A Survey on Quantum Machine Learning

Turan Kaan Elgin - Yiming Huang - Venkatraman Narayanan

December 2018

1 Introduction

Machine learning and data analysis are emerging disciplines in modern technologies. They are used in various fields as computer vision, computational biology, computer security and semiconductor industry. These techniques are based on analyzing data using statistical techniques and giving computers learning abilities based on the analysis of observed data.

Machine learning algorithms can be broadly classified, based on learning style, into supervised learning, unsupervised learning and semi-supervised learning. Supervised learning has access to data categories, while unsupervised learning does not know about those. Semi-supervised learning tries to have a balanced between the two.

The major drawbacks of using machine learning techniques are computation time and storage since in most cases, they require large amounts of data. One example for that is computer vision applications. For example, in a classical activity recognition task, there needs to be around 200 videos for training a classifier. Also by using modern deep learning algorithms, the training time can become even longer. Because of that, researchers usually use Graphics Processing Units (GPUs) in order to make computations feasible.

A next generation method for reducing the storage and computation time for those algorithms is utilizing quantum computers. The purpose of this project is to make a survey on quantizing classical machine learning algorithms and addressing some open questions.

Many machine learning problems utilize linear algebra as there are efficient ways to compute matrix operations by representing the data in matrices. Quantum computing makes some linear algebra computations faster, so implicitly improves classical machine learning tasks. An example for this is fast matrix inversion [10] which has been used in generating the hyperplane for QSVM [17].

Optimization also plays an important role in classical machine learning, since most of the time learning training rules is based on optimizing cost functions which arise from the interpretations that computers make. Numerical methods for optimization is a popular research area because of that reason; which aims to improve the computation of those optimization procedures. Quantum optimization is a branch of quantum computing which tries to improve these methods even further. Two popular examples for those methods are Quantum Gradient Descent [11] and Quantum Approximate Optimization Algorithm (QAOA) [7] which is used in quantum neural networks like Quantum Boltzman Machines [1].

In addition to those implicit methodologies used in Quantum Machine Learning, there are some certain quantum versions of classical machine learning algorithms. Some examples are Quantum Support Vector Machines (QSVM) [17] used for linear classification, Quantum Principle Component Analysis (QPCA) [14] used for dimensionality reduction, and Quantum Guassian Mixture Models [16] used for clustering and density estimation.

Another emerging subdiscipline of Machine Learning is Deep Learning and quantum computers are being utilized for deep learning applications as well which requires significant time and storage. Examples of these applications are Quantum Generative Adversarial Networks [15], Quantum Boltzmann Machines [1], Quantum Variational Autoencoders [12], and Quantum Convolutional Neural Networks [4].

In addition to those, there is a more sophisticated field inside Machine Learning which is known as Reinforcement Learning. It is based on learning as time goes on by exploring the environment. Quantum Reinforcement Learning [5] is also covered in this survey report.

2 Common Techniques and Concepts

2.1 Quantum Random Access Memory (QRAM)

Similar to classical random access memory (RAM), quantum access memory uses n qubits to address any quantum superposition of $N = 2^n$ memory cells. Suppose the data are $x_1, x_2, ..., x_m$. Then we can access the data using QRAM as the following: $|i\rangle|0\rangle|0\rangle \rightarrow |i\rangle|x_i\rangle||\vec{x_i}|\rangle$ where $|\vec{x_i}|$ is the magnitude and $|x_i\rangle$ is the state representing the data.

2.2 Hamiltonian Simulation

In quantum machine learning, there are some important techniques to implement unitary matrix U, which is used as a subroutine in phase estimation. Hamiltonian simulation is a technique, in which given an initial wave function $|\varphi(0)\rangle$ and Hamiltonian H; for any time $t > 0, \epsilon > 0$, there is a quantum circuit $U = e^{-iHt}$ acting on this initial quantum states such that $\left\|U - e^{-iHt}\right\| < \varepsilon$.
There are quite a lot of techniques under different settings for Hamiltonian simulation [2][3][13][14]. The technique proposed by Llyod et. al. gives us an efficient way to construct unitary U widely used in QML [14]. We can implement any unitary $e^{-i\rho t}$ with given n copies of density matrix ρ by

$$\begin{aligned} tr_P\left\{e^{-iS\Delta t}\left(\rho\otimes\sigma\right)e^{-iS\Delta t}\right\} &= \left(\cos^2\Delta t\right)\sigma + \left(\sin^2\Delta t\right)\sigma - i\sin^2\Delta t\cos^2\Delta t\left[\rho,\sigma\right] \\ &= \sigma - i\Delta t\left[\rho,\sigma\right] + O\left(\Delta t^2\right) \end{aligned}$$

where $\Delta t \rightarrow 0$ and S is a swap gate.

2.3 Amplitude Amplification

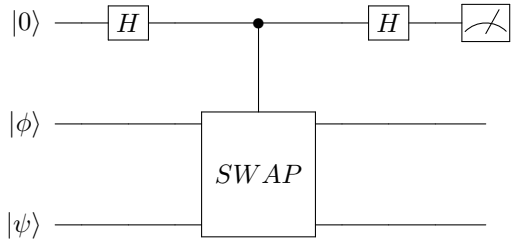
Amplitude amplification is the core part of Grover's algorithm, which is used to amplify the states which one cares about. There are basically two states as 'good' and 'bad' and by reflection techniques, the amplitude of the good state is enhanced.

2.4 Phase Estimation

Phase estimation is an algorithm for finding the eigenvalues of a unitary matrix. It is the core part of HHL algorithm [10] for solving linear systems which is the base of Quantum Machine Learning.

