then we obtain

$$\frac{1}{\sqrt{M}}\sum_{t=1}^{M}|t\rangle|\phi_t\rangle\big|\|\mathbf{x}-\mathbf{x}^{(t)}\|\big\rangle. \tag{2.10}$$

Step 3. Apply $I\otimes I\otimes U_\phi$ to the above state to get

$$\frac{1}{\sqrt{M}}\sum_{t=1}^{M}|t\rangle|\phi_t\rangle\big|\|\mathbf{x}-\mathbf{x}^{(t)}\|\big\rangle\big|\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)\big\rangle. \tag{2.11}$$

Step 4. Apply control rotation to put the value $\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)$ into the coefficient of $|t\rangle$. This yields

$$\begin{aligned}&\frac{1}{\sqrt{M}}\sum_{t=1}^{M}|t\rangle|\phi_t\rangle\big|\|\mathbf{x}-\mathbf{x}^{(t)}\|\big\rangle\big|\phi[\|\mathbf{x}-\mathbf{x}^{(t)}\|)\big\rangle\left[\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)\,|0\rangle+\sqrt{1-\phi^2(\|\mathbf{x}-\mathbf{x}^{(t)}\|)}\,|1\rangle\right]\\ =\ &\frac{1}{\sqrt{M}}\sum_{t=1}^{M}\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)|t\rangle|\phi_t\rangle\big|\|\mathbf{x}-\mathbf{x}^{(t)}\|\big\rangle\big|\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)\big\rangle|0\rangle+|0^\perp\rangle,\end{aligned} \tag{2.12}$$

where $|0^\perp\rangle$ refers to the quantum state that is orthogonal to the first part.

Step 5. Undo **Step 2, 3**, then we get

$$|\widetilde{\varphi(\mathbf{x})}\rangle = \frac{1}{\sqrt{M}}\sum_{t=1}^{M}\phi(\|\mathbf{x}-\mathbf{x}^{(t)}\|)|t\rangle|0\rangle+|0^\perp\rangle. \tag{2.13}$$

The quantum information of $\varphi(\mathbf{x})$ is stored in the first part of the above quantum state. And the inner product between $|\mathbf{w}\rangle$ and $|\varphi(\mathbf{x})\rangle$ can be obtained by estimating the inner product between $|\mathbf{w}\rangle$ and $|\widetilde{\varphi(\mathbf{x})}\rangle$. In the above procedure, the amplitude estimation is implemented in parallel in the quantum computer, thus the dependence of the complexity on M is still logarithm. In **Step 2**, the complexity to approximate $\|\mathbf{x}-\mathbf{x}^{(t)}\|$ will be affected by $\sqrt{2(\|\mathbf{x}\|^2+\|\mathbf{x}^{(t)}\|^2)}$. Thus the complexity of the above procedure to get $|\widetilde{\varphi(\mathbf{x})}\rangle$ will also be affected by $\sqrt{2(\|\mathbf{x}\|^2+\|\mathbf{x}^{(t)}\|^2)}$. This seems unavoidable; however, the whole procedure is satisfied since the complexity is still logarithmic at M.

3 Training RBF networks with type II choice of the centers

Choosing the sample themselves as the centers is a simple idea used in training RBF network classically. However, it works not so well when a large amount of samples are used. On one hand, such a choice will increase the computing load for a classical computer. More importantly, it is not suitable to manipulate samples with noises. Training based on noisy data could lead to misleading results. One practical modification is to choose some other K centers that can preserve the local properties of the data. Generally, this is measured by distance. Samples with small relative distances are supposed to share similar features, and should be treated equally. K-means [38] is one such unsupervised machine learning algorithm that is used to find the K centers in RBF network. It is simple in implementation and effective in performance.

3.1 The K-means algorithm and its quantum speedup

Assume that $\{\mathbf{x}^{(t)} \in \mathbf{R}^N : t = 1, \ldots, M\}$ is a given set of samples, and K is a nonnegative integer. Clustering is an unsupervised learning algorithm that aims to find K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ based on certain criterions, such that samples with similar features are assigned into the same cluster. The K-means is an algorithm that aims at finding K vectors $\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(K)} \in \mathbf{R}^N$, called the means (or centers) of the clusters, such that they minimize

$$\sum_{s=1}^{K} \sum_{t=1}^{M} c_{st} \|\mathbf{x}^{(t)} - \mathbf{m}^{(s)}\|^2, \tag{3.1}$$

