Quantum Machine Learning : An Overview

Monit Sharma
Indian Institute of Science Education And Research
Mohali

(Dated: July 11, 2021)

With the increase in computing power and new, more efficient algorithms, Machine Learning has become a powerful tool for finding data patterns. Quantum systems have this one advantage over their classical counterpart in that they can produce atypical patterns and hence can outperform classical computers. The field of *Quantum Machine Learning* works on finding different methods to implement the quantum software that helps us in running Machine Learning tasks faster than that on classical computers. Many researchers have built many different algorithms to implement Quantum Machine Learning, but the Quantum Hardware and Quantum Software still possess a challenge.

I. INTRODUCTION

The making of Digital computers in the twentieth century helped us automate various data analysis techniques. The rapid progression in computing power has helped us implement linear algebraic data analysis as regression