2.5 Swap Test

Swap test is a technique to measure the similarity between two states. The following is the representation of the framework used for swap test:



$$|0\rangle|\phi\rangle|\psi\rangle \rightarrow |0\rangle\otimes\frac{1}{2}(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) + |1\rangle\otimes\frac{1}{2}(|\phi\rangle|\psi\rangle - |\psi\rangle|\phi\rangle)$$

The measurement of the first qubit has the following probability: $\text{Prob}[|0\rangle] = \frac{1}{2} + \frac{1}{2}|\langle\psi|\theta\rangle|^2$. So if the qubit is 0, then it is likely for the two states to be equivalent.

2.6 Controlled Rotation

Controlled rotation gate is used for extracting eigenvalues from their state representations into amplitudes as the following when inverting a matrix:

$$\sum_{j=1}^{N} \beta_j |\lambda_j\rangle |u_j\rangle \rightarrow \sum_{j=1}^{N} \beta_j |u_j\rangle (\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle)$$

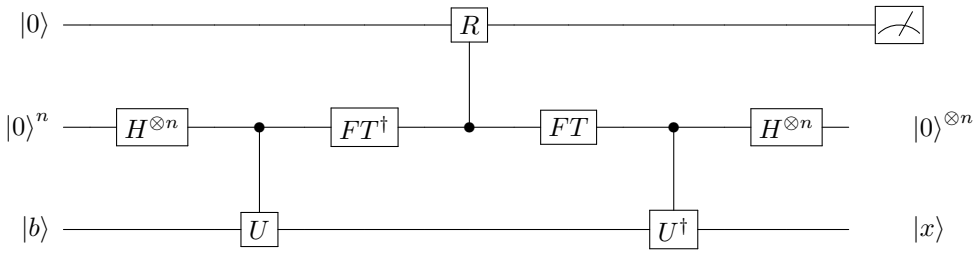
After that when the last qubit is measured the resulting state is the following:

$$\frac{1}{Z} \sum_{j=1}^{N} \beta_j \frac{C}{\lambda_j} |u_j\rangle \text{ where } Z \text{ is the normalization factor.}$$

3 Algorithms

3.1 Solving Linear Systems

The most basic algorithm which lead to many QML achievements is the quantum algorithm for solving linear systems which has been published in 2009 [10]. The algorithm tries to determine x from $Ax = b$. The following is the overall diagram of the algorithm.



At first, $|b\rangle$ is represented as $\sum_{i=1}^{N} b_i |i\rangle$, and e^{iAt} is generated using Hamiltonian simulation. Then in the first phase of the circuit, eigenvalues of A are computed using phase estimation as $e^{iAt}|b\rangle = e^{2\pi i\lambda}|b\rangle$ where $|\lambda\rangle$ is the output of the circuit. So the output of the first phase estimation is $\sum_{j=1}^{N} \beta_j |u_j\rangle |\lambda_j\rangle$ where $|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$ is the representation of $|b\rangle$ in the basis of eigenvectors of A.
After that a controlled rotation gate is applied to get the λ_j as the amplitude instead of state. After that reverse phase estimation uncomputes the state $|\lambda_j\rangle$ and measuring the last qubit gives the desired result as the following:

$$\sum_{j=1}^{N} \beta_j |u_j\rangle \Big(\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\Big) \rightarrow \sum_{j=1}^{N} \beta_j \lambda_j^{-1} |u_j\rangle$$

up to a normalization factor.
However in this algorithm, A should be Hermitian in order to use the Hamiltonian simulation. So the following matrix can be used instead: $\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$.

3.2 Support Vector Machines

3.2.1 Classical Support Vector Machines

Support Vector Machines is a famous machine learning algorithm which has a quantum version as well. In the classical algorithm, data is separated by a hyperplane and the margin between the two sides is maximized in order to provide a safety margin. The equation of the hyperplane is the following: $\vec{w}.\vec{x_j} - b$. The problem is solved by the following optimization framework: $\min_{\vec{w},b} \|\vec{w}\|$ s.t. $y_i(\vec{w}.\vec{x_i} - b) \geq 1$. The dual formulation is the following:

$$\max_{\vec{\alpha}} L(\vec{\alpha}) = \sum_{j=1}^{M} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j x_j . x_k \alpha_k \quad \text{s.t. } \sum_{j=1}^{M} \alpha_j = 0,\ y_j \alpha_j \geq 0$$

For nonlinear classification, kernel function $K_{jk} = k(x_j, x_k)$ is introduced and by Mercer's theorem, $\sum_{j,k=1}^{M} \alpha_j x_j . x_k \alpha_k$ can be transformed into the form $\sum_{j,k=1}^{M} \alpha_j K_{jk} \alpha_k$. At the end, the primal parameters are recovered as $\vec{w} = \sum_{j=1}^{M} \alpha_j \vec{x_j}$, and $b = y_j - \vec{w}.\vec{x_j}$. The decision for binary classification is the following: $y(\vec{x}) = \text{sign}\big(\sum_{j=1}^{M} \alpha_j k(\vec{x_j}, \vec{x}) + b\big)$.