where

$$c_{st} := \begin{cases} 1, & \text{if } s = \arg\min_r \|\mathbf{x}^{(t)} - \mathbf{m}^{(r)}\|; \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The K-means algorithm goes as follows: (1). Randomly choose K different vectors $\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(K)}$, and set $\mathcal{C}_1 = \cdots = \mathcal{C}_K = \emptyset$. (2). For $1 \leq t \leq M$, assign $\mathbf{x}^{(t)}$ into the s-th cluster $\mathcal{C}_s$ if $s = \arg\min_r \|\mathbf{x}^{(t)} - \mathbf{m}^{(r)}\|$. (3). For $1 \leq s \leq K$, define $\mathbf{m}^{(s)} = \frac{1}{\#(\mathcal{C}_s)} \sum_{\mathbf{x}^{(t)} \in \mathcal{C}_s} \mathbf{x}^{(t)}$. Repeat step 2, 3 until converges.

It is not hard to see that each iteration of K-means algorithm needs $O(KMN)$ operations. The algorithm converges if the change in the centers during two adjacent iterations is sufficiently small. More precisely, denote the centers obtained in the l-th step of iteration as $\mathbf{m}_l^{(1)}, \ldots, \mathbf{m}_l^{(K)}$, then we say the algorithm converges at l-th iteration if

$$\frac{1}{K} \sum_{s=1}^{K} \|\mathbf{m}_l^{(s)} - \mathbf{m}_{l+1}^{(s)}\| \leq \delta$$

for some small threshold δ.

Therefore, in type II the complexity of classical algorithms to train RBF network is polynomial at K, M, N. For the RBF network, usually the data are mapped into a higher dimensional space, thus we have $N < K < M$. This means achieving speedup at M is still the most important thing to train a RBF network in a quantum computer. Next, we show how to speed up the training of RBF network in the quantum computer.

In [30], Lloyd et al. proposed a quantum K-means algorithm based on adiabatic algorithm. With the notation given in equation (3.2), the result they obtain is the following quantum state

$$|\chi\rangle = \frac{1}{\sqrt{M}} \sum_{s,t} c_{st}|t\rangle|s\rangle = \frac{1}{\sqrt{M}} \sum_{j=1}^{K} |j\rangle \sum_{\mathbf{x}^{(i)} \in \mathcal{C}_j} |i\rangle. \tag{3.3}$$

The complexity to get $|\chi\rangle$ is $O(\epsilon^{-1} K \log(MN))$ or $O(\epsilon^{-1} \log(KMN))$ when the centers are relatively well separated. In the following, we will simply denote the complexity to obtain the above state as $O(T_K)$.

Remark 1 In [39], Kerenidis et al. proposed the so called q-means algorithm, an extension of classical δ-K-means algorithm. The output of this algorithm is the classical information of the centers, which can be used directly to solve real-world problems. The complexity is polynomial at N and K but polylogarithmic at M. If N, K are not large, we can also use this result to train RBF network.

3.2 Training the RBF networks

In the subsection, we show how to solve the linear system (1.2) based on HHL algorithm and the quantum state $|\chi\rangle$ defined in equation (3.3).

The idea used in section 2 does not work here with at least three reasons: (i). The method used in section 2 only works for density matrices. However, A is generally not Hermitian in type II. Thus it cannot be represented as a density matrix. (ii). Even though we can extend A into a Hermitian matrix $\widetilde{A} = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$, it does not correspond to a density matrix. (iii). Although $A^\dagger A$ corresponds to a density matrix, it is not easy to prepare the quantum state $A^\dagger \mathbf{r}$ of the right side of the linear system (1.1).

In the following, to apply HHL algorithm to solve the linear system (1.1) we need a new Hamiltonian technique. Note that to solve the linear system (1.1) by HHL algorithm, we need to solve the following linear system

$$\widetilde{A} \begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ 0 \end{bmatrix}.$$

Based on the above analysis, this requires a new method to exponentiate $\widetilde{A}$.

Similar to the idea of [20], Rebentrost et al. in [29] proposed a new method to exponentiate general Hermitian matrices. The basic idea can be stated as follows: Let $H = (H_{ij})$ be an $M \times M$ Hermitian matrix, and $S_H = \sum_{i,j} H_{ij}|i,j\rangle\langle j,i|$. Let $\rho = \frac{1}{M}\sum_{i,j}|i\rangle\langle j|$ and τ be a fixed density matrix. Then $\mathrm{Tr}_1\{e^{-\mathbf{i}S_H\Delta t}\rho \otimes \tau\, e^{\mathbf{i}S_H\Delta t}\} = \tau - \mathbf{i}[\rho,\tau]\frac{\Delta t}{M} + O((\Delta t)^2\|H\|^2_{\max}) = e^{-\mathbf{i}H\Delta t/M}\tau e^{\mathbf{i}H\Delta t/M} + O((\Delta t)^2\|H\|^2_{\max})$, where $\|H\|_{\max} = \max_{i,j}|H_{ij}|$. Similar analysis in section 2, implementing this procedure $O(t^2\|H\|^2_{\max}/\epsilon)$ steps, one can simulate $\exp(\mathbf{i}tH/M)$ to accuracy ϵ. The efficiency of this procedure depends on the efficiency of Hamiltonian simulation of the sparse matrix S_H.