3.2.2 Quantum Support Vector Machines

Quantum SVM has been published in 2014 [17] for solving least-squares SVM. In QSVM, at first the training data is represented as a state by utilizing QRAM. The state $|\vec{x_j}\rangle = \frac{1}{|\vec{x_j}|} \sum_{k=1}^{N} (\vec{x_j})_k |k\rangle$ represents the training data. Training data oracle gives the following state as it maps indices to data:

$$\frac{1}{\sqrt{M}} \sum_{i=1}^{M} |i\rangle \rightarrow |\chi\rangle = \frac{1}{\sqrt{N_\chi}} \sum_{i=1}^{M} |\vec{x_i}||i\rangle|\vec{x_i}\rangle \text{ where } N_\chi \text{ is the normalization factor.}$$

By using Hamiltonian simulation, $e^{-i\hat{K}\Delta t}$ is obtained where $\hat{K} = \frac{K}{\text{tr}K}$.
The algorithm solves the least-squares SVM problem defined as the following:

$$\min_{w,b,e} J(w, b, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{N} e_k^2 \quad \text{s.t.} \quad y_k(w^T \phi(x_k) + b) = 1 - e_k$$

When the dual problem is solved, one gets the following result:

$$F \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \gamma^{-1} I \end{pmatrix} \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix}$$

Then by using Hamiltonian technique, $e^{-i\hat{F}\delta t}$ can be constructed and the system is solved by HHL algorithm explained above [10]. Then the state for the variables is defined in the computational basis as the following:

$$|b, \vec{\alpha}\rangle = \frac{1}{C}(b|0\rangle + \sum_{k=1}^{M} \alpha_k |k\rangle)$$

where C is the normalizing state.
In the classification part, the aim is to classify the query state defined as the following:

$$|\tilde{x}\rangle = \frac{1}{N_{\tilde{x}}}(|0\rangle|0\rangle + \sum_{i=1}^{M} |\vec{x}||k\rangle|\vec{x_k}\rangle)$$

Training data is constructed by using oracle in the following way:

$$|b, \vec{\alpha}\rangle \rightarrow \frac{1}{\sqrt{N_{\tilde{u}}}}(b|0\rangle|0\rangle + \sum_{i=1}^{M} \alpha_k |\vec{x_k}||k\rangle|\vec{x_k}\rangle)$$

At the end the likelihood of the two states is measured using swap test.
By this algorithm a very famous machine learning algorithm called SVM is represented on a quantum computer with a logarithmic complexity.

3.3 Principle Component Analysis

PCA is a commonly used machine learning algorithm for reducing the dimensionality of data. Quantum version of PCA, which has been published in 2014 [14], utilizes common techniques such as Hamiltonian simulation and phase estimation.
By using phase estimation algorithm, one can decompose a density matrix into its eigenvalues and eigenvectors where the density matrix represents the data. Suppose the following is the representation of an arbitrary vector $|\psi\rangle$ in the eigenbasis of ρ.

$$|\psi\rangle|0\rangle \rightarrow \sum_i \psi_i |\chi_i\rangle|\tilde{r_i}\rangle$$

Then one can decompose ρ in its own eigenbasis as the following:

$$\rho|0\rangle \rightarrow \sum_i r_i|\chi_i\rangle\langle\chi_i| \otimes |\tilde{r_i}\rangle\langle\tilde{r_i}|$$

Then using some sampling techniques, the most relevant eigenvectors can be taken from this state in logarithmic time which provides an exponential speedup.
QPCA is used for state discrimination and assignment which are the quantum versions of supervised learning and clustering. Suppose there are two sets as $\{|\phi_i\rangle\}$ and $\{|\psi_i\rangle\}$ characterized by density matrices $\rho = \frac{1}{m}\sum_i |\phi_i\rangle\langle\phi_i|$ and $\sigma = \frac{1}{m}\sum_i |\psi_i\rangle\langle\psi_i|$. If we want to classify a state $|\chi\rangle$, we can represent it in the eigenbasis of $\rho - \sigma$. If the eigenvalues are positive, it should be assigned to the first set; otherwise, to the second set.

3.4 Singular Value Decomposition

3.4.1 Classical Singular Value Decomposition

SVD algorithm is used for computing eigenvalues and eigenvectors of matrices in an efficient way. Singular Value Decomposition, known as SVD, is a widely used technique in linear algebra used for factorizing and revealing the eigenvalues and eigenvectors of given matrix M [9]. Given a matrix $M \in \mathbb{C}^{m\times n}$, the goal is to factorize M into a form $U\Sigma V^\dagger$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are matrices with columns which are orthonormal eigenvectors of MM^* and M^*M, respectively. $\Sigma \in \mathbb{C}^{m\times n}$ is a rectangular diagonal matrix with non-negative singular values on the diagonal. As the Σ stores singular values in descending order, the larger singular value corresponding to more important components, which are eigenvectors of matrix M which represent the data more effectively. So it provides a way to extract most important components of given data. Recently, Rebentrost et.al. proposed a quantum version of singular decomposition method for non-sparse low-rank matrices [19], which is based on the techniques of Quantum Principle Component Analysis [14] and not restricted to deal with positive semi-definite matrices.

3.4.2 Quantum Singular Value Decomposition

Since it is used in some machine learning algorithms, QSVD algorithm, which has been published in 2016 [19], has a significant place in the development of QML field. In order to reveal eigenvalues and eigenvectors of a given matrix, the most intuitive way is to apply framework of phase estimation. So the question can be converted into two problems:

- How can matrix A be formed?
- How can control unitary $e^{-i\frac{A}{N}t}$ be constructed?

After solving these problems, one is able to use the general phase estimation framework to find eigenvalues and eigenvectors.
QSVD uses modified swap gate instead of normal one. The modified swap matrix $S \in \mathbb{C}^{N^2\times N^2}$ between single copy ρ and σ is

$$S = \sum_{j,k=1}^{N} A_{jk} |j\rangle \langle k| \otimes |k\rangle \langle j|.$$

The authors provided a swap gate on bigger space with reasonable resource to implement. Firstly, they assume one has an oracle to access the element of matrix A by QRAM [8], as

$$|j\rangle |k\rangle |0, ..., 0\rangle \rightarrow |j\rangle |k\rangle |A_{jk}\rangle$$

using $O(N^2)$ storage space and $O(log^2 N)$ operations. Similarly, S can be constructed by the same oracle,

$$|(j,k)\rangle |0, ..., 0\rangle \rightarrow |(j,k)\rangle \left|(k,j), S_{(k,j)(j,k)}\right\rangle$$

The unitary $e^{-i\frac{A}{N}t}$ can be used to reveal the eigenvalues and eigenvectors of matrix A based on the Hamiltonian simulation techniques proposed in QPCA [14]. As this modified swap matrix extends usual swap matrix to higher dimension, it makes this matrix one-sparse. By using a similar Hamiltonian simulation method, one has

$$\mathrm{tr}_1\left\{e^{-iS\Delta t}\rho\otimes\sigma e^{iS\Delta t}\right\} = \sigma - i\mathrm{tr}_1\left\{S\rho\otimes\sigma\right\}\Delta t + i\mathrm{tr}_1\left\{\rho\otimes\sigma S^\dagger\right\}\Delta t + O\left(\Delta t^2\right)$$

where

$$\mathrm{tr}_1\left\{S\rho\otimes\sigma\right\}=\mathrm{tr}_1\left\{\sum_{j,k=1}^{N}S_{jk}\left|j\right\rangle\left\langle k\right|\otimes\left|k\right\rangle\left\langle j\right|\left(\rho\otimes\sigma\right)\right\}.$$

Assume $\rho=\frac{1}{N}\sum\limits_{l=1}^{N}\left|l\right\rangle\left\langle l\right|$ is composed of pure states, one has $\mathrm{tr}_1\left\{S\rho\otimes\sigma\right\}=\frac{A}{N}\sigma$.
Similarly, one has $\mathrm{tr}_1\left\{\rho\otimes\sigma S^{\dagger}\right\}=\sigma\frac{A}{N}$. For evolving with this modified swap gates in each small timestep on system ρ, one has

$$\lim_{\Delta t\to 0}\mathrm{tr}_1\left\{e^{-iS\Delta t}\rho\otimes\sigma e^{iS\Delta t}\right\}=\lim_{\Delta t\to 0}\left(\sigma-i\frac{\Delta t}{N}\left[S,\sigma\right]+O\left(\Delta t^2\right)\right)=e^{-i\frac{S}{N}\Delta t}\sigma e^{i\frac{S}{N}\Delta t}$$

After constructing $e^{-i\frac{S}{N}t}$, controlled-$e^{-i\frac{S}{N}t}$ can easily be constructed using controlled-modified swap gate, like $e^{-i(\left|1\right\rangle\left\langle 1\right|\otimes S\Delta t)}$.

3.5 Gradient Descent

Gradient descent [11] is one of the most basic iterative optimization algorithms which optimizes a function by updating parameters in multiple iterations. It has a wide usage in machine learning algorithms, so quantizing it lead to many developments in QML. The original paper has been published in 2016 [18] for polynomial optimization. Later on an updated version for linear systems and least squares has been published in 2017 [11].
In the classical case, suppose the update rule is the following: $\theta_{t+1}=\theta_t+\alpha r_t$ where θ is the parameter to be optimized. Also assume $r_{t+1}=S(r_t)$. Then one has $\theta_\tau=r_0+\alpha\sum_{t=1}^{T}S^{t-1}(L(r_0))$. Suppose one has the following unitaries:

$$\begin{aligned}V:&\left|0\right\rangle\left|r_0\right\rangle\left|0\right\rangle\to\left|1\right\rangle(\alpha\|\tilde{L}(r_0)\|\left|\tilde{L}(r_0)\right\rangle\left|0\right\rangle+\left|G_1\right\rangle\left|1\right\rangle)\\&:\left|t\right\rangle\left\|r_t\right\|\left|r_t\right\rangle\left|0\right\rangle\to\left|t+1\right\rangle(\|\tilde{S}(r_t)\|\left|\tilde{S}(r_t)\right\rangle\left|0\right\rangle+\left|G_{t+1}\right\rangle\left|1\right\rangle)\end{aligned}\tag{1}$$

$$\begin{aligned}U:&\left|0\right\rangle\left|0\right\rangle\left|0\right\rangle\left|r_0\right\rangle\left|0\right\rangle\to\left|0\right\rangle\left|0\right\rangle\left|0\right\rangle\left|r_0\right\rangle\left|0\right\rangle\\&:\left|t\right\rangle\left|0\right\rangle\left|0\right\rangle\left|r_0\right\rangle\left|0\right\rangle\to\left|t\right\rangle\left|0\right\rangle(\alpha\|\tilde{S^{t-1}}(\tilde{L}(r_0))\|\left|t\right\rangle\left|\tilde{S}^{t-1}(\tilde{L}(r_0))\right\rangle\left|0\right\rangle+\left|G'_t\right\rangle\left|1\right\rangle)\end{aligned}\tag{2}$$

where $\|S^{t-1}(L(r_0))-\tilde{S}^{t-1}(\tilde{L}(r_0))\|\leq t\epsilon$ and $\left|G\right\rangle$ is a garbage state. So $\tilde{S}^{t-1}(\tilde{L}(r_0))$ provides a reasonable approximation for $S^{t-1}(L(r_0))$ and having a garbage state means the mapping is done up to some probability which should be enhanced further.
Then the algorithm does the following: It starts from the state $\frac{1}{\sqrt{\tau+1}}\sum_{t=0}^{\tau}\left|t\right\rangle\left|0\right\rangle\left|0\right\rangle\left|r_0\right\rangle\left|0\right\rangle$ and applies the unitaries to get the final state: $\frac{1}{T}\left|\tilde{\theta_\tau}\right\rangle\left|0\right\rangle+\left|G\right\rangle\left|0^{\perp}\right\rangle$. After that by using amplitude amplification technique, one discards the garbage state and reaches to the update rule.
Development of the gradient descent algorithm lead to the development of more sophisticated algorithms like Newton's method.