In our situation, to obtain the Hamiltonian simulation of the sparse matrix $S_{\widetilde{A}}$ by the method [40], we need to build an oracle to query the entries of A:

$$\mathcal{O} : |i,j\rangle|0\rangle \mapsto |i,j\rangle|a_{ij}\rangle, \tag{3.4}$$

and an oracle to indicate the positions of nonzero entries of $\widetilde{A}$. The second oracle is easy to implement in that $\exp(-\|\mathbf{x}^{(i)}-\mathbf{m}^{(j)}\|^2/2\sigma^2)$ is nonzero for all i,j. In the following, we show how to get the first oracle (3.4).

Denote the matrix generated by samples as $X=(\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(M)})$, which is called the *sample matrix*. It is an $N\times M$ matrix. For convenience, we assume that the singular values of X are smaller than 1. Otherwise, we can make a suitable scaling on the samples. Apply any quantum linear algebraic technique, such as HHL algorithm, to multiply X on the second register of $|\chi\rangle$, then we get

$$\begin{aligned}|\widetilde{\chi}\rangle &= \frac{1}{\sqrt{M}}\sum_{j=1}^{K}|j\rangle\sum_{\mathbf{x}^{(i)}\in\mathcal{C}_j}\|\mathbf{x}^{(i)}\|\,|\mathbf{x}^{(i)}\rangle|0\rangle+|0^{\perp}\rangle\\ &= \frac{1}{\sqrt{M}}\sum_{j=1}^{K}\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|\,|j\rangle|\mathbf{m}^{(j)}\rangle|0\rangle+|0^{\perp}\rangle,\end{aligned}\tag{3.5}$$

where $|0^{\perp}\rangle$ refers to an state that is orthogonal to the first part.

Remark 2 Let H be a Hermitian matrix such that its singular values are smaller than 1. Assume that the singular value decomposition of H is $\sum\sigma_j|u_j\rangle\langle u_j|$ and $|b\rangle=\sum\beta_j|u_j\rangle$, then by quantum phase estimation, we can obtain $\sum\beta_j|u_j\rangle|\tilde{\sigma}_j\rangle$ by choosing $|b\rangle$ as the initial state, where $\tilde{\sigma}_j$ is an approximation of σ_j. Since $\tilde{\sigma}_j<1$, we can apply control rotation generated by $|\tilde{\sigma}_j\rangle$ to prepare $\sum\beta_j|u_j\rangle|\tilde{\sigma}_j\rangle(\tilde{\sigma}_j|0\rangle+\sqrt{1-\tilde{\sigma}_j^2}|1\rangle)$. Undo quantum phase estimation gives $\sum\beta_j|u_j\rangle(\tilde{\sigma}_j|0\rangle+\sqrt{1-\tilde{\sigma}_j^2}|1\rangle)=H|b\rangle|0\rangle+|0^{\perp}\rangle$. If H is not Hermitian, then we can extend it into a Hermitian matrix similar to $\widetilde{A}$ introduced above. The result does not change. In (3.5), $\sum_{\mathbf{x}^{(i)}\in\mathcal{C}_j}X|i\rangle=\sum_{\mathbf{x}^{(i)}\in\mathcal{C}_j}\|\mathbf{x}^{(i)}\|\,|\mathbf{x}^{(i)}\rangle$. The above analysis shows that the state $|\widetilde{\chi}\rangle$ is well-defined.

Assume that the condition number of X as $\kappa(X)$, then the complexity to get $|\widetilde{\chi}\rangle$ is

$$O(T_K+\epsilon^{-1}\kappa(X)\log(MK)),\tag{3.6}$$

where $O(T_K)$ is the cost to prepare $|\chi\rangle$, and $O(\epsilon^{-1}\kappa(X)\log(MK))$ is the cost to implement quantum matrix multiplication by HHL algorithm. Since we perform no measurement to get $|\widetilde{\chi}\rangle$, the dependence on $\kappa(X)$ is linear here.

Note that $\|\mathbf{x}^{(i)}\|$ is given in advance, thus to evaluate $a_{ij}=\exp(-\|\mathbf{x}^{(i)}-\mathbf{m}^{(j)}\|^2/2\sigma^2)$, we need to estimate $\|\mathbf{m}^{(j)}\|$ and $\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$. These values only relate to inner product of vectors, thus amplitude estimation and swap test are the methods that we can use. The basic idea is:

1. Apply amplitude estimation to estimate the amplitude of $|j\rangle$ of the first register of $|\chi\rangle$ to get get an approximation of $\#(\mathcal{C}_j)$.
2. Apply amplitude estimation to estimate the amplitude of $|j,0\rangle$ in $|\widetilde{\chi}\rangle$ to get an approximate of $\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|$. Together with the result of the first step, we can obtain an approximation of $\|\mathbf{m}^{(j)}\|$.
3. Apply swap test to estimate the inner product between $|\widetilde{\chi}\rangle$ and $|j\rangle|\mathbf{x}^{(i)}\rangle|0\rangle$. This will return an approximation of $\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$.

To achieve speedup in a quantum computer, we have to implement the above procedures in parallel for all i, j. And this is accomplished by the quantum parallelism feature. In the following, we show more detail of constructing the oracle.

Step 1. Set the initial state as $|i, j\rangle$.

Step 2. Prepare the following quantum state

$$|i, j\rangle|\chi\rangle|\widetilde{\chi}\rangle|\widehat{\chi}\rangle, \tag{3.7}$$

where $|\widehat{\chi}\rangle = \frac{1}{\sqrt{2}}\big(|+\rangle|\widetilde{\chi}\rangle + |-\rangle|j\rangle|\mathbf{x}^{(i)}\rangle|0\rangle\big)$.

Step 3. Apply amplitude estimation algorithm respectively to estimate the amplitude of $|j\rangle$ in $|\chi\rangle$, the amplitude of $|j, 0\rangle$ in $|\widetilde{\chi}\rangle$, and the amplitude of the first register $|0\rangle$ in $|\widehat{\chi}\rangle$. These three amplitudes respectively return approximations of $\#(\mathcal{C}_j)$, $\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|$ and $\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$. Instead of returning classical information, we can store them in ancilla registers (see equation (A.2) in appendix 1) to get

$$|i, j\rangle|\chi\rangle|\widetilde{\chi}\rangle|\widehat{\chi}\rangle\big|\#(\mathcal{C}_j)\big\rangle\big|\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|\big\rangle\big|\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle\big\rangle. \tag{3.8}$$

Step 4. Since $\|\mathbf{x}^{(i)}\|$ is known, apply an oracle to query the values in the ancilla registers, we can compute the value $a_{ij} = \exp(-\|\mathbf{x}^{(i)} - \mathbf{m}^{(j)}\|^2/2\sigma^2)$. Thus we get

$$|i, j\rangle|\chi\rangle|\widetilde{\chi}\rangle|\widehat{\chi}\rangle\big|\#(\mathcal{C}_j)\big\rangle\big|\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|\big\rangle\big|\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle\big\rangle|a_{ij}\rangle. \tag{3.9}$$

The oracle used here is generated by some simple functions. To be more exact, let d be a fixed number, then for any a, b, c, where $a \neq 0$, then the oracle used in preparing (3.9) is the composition of the following procedures

$$\begin{aligned}
|a\rangle|b\rangle|c\rangle|0\rangle|0\rangle|0\rangle \quad &\to \quad |a\rangle|b\rangle|c\rangle|\frac{b}{a}\rangle|0\rangle|0\rangle \\
&\to \quad |a\rangle|b\rangle|c\rangle|\frac{b}{a}\rangle|\frac{2bcd}{a}\rangle|0\rangle \\
&\to \quad |a\rangle|b\rangle|c\rangle|\frac{b}{a}\rangle|\frac{2bcd}{a}\rangle|\exp(-(d^2 - \frac{2bcd}{a} + \frac{b^2}{a^2})/2\sigma^2)\rangle \\
&\to \quad |a\rangle|b\rangle|c\rangle|0\rangle|0\rangle|\exp(-(d^2 - \frac{2bcd}{a} + \frac{b^2}{a^2})/2\sigma^2)\rangle.
\end{aligned} \tag{3.10}$$

The last step undoes the first two procedures. In the situation of (3.9), we have $a = \#(\mathcal{C}_j), b = \#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|, c = \langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$ and $d = \|\mathbf{x}^{(i)}\|$.

Step 5. Undo **Step 2, 3**, then we obtain the desired result $|i, j\rangle|a_{ij}\rangle$.