3.6 Generative Adversarial Networks

Neural networks are one of the most widely used techniques in today's machine learning technologies. They usually require significant amount of time to train and used with GPUs. Quantum computing can make them more efficient and lots of research has been done in the recent past.
One example of quantum neural networks is generative adversarial networks (GANs) published in 2018 [15]. It has the following structure: There are two components called discriminator and generator. Generator's duty is to generate fake data to fool the discriminator and discriminator tries to discriminate among the real and fake data.
In the quantum case, the discriminator makes a positive operator valued measurement. Suppose the associated measurement operators are T for true outcome and F for false outcome where $T+F=I$. Further suppose, the real and fake data are represented by density matrices called σ and ρ. Then

$$P(T|\text{real data})=\mathrm{tr}(T\sigma),\ P(T|\text{fake data})=\mathrm{tr}(T\rho)$$

Discriminator tries to maximize the first probability, while the generator tries to maximize the second one. Since $\|T\|,\|F\|\leq 1$, the optimization problem is convex. Some optimization techniques as quantum gradient descent

can be used in that problem. The discriminator's objective is to find an optimal measurement operator T, while generator's objective is to find a good ensemble ρ representing the fake data. The equilibrium is where $P(T|\text{real data}) = P(T|\text{fake data}) = \frac{1}{2}$. Since quantum systems are intrinsically probabilistic, the model is more intuitive than the classical case.

3.7 Boltzmann Machines

The Boltzmann machine (BM) is a classical machine learning technique, and serves as the basis of powerful deep learning models such as deep belief networks and deep Boltzmann machines. The Quantum Boltzmann Machine (QBM) is a probabilistic model based on Boltzmann distribution of a quantum Hamiltonian. The model has been published in 2018 [1].

3.7.1 Classical Boltzmann Machines

Classical Boltzmann machines comprises a probabilistic network of binary units with a quadratic energy function. A BM consists of hidden and visible units. The training setup is as follows:
The energy function is defined as,

$$E_z = -\sum_a b_a z_a - \sum_{a,b} w_{ab} z_a z_b$$

where z_a is the binary value associated to unit a. The parameters b_a and w_{ab} are optimized in training phase which are bias parameter of unit a and weight parameter between different units. The objective is to determine those parameters, $\theta \in \{b_a, w_{ab}\}$, such that the Boltzmann marginal probability P_v is close to the probability of data, ${P_v}^{data}$. The Boltzmann marginal probability is the probability of visible units after marginalization over hidden units. The formulation is the following:

$$P_v = Z^{-1} \sum_h e^{-E_z}, Z = \sum_z e^{-E_z}$$

The optimization is carried out with minimization of negative average log likelihood function defined by,

$$\mathcal{L} = -\sum_v {P_v}^{data} \log P_v$$

which measures the similarity between two probability distributions. A gradient descent approach is used to achieve minimization of the loss function.

$$\delta\theta = -\eta \partial_\theta \mathcal{L}$$

where η is the learning rate parameter.

3.7.2 Quantum Boltzmann Machines

The mentioned classical Boltzmann machine is converted to a quantum variant by replacing classical bits z_a by qubits, thereby making the energy function to a Hamiltonian,

$$H = -\sum_a b_a {\sigma_a}^z - \sum_{a,b} w_{ab} {\sigma_a}^z {\sigma_b}^z$$

where ${\sigma_a}^z = I^{\otimes(a-1)} \otimes \sigma^Z \otimes I^{\otimes(N-a)}$ and N is the total number of units in the network. The diagonal elements of the density matrix ρ gives the Boltzmann probabilities of all 2^N states where

$$\rho = Z^{-1} e^{-H}, Z = Tr\left[e^{-H}\right]$$

In the classical setting, marginalization is accomplished by summing over hidden variables; while in the quantum setting, it is accomplished by projection on visible variables as

$$P_v = Tr\left[\Lambda_v \rho\right]$$

where Λ_v provides projection over visible variables. Hence,

$$\Lambda_v = |v\rangle\langle v| \otimes I_h$$

and I_h is a identity matrix acting on hidden variables.
Finally, to train QBM, one optimises the parameters θ such that the probability distribution P_v is close to ${P_v}^{data}$ by minimising the negative log-likelihood function,

$$\mathcal{L} = -\sum_v {P_v}^{data} \log \frac{Tr\left[\Lambda_v e^{-H}\right]}{Tr\left[e^{-H}\right]}$$

3.8 Variational Autoencoders

Variational autoencoders are used for generating data which look like the input data but not exactly the same. At first, a network called encoder encodes the data into a hidden space, then decoder network reconstructs the data.

3.8.1 Classical Variational Autoencoders [12]

Let $X = \{x_i\}_{i=1}^n$ denote the training data and θ be the model parameters. Furthermore, let p_{data} represent the data distribution and $p_\theta(x)$ represent the model distribution which are supposed to be close enough for a good representation. According to maximum likelihood estimation, one needs to maximize $\mathbb{E}_{x\sim p_{data}}[\log p_\theta(x)]$. Let z denote the latent variables. In that case, $p_\theta(x) = \sum_z p_\theta(x,z)$. By Bayes rule, $p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)}$. Therefore the following derivation holds:

$$\begin{aligned} \mathop{\mathbb{E}}_{x\sim p_{data}}[\log p_\theta(x)] &= \mathop{\mathbb{E}}_{x\sim p_{data}}\left[\mathop{\mathbb{E}}_{z\sim p_\theta(z|x)}\left[\log \frac{p_\theta(x,z)}{p_\theta(z|x)}\right]\right] \\ &= \mathop{\mathbb{E}}_{x\sim p_{data}}\left[\mathop{\mathbb{E}}_{z\sim p_\theta(z|x)}[\log p_\theta(x|z)] - \mathop{\mathbb{E}}_{z\sim p_\theta(z|x)}\left[\log \frac{p_\theta(z|x)}{p_\theta(z)}\right]\right] \end{aligned} \tag{3}$$

The second term inside the outer expectation is known as KL divergence which measures the distance between two distributions. Variational autoencoder approximates the posterior distribution since it is intractable to compute: $p_\theta(z|x) \approx q_\phi(z|x)$. In this case $q_\phi(z|x)$ is known as variational distribution and ϕ as variational parameters. Since KL divergence is always greater than or equal to zero, equation (3) provides a tractable lower bound. First the following objective function is defined:

$$L(\theta,\phi,x) = \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p_\theta(z)}\right]$$

Then one has the following:

$$\mathop{\mathbb{E}}_{x\sim p_{data}}[L(\theta,\phi,x)] \leq \mathop{\mathbb{E}}_{x\sim p_{data}}[\log p_\theta(x)]$$

Next thing is how to represent the latent space. One way to do that is using Boltzmann Machines. The following is the formulation:

$$p_\theta(z) \equiv e^{-\mathbb{E}_\theta(z)}/Z_\theta,\ Z_\theta = \sum_z e^{-\mathbb{E}_\theta(z)}$$

where $\mathbb{E}_\theta$ is the energy function computed by the network. The cross entropy between the data distribution and the variational distribution is defined as

$$H(q_\phi, p_\theta) = -\mathop{\mathbb{E}}_{z\sim q_\phi}[\log p_\theta]$$

This terms gives the log-likelihood of latent variable z under the model p_θ. However computing this term is not tractable, so one needs an approximation called reparametrization trick: Define ρ as a random variable sampled from uniform distribution. Then

$$H(q_\phi, p_\theta) = -\mathop{\mathbb{E}}_{\rho\sim U}[\log p_\theta(z(\rho,\phi))] = \mathop{\mathbb{E}}_{\rho\sim U}[\mathop{\mathbb{E}}_{\theta}(z)] + \log Z_\theta$$

3.8.2 Quantum Variational Autoencoders

Quantum Variational Autoencoders has been published in 2018 [12]. In the quantum case, the formulation of Boltzmann machines needs to be utilized. Define H_θ as the energy function of a quantum Boltzmann machine defined as in section 3.7. Then after reparametrization, one has the following:

$$H(q_\phi, p_\theta) = -\mathop{\mathbb{E}}_{\rho\sim U}[\log(\mathrm{Tr}[\Lambda_z e^{-H_\theta}])] + \log Z_\theta$$

where Λ_z is the projection on the latent variables. The problem with this formulation is the gradient of the first term which is intractable. One needs to use Golden-Thompson inequality defined as below to overcome this issue:

$$\mathrm{Tr}[e^A e^B] \geq \mathrm{Tr}[e^{A+B}]$$

So the above expression becomes the following which gives a tractable lower bound as in the classical case:

$$H(q_\phi, p_\theta) \geq - \mathop{\mathbb{E}}_{z\sim q_\phi}[\log(\mathrm{Tr}[e^{-H_\theta + \ln \Lambda_z}])] + \log Z_\theta = \mathop{\mathbb{E}}_{\rho\sim U}[H_\theta(z(\rho,\phi))] + \log Z_\theta.$$

3.9 Gaussian Mixture Models

3.9.1 Classical Gaussian Mixture Models [16]

Gaussian Mixture Models is an algorithm used for clustering and density estimation. This technique is known as soft clustering because the cluster assignments are probabilistic. Each cluster is associated with a Gaussian distribution and the probability of a point p_i assigned to cluster k is the following:

$$P(p_i|k, \mu_k, \Sigma_k) = \frac{1}{Z_k} e^{-\frac{1}{2}(p_i-\mu_k)\Sigma_k^{-1}(p_i-\mu_k)}$$

where μ_k and Σ_k are the mean and covariance matrix for the particular cluster and Z_k is the normalization factor. In that case, one can estimate the probability density of each point as

$$\begin{aligned} P(p_i|\{\theta_k\}) &= \sum_{k=1}^{K} P(p_i, k|\{\theta_k\}) P(p_i|k, \{\theta_k\}) \\ &= \sum_{k=1}^{K} \Big[P(k|\{\theta_k\}) \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} e^{-(p_i-\mu_k)\Sigma_k^{-1}(p_i-\mu_k)} \Big] \end{aligned} \tag{4}$$

So, a multimodal distribution is estimated by using mixture of Gaussian probabilities.