Proposition 1 *The oracle (3.4) can be implemented in time*

$$O\Big(\frac{\sqrt{M}T_K}{\epsilon} + \frac{\sqrt{M}\kappa(X)\log(MK)}{\epsilon^2}\Big). \tag{3.11}$$

Proof. The main cost of above procedure is **Step 3**.

(1). Apply amplitude estimation to estimate the amplitude of $|j\rangle$ of the first register of $|\chi\rangle$, then we get a value α in time $O(T_K/\epsilon_0)$, such that $|\alpha - \#(\mathcal{C}_j)/M| \leq \epsilon_0$. Hence $\sqrt{M}\alpha$ gives an $\epsilon_0\sqrt{M}$-approximation of $\#(\mathcal{C}_j)/\sqrt{M}$. To make the error small in size ϵ, we set $\epsilon_0 = \epsilon/\sqrt{M}$. Therefore, the complexity to get an ϵ-approximation of $\#(\mathcal{C}_j)/\sqrt{M}$ is $O(T_K\sqrt{M}/\epsilon)$.

(2). By amplitude estimation and equation (3.6), we can obtain an ϵ_1-approximation β of the amplitude, which equals $\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|/\sqrt{M}$, of $|j,0\rangle$ in $|\widetilde{\chi}\rangle$ in time

$$O(\epsilon_1^{-1}T_K + \epsilon_1^{-1}\epsilon^{-1}\kappa(X)\log(MK)).$$

Thus, $\sqrt{M}\beta/\#(\mathcal{C}_j)$ returns a $\sqrt{M}\epsilon_1/\#(\mathcal{C}_j)$-approximation of $\|\mathbf{m}^{(j)}\|$. Set $\epsilon_1 = \#(\mathcal{C}_j)\epsilon/\sqrt{M}$, then we obtain an ϵ-approximation of $\|\mathbf{m}^{(j)}\|$. The complexity is

$$O\Big(\frac{\sqrt{M}T_K}{\#(\mathcal{C}_j)\epsilon} + \frac{\sqrt{M}\kappa(X)\log(MK)}{\#(\mathcal{C}_j)\epsilon^2}\Big) = O\Big(\frac{\sqrt{M}T_K}{\epsilon} + \frac{\sqrt{M}\kappa(X)\log(MK)}{\epsilon^2}\Big). \tag{3.12}$$

(3). Apply amplitude estimation to estimate the amplitude of $|0\rangle$ in $|\widehat{\chi}\rangle$, then we obtain an ϵ_2-approximation γ of $\#(\mathcal{C}_j)\|\mathbf{m}^{(j)}\|\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle/\sqrt{M}$ in time $O(\epsilon_2^{-1}T_K+\epsilon_2^{-1}\epsilon^{-1}\kappa(X)\log(MK))$. Hence, $\sqrt{M}\gamma/\#(\mathcal{C}_j)$ gives a $\sqrt{M}\epsilon_2/\#(\mathcal{C}_j)$-approximation $\|\mathbf{m}^{(j)}\|\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$. By setting $\epsilon_2 = \#(\mathcal{C}_j)\epsilon/\sqrt{M}$, we will obtain an ϵ-approximation of $\|\mathbf{m}^{(j)}\|\langle\mathbf{x}^{(i)}|\mathbf{m}^{(j)}\rangle$ in time (3.12). Therefore, the total cost of **Step 3** is (3.11). □

For convenience, we denote the complexity (3.11) to implement the oracle (3.4) as $O(T_{\mathcal{O}})$. Substituting the value of $O(T_K)$ into it, we conclude that the complexity to implement the oracle is $O(\epsilon^{-2}\sqrt{M}(K+\kappa(X))\log(MN))$ or $O(\epsilon^{-2}\sqrt{M}\kappa(X)\log(KMN))$ when the centers are relatively well separated.

With the above constructed oracle, we can simulate $\exp(-\mathbf{i}t\widetilde{A}/(M+K))$ in time $\widetilde{O}(t^2T_{\mathcal{O}}/\epsilon)$ since $\|A\|_{\max} = O(1)$. Similar to the complexity analysis in section 2, solving the linear system (1.2) costs $\widetilde{O}(\epsilon^{-3}\kappa^3T_{\mathcal{O}}\log(M+K))$. Concluding this, we have

Theorem 2 *Let $\mathcal{X} = \{\mathbf{x}^{(t)} \in \mathbf{R}^N : t = 1,\ldots,M\}$ be the training samples of the RBF network, and $\mathcal{M} = \{\mathbf{m}^{(t)} \in \mathbf{R}^N : t = 1,\ldots,K\}$ are the centers of the clusters of $\mathcal{X}$ determined by K-means. If the centers are $\mathcal{M}$, then the quantum state of the weight of the RBF network can be obtained in time*

$$O(\epsilon^{-5}\sqrt{M}\kappa^3(K+\kappa(X))\log(MN)) \tag{3.13}$$

or

$$O(\epsilon^{-5}\sqrt{M}\kappa^3\kappa(X)\log(KMN)) \tag{3.14}$$

when the centers are relatively well separated, where ϵ is the precision, κ is the condition number of $A = (\exp(-\|\mathbf{x}^{(i)} - \mathbf{m}^{(j)}\|^2/2\sigma^2))_{M\times K}$ and $\kappa(X)$ is the condition number of the sample matrix $X = (\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(M)})$.