3.9.2 Quantum Gaussian Mixture Models

Quantum Gaussian Mixture Models, published in 2016 [16], uses quantum techniques to represent mixture of probability distributions. Since quantum systems are intrinsically probabilistic, one can generate a wave function and use it as a probability distribution. The following wave function helps in this particular case:

$$\psi_k(p_i|\theta_k) = \frac{1}{\sqrt{Z_k}} e^{-\frac{1}{4}(p_i-\mu_k)\Sigma_k^{-1}(p_i-\mu_k)} e^{-i\phi_k}$$

The square of this wave function is associated with the Gaussian probability:

$$P(p_i|k, \theta_k) = |\psi_k(p_i|\theta_k)|^2 = \frac{1}{Z_k} e^{-\frac{1}{2}(p_i-\mu_k)\Sigma_k(p_i-\mu_k)}$$

The phase factor $e^{-i\phi_k}$ does not have an affect on the probability itself, but when it comes to mixture of probabilities, the interference of the wave function becomes important. In the overall formulation, there is a mixture coefficient for each wave function which are called α_k. Then the mixture formulation becomes the following:

$$\psi(p_i|\{\alpha_k, \theta_k\}) = \sum_{k=1}^{K} \alpha_k \psi_k(p_i|\theta_k) = \sum_{k=1}^{K} \Big[\frac{\alpha_k}{\sqrt{Z_k}} e^{-\frac{1}{4}(p_i-\mu_k)\Sigma^{-1}(p_i-\mu_k)} e^{-i\phi_k} \Big]$$

$$\implies P(p_i) = |\psi(p_i|\{\alpha_k, \theta_k\})|^2 = \sum_{k=1}^{K} \Big[\frac{\alpha_k}{\sqrt{Z_k}} e^{-\frac{1}{4}(p_i-\mu_k)\Sigma_k(p_i-\mu_k)} \sum_{k=1}^{K} \frac{\alpha_l *}{\sqrt{Z_k}} \cos(\phi_{l,k}(p_i)) e^{-\frac{1}{4}(p_i-\mu_l)\Sigma_l(p_i-\mu_l)} \Big]$$

where $\phi_{l,k} = \phi_k - \phi_l$. One can define a Gaussian wave as

$$G_{i,k} = \frac{1}{\sqrt{Z_k}} e^{-\frac{1}{4}(p_i-\mu_k)\Sigma_k^{-1}(p_i-\mu_k)}$$

Then the final model can be defined as the following:

$$P(p_i, k|\{\alpha_k, \theta_k\}) = \alpha_k G_{i,k}(\sum_{l=1}^{K} \alpha_l * \cos\phi_{l,k} G_{i,l}) \text{ subject to } \sum_i \sum_k P(p_i, k|\{\alpha_k, \theta_k\}) = 1.$$

This algorithm also has an improvement on the performance as well as time and storage as seen in the experiments performed [16].

3.10 Reinforcement Learning

Another field of machine learning is reinforcement learning where learning process happen as time progresses instead of training the model in advance. Quantum computing can be utilized for this field as well.

3.10.1 Classical Reinforcement Learning [5]

Reinforcement learning setting contains an agent and an environment. The agent tries to learn a policy by getting some information from the environment. A typical reinforcement learning setting is based on Markov decision processes. A Markov decision process is defined with the following set: $\{S, A_{(i)}, p_{ij}(a), r_{(i,a)}, V, i, j \in S, a \in A_{(i)}\}$. S denotes the set of the states, $A_{(i)}$ denotes the set of actions corresponding to state i, $p_{ij}(a)$ denotes the probability of transitioning from state i to j when action a is executed, $r_{(i,a)}$ denotes the reward of executing action a in state i, V is the value function that the agent tries to maximize. Reward function is defined from Γ to $(-\infty, +\infty)$ where $\Gamma = \{(i, a) : i \in S, a \in A_{(i)}\}$. π denotes the policy that the agent tries to learn and it is defined from $S \times \bigcup_{i \in S} A_{(i)}$ to $[0, 1]$. The value function is defined as the following:

$$V_s^\pi = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + ...|s_t = s, \pi] = \mathbb{E}[r_{t+1} + \gamma V_{s_{t+1}}^\pi | s_t = s, \pi] = \sum_{a \in A_s} \pi(s, a)[r_s^a + \gamma \sum_{s'} p_{ss'}^a V_{s'}^\pi]$$

where t denotes a timestep and γ is the discount factor in the range $[0, 1]$. It decreases the significance of the past actions as time progresses. The notations are defined as the following: $p_{ss'}^a = P[s_{t+1} = s'|s_t = s, a_t = a]$, $r_s^a = \mathbb{E}[r_{t+1}|s_t = s, a_t = a]$. The update rule for the value function is based on gradient descent technique and defined as

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$$

Then the optimal value is selected as $V_s^* = \max_{a \in A_s}[r_s^a + \gamma \sum_{s'} p_{ss'}^a V_{s'}^*]$ and the optimal policy is selected as $\pi^* = \text{argmax}_\pi V_s^\pi$.

3.10.2 Quantum Reinforcement Learning

In the quantum setting, published in 2008 [5], there are two Hilbert spaces H_S and H_A containing states and actions. There is a set of states $\{|s_1\rangle, |s_2\rangle, ..., |s_n\rangle\}$ which is an orthonormal basis for H_S and they are called eigenstates. Similarly the set of actions $\{|a_1\rangle, |a_2\rangle, ..., |a_n\rangle\}$ is an orthonormal basis for H_A and known as eigenactions. The number of states and actions are denoted as N_s and N_a respectively. The states are defined as superpositions of eigenstates as $|s^{N_s}\rangle = \sum_{i=1}^{N_s} C_i|s_i\rangle$ where C_i are the probability amplitudes. Similarly actions are defined as $|a^{N_a}\rangle = \sum_{j=1}^{N_a} C_j|a_j\rangle$. This setting also utilizes the fact that quantum systems are probabilistic. An important concept of reinforcement learning is the trade-off between exploration and exploitation which means exploring new states vs. exploiting the optimal action up to that point. That concept can be intrinsically provided by the probabilistic setting in quantum systems.

In reinforcement learning algorithms, there is a policy for selecting actions by considering exploration-exploitation trade-off. In the classical setting, exploration and exploitation are selected with some probabilities which can change over time. In the quantum setting, when an action, defined as superpositions of eigenactions, is measured, a particular eigenaction a_j is observed with the probability $|C_j|^2$. In that way, it provides a natural probabilistic selection of actions. This strategy is called action collapse where the superposition state collapses into a single eigenaction.

The next step is to amplify the probability of the selected action by using amplitude amplification technique. However there is key difference of this step with the Grover's algorithm: The probability of the particular action should be proportional to the value of that action instead of 1. In that way, actions having high values can be more likely selected compared to others. So $L = \text{int}(k(r + V(s')))$ where k is the proportion constant, r is the reward received from the action, s' is the next state and L is the number of iterations. According to this setting, the environment is represented by the reflections in the amplitude amplification since agent gets the information about those reflections when it measures the action state.

4 Open Problems

As further open problems in the area of QML, in PCA algorithm; Gaussian kernels can be used for nonlinear representations since existing approaches are based on simpler kernels as polynomials. Also sparse approximation problem based on $l0$ norm is NP-hard and quantum computing can be used in order to make it more efficient. Similar to this problem, there is a concept called Lasso regression based on $l1$ norm which is not currently represented in quantum setting and algorithms like quantum gradient descent and qauntum SVM are based on least-squares regularization. Also there is still ongoing research about noisy methods in quantum machine learning since in some cases, noise might make the optimization procedure get away from local minima.
In classical machine learning there are some methods which require less amount of data. One example for these is one-shot learning which tries to learn from one or few training samples. One of the techniques used in this context is data augmentation which is producing new data by modifying existing ones such cropping images from some position. However data augmentation would be more challenging in the case of quantum representation of the data. Simple operations like rotations can be simulated by unitaries, however more complex operations as cropping or pooling might require more sophisticated techniques. In the recent published paper on Quantum Convolutional Networks in 2018 [4], there is such a technique used for pooling the data: Some fraction of qubits are measured, and based on the measurement outcomes, nearby data are altered by relevant unitary operations. This also provides nonlinearity transformation to the network by reducing the degrees of freedom; however it might not be as strong as max pooling in the classical case or rectified linear unit used for introducing nonlinearity.
A stronger concept in classical machine learning is zero-shot learning in which the algorithm is able to detect a category which has no training data at all. The general way to do this is defining semantic relationships between seen and unseen labels and using those semantics to train the algorithm. However in quantum setting, it would be very challenging to define semantics which relate multiple data. The data might be related to each other by probabilities of measurement outcomes or even using entanglements as relationships between the data. Feasibility of these approaches might be considered for further improvements.
Another open problem is related to the environment setting in quantum reinforcement learning. As described in Section 3.10, the environment is defined as the reflections in the amplitude amplification algorithm since the agent amplifies the probabilities of the optimal actions by that algorithm. Another way to define the concept of environment might be oraculization which means using the environment as an oracle which gives the optimal actions [6]. There is still ongoing research about the feasibility of this approach.

References

[1] Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B., and Melko, R. Quantum boltzmann machine. *Physical review letters*, 8 (2018), 021050.

[2] Childs, A. M., Maslov, D., Nam, Y., Ross, N. J., and Su, Y. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences 115*, 38 (2018), 9456–9461.

[3] Childs, A. M., Ostrander, A., and Su, Y. Faster quantum simulation by randomization. *arXiv preprint arXiv:1805.08385* (2018).

[4] Cong, I., Choi, S., and Lukin, M. D. Quantum convolutional neural networks. *arXiv:1810.03787v1 [quant-ph]* (Oct. 2018).

[5] Dong, D., Chen, C., Li, H., and Tarn, T.-J. Quantum reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics 38*, 5 (2008), 1207–1220.

[6] Dunjko, V. A route towards quantum-enhanced artificial intelligence. 2018.

[7] Farhi, E., Goldstone, J., and Gutmann, S. A quantum approximate optimization algorithm.

[8] Giovannetti, V., Lloyd, S., and Maccone, L. Quantum random access memory. *Physical review letters 100*, 16 (2008), 160501.

[9] Golub, G. H., and Reinsch, C. Singular value decomposition and least squares solutions. *Numerische mathematik 14*, 5 (1970), 403–420.

[10] Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum algorithm for solving linear systems of equations. *Physical review letters 15*, 103 (2009), 150502.

[11] Kerenidis, I., and Prakash, A. Quantum gradient descent for linear systems and least squares.

[12] Khoshaman, A., Vinci, W., Denis, B., Andriyash, E., and Amin, M. H. Quantum variational autoencoder. *arXiv:1802.05779v1 [quant-ph]* (2018).

[13] Lloyd, S. Universal quantum simulators. *Science* (1996), 1073–1078.

[14] Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum principal component analysis. *Nature Physics 10*, 9 (2014), 631.

[15] Lloyd, S., and Weedbrook, C. Quantum generative adversarial learning. *Physical review letters* (2018), 040502.

[16] Rahman, M., and Geiger, D. Quantum clustering and gaussian mixtures. *arXiv:1612.09199v1 [stat.ML]* (2016).

[17] Rebentrost, P., Mohseni, M., and Lloyd, S. Quantum support vector machine for big data classification. *Physical review letters 113*, 13 (2014), 130503.

[18] Rebentrost, P., Schuld, M., Wossnig, L., Petruccione, F., and Lloyd, S. Quantum gradient descent and newton's method for constrained polynomial optimization. *arXiv:1612.01789 [quant-ph]* (Dec. 2016).

[19] Rebentrost, P., Steffens, A., Marvian, I., and Lloyd, S. Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical review A 97*, 1 (2018), 012327.