This achieves an quadratic speedup at M and exponential speedup at N over classical training algorithms. When the centers are relatively well separated, then exponential speedup at K is also obtained.

Remark 3 Based on the proof of proposition 1, $\sqrt{M}$ in theorem 2 is caused by the state $|\chi\rangle$. If we use other quantum linear solvers, such as quantum linear solver based on block-encoding, we need to encode the matrix A into a unitary. A simple method is to use SVE, which requires the construction of the quantum states of the columns of A. The problem still exists if we start from $|\chi\rangle$, thus $\sqrt{M}$ cannot be improved in this way. The dependence on the condition number and the precision may be improved. However, the dependence on the precision cannot be improved into logarithm since amplitude estimation still should be used

to estimate $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|$ in type I and $\|\mathbf{x}^{(i)} - \mathbf{m}^{(j)}\|$ in type II. On the other hand, if we use the result [39], then we can achieve exponential speedup at M. However, the dependence on N, K becomes polynomial.

At the end of this section, we show how to apply the above method to achieve exponential speedup at N for the type I choices of centers studied in section 2. The centers in type I are the sample themselves that do not depend on K-means algorithm. Similar to above analysis, it suffices to build the oracle (3.4) for type I choices of the centers. The basic procedure is similar to but simple than **Step 1-5**. More precisely, for any i, j, denote $|\phi_{ij}\rangle$ as the quantum state proportional to $\|\mathbf{x}^{(i)}\||+\rangle|\mathbf{x}^{(i)}\rangle - \|\mathbf{x}^{(j)}\||-\rangle|\mathbf{x}^{(j)}\rangle$. Now prepare $|i, j\rangle|\phi_{ij}\rangle$, and apply amplitude estimation to estimate the amplitude of $|0\rangle$ of the first register of $|\phi_{ij}\rangle$. Then we get $|i, j\rangle|\phi_{ij}\rangle|\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|\rangle$. Finally, we can obtain $|i, j\rangle|\exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma^2)\rangle$ by applying an oracle to compute $\exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma^2)$, and undo amplitude estimation and the preparation of $|\phi_{ij}\rangle$. Since the preparation of $|\phi_{ij}\rangle$ is efficient in time $O(\log N)$ due to qRAM, the amplitude estimation costs $O(\epsilon^{-1}\log N)$. This is also the runtime to implement the required oracle. Therefore, theorem 1 can be improved into

Theorem 3 *Let $\{\mathbf{x}^{(t)} \in \mathbf{R}^N : t = 1, \ldots, M\}$ be the training samples of the RBF network. If the centers are the training samples, then the quantum state of the weight of the RBF network can be obtained in time*

$$O(\kappa^3(\log MN)/\epsilon^4), \tag{3.15}$$

where ϵ is the precision and κ is the condition number of $A = (\exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma^2))_{M\times M}$.

4 Conclusions

RBF network is an important approach in machine learning. To apply it to solve practical problems in big data era efficiently, faster training algorithms are required. In this paper, based on Hamiltonian simulation technique of Hermitian matrices and quantum K-means algorithm, we proposed two quantum algorithms to train RBF networks. The results show that quantum computer can achieve quadratic or even exponential speedup at the number of samples. The construction of the quantum algorithms depict the great power of quantum parallelism feature. The techniques used in this paper are also applicable to other weight training problems based on the idea of kernel method.

Since the problem relates to linear system solving, the limitations of quantum linear solvers also exist in our algorithms. The typical one is the preparation of quantum states, which is a bottleneck of many quantum machine learning algorithms. Another one is the readout problem from the quantum state of the weight to its classical information. However, for RBF network, the quantum state of the weight is already enough for us to do future predictions. This means we do not need to recover the classical information of the weight from its quantum state.

Acknowledgement

This research was supported partially by the National Natural Science Foundation of China under Grant No. 11671388 and CAS Project QYZDJ-SSW-SYS022.

References

1. T.M. Cover (1965), *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, IEEE Transactions on Electronic Computers, vol. EC-14, pp. 326-334.
2. E. Alpaydin (2015), *Introduction to Machine Learning*, 3rd edition, MIT press.
3. B. Schölkopf (1998), *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, Neural Computation, 10, pp. 1299-1319.
4. S. Haykin (2009), *Neural Networks and Learning Machines*, 3rd edition, Pearson.
5. W. Liu, J. C. Principe and S. Haykin (2010), *Kernel Adaptive Filtering: A Comprehensive Introduction*, John Wiley.
6. M.A. Aizerman, E.M. Braverman and L.I. Rozoner (1964), *Theoretical foundations of the potential function method in pattern recognition learning*, Automation and Remote Control, 25, pp. 821-837.
7. M. Schuld and N. Killoran (2019), *Quantum Machine Learning in Feature Hilbert Spaces*, Phys. Rev. Lett. 122, pp. 040504.
8. B. Schölkopf and A. J. Smola (2002), *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press.
9. A.W. Harrow, A. Hassidim and S. Lloyd (2009), *Quantum algorithm for linear systems of equations*, Phys. Rev. Lett. 103, pp. 150502.
10. A. Ambainis (2012), *Variable time amplitude amplification and quantum algorithms for linear algebra problems*, in: STACS'12 (29th Symposium on Theoretical Aspects of Computer Science), vol. 14. LIPIcs, 2012, pp. 636-647.
11. A.M. Childs, E. Kothari and R.D. Somma (2017), *Quantum algorithm for systems of linear equations with exponentially improved dependence on precision*, SIAM J. Comput. 46, pp. 1920-1950.
12. B.D. Clader, B.C. Jacobs and C.R. Sprouse (2013), *Preconditioned quantum linear system algorithm*, Phys. Rev. Lett. 110, pp. 250504.
13. L. Wossnig, Z.K. Zhao and A. Prakash (2018), *A quantum linear system algorithm for dense matrices*, Phys. Rev. Lett. 120, pp. 050502.
14. C. Shao and H. Xiang (2018), *Quantum Circulant Preconditioner for Linear System of Equations*, Phys. Rev. A, 98, pp. 062321.
15. S. Chakraborty, A. Gilyén and S. Jeffery (2018), *The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation*, arXiv:1804.01973.
16. A. Gilyén, Y. Su, G.H. Low and N. Wiebe (2018), *Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics*, arXiv:1806.01838.
17. M. Schuld, I. Sinayskiy and F. Petruccione (2016), *Prediction by linear regression on a quantum computer*, Phys. Rev. A, 94, pp. 022342.
18. G.M. Wang (2017), *Quantum algorithm for linear regression*, Phys. Rev. A, 96, pp. 012335.
19. N. Wiebe, D. Braun and S. Lloyd (2012), *Quantum Algorithm for Data Fitting*, Phys. Rev. Lett. 109, pp. 050505.
20. S. Lloyd, M. Mohseni and P. Rebentrost (2014), *Quantum principal component analysis*, Nature Physics, 10, pp. 631-633.
21. P. Rebentrost, M. Mohseni and S. Lloyd (2014), *Quantum support vector machine for big data classification*, Phys. Rev. Lett. 113, pp. 130503.
22. P. Wittek (2014), *Quantum Machine Learning: What Quantum Computing Means to Data Mining*, Academic Press.
23. D.S. Broomhead and D. Lowe (1988), *Multivariable functional interpolation and adaptive networks*, Complex Systems, 2, pp. 321-355.
24. R.M. Rifkin (2002), *Everything old is new again: A fresh look at historical approaches in machine learning*, Ph.D. thesis, MIT.
25. P.V. Yee and S. Haykin (2001), *Regularized Radial Basis Function Networks: Theory and Applications*, John Wiley.
26. J. Park and I.W. Sandberg (1991), *Universal Approximation Using Radial-Basis-Function Networks*, Neural Computation, 3, pp. 246-257.

27. M.E. Biancolini (2017), *Fast Radial Basis Functions for Engineering Applications*, Springer.
28. I.S. Chen, C.F.N. Cowan and P.M. Grant (1991), *Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks*, IEEE Transactions on Neural Networks, 2, pp. 302-309.
29. P. Rebentrost, A. Steffens, I. Marvian and S. Lloyd (2018), *Quantum singular-value decomposition of nonsparse low-rank matrices*, Phys. Rev. A, 97, pp. 012327.
30. S. Lloyd, M. Mohseni and P. Rebentrost (2013), *Quantum algorithms for supervised and unsupervised machine learning*, arXiv:1307.0411v2.
31. V. Giovannetti, S. Lloyd and L. Maccone (2008), *Quantum random access memory*, Phys. Rev. Lett. 100, pp. 160501.
32. I. Kerenidis and A Prakash (2017), *Quantum Recommendation System*, ITCS, 2017, pp. 49:1-49:21.
33. R. Chatterjee and Y. Ting (2017), *Generalized coherent states, reproducing kernels, and quantum support vector machines*, Quantum Inf. Commun. 17 (15 & 16), pp. 1292-1306.
34. M. Schuld and F. Petruccione (2018), *Supervised Learning with Quantum Computers*, Springer.
35. S. Kimmel, C.Y.Y. Lin, G.H. Low, M. Ozols and T.J. Yoder (2017), *Hamiltonian simulation with optimal sample complexity*, npj Quantum Informationvolume 3: 13.
36. H. Buhrman, R. Cleve, J. Watrous and R. de Wolf (2001), *Quantum Fingerprinting*, Phys. Rev. Lett. 87, pp. 167902.
37. G. Brassard, P. Høyer and M. Mosca (2002), *Quantum Amplitude Amplification and Estimation*, Quantum Computation and Quantum Information, Samuel J. Lomonaco, Jr. (editor), AMS Contemporary Mathematics, 305, pp. 53-74.
38. S. Lloyd (1982), *Least squares quantization in pcm*, IEEE transactions on information theory, 28(2), pp. 129-137.
39. I. Kerenidis, J. Landman, A. Luongo and A. Prakash (2018), *q-means: A quantum algorithm for unsupervised machine learning*, arXiv:1812.03584v2.
40. D.W. Berry, G. Ahokas, R. Cleve and B.C. Sanders (2007), *Efficient quantum algorithms for simulating sparse Hamiltonians*, Comm. Math. Phys. 270, pp. 359-371.

Appendix A: Brief overview of amplitude estimation

Let $|\phi\rangle = \cos\theta|0\rangle|u\rangle + \sin\theta|1\rangle|v\rangle$ be a quantum state that can be prepared in time $O(T)$. Using quantum phase estimation (QPE), one can obtain ϵ-approximations of $\pm\theta$ in a quantum computer with high success probability close to 1 in time $O(T/\epsilon)$. Thus, estimates of $|\sin\theta|^2$ and $|\cos\theta|^2$ can be obtained.

Assume that there is a unitary operator U with implementation time $O(T)$ such that $|\phi\rangle = U|0\rangle^{\otimes k}$. Let Z be the Pauli-Z matrix that maps $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $-|1\rangle$. Denote $G = (2|\phi\rangle\langle\phi| - I)(Z \otimes I)$. It is easy to check that

$$G = \begin{bmatrix} \cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{bmatrix}$$

in the space spanned by $\{|0\rangle|u\rangle, |1\rangle|v\rangle\}$. The eigenvalues of G are $e^{\mp \mathbf{i}2\theta}$ and the corresponding eigenvectors are $|w_\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle|u\rangle \pm \mathbf{i}|1\rangle|v\rangle)$.

To apply QPE, we choose the initial state as $|0\rangle^{\otimes n}|\phi\rangle$, where $n = O(\log 1/\delta\epsilon)$. It can be rewritten as

$$|0\rangle^{\otimes n}|\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle^{\otimes n}(e^{\mathbf{i}\theta}|w_-\rangle + e^{-\mathbf{i}\theta}|w_+\rangle).$$

By QPE, we obtain the following state

$$\frac{1}{\sqrt{2}}(e^{\mathbf{i}\theta}|y\rangle|w_-\rangle + e^{-\mathbf{i}\theta}|-y\rangle|w_+\rangle) \tag{A.1}$$

in time $O(T/\epsilon\delta)$, where $y \in \mathbf{Z}_{2^n}$ satisfies $|\theta - y\pi/2^n| \leq \epsilon$. Performing measurements on (A.1), we will get an ϵ-approximation $\tilde{\theta}$ of θ or $-\theta$ with probability at least $1-\delta$. From the approximation of $\pm\theta$, we can estimate the probabilities $|\sin\theta|^2$ and $|\cos\theta|^2$.

To further apply the information of θ to solve other problems, instead of performing measurements in (A.1) to get classical information of $\tilde{\theta}$, it is more useful to generate the following quantum state

$$|\phi\rangle|f(\tilde{\theta})\rangle, \tag{A.2}$$

where f is an even function. The quantum state (A.2) is obtained by adding a register to store $f(\tilde{\theta})$ and undoing the QPE. If we choose f as the cosine function, then we can store the amplitude of $|0\rangle$ of $|\phi\rangle$ in the ancilla register in (A.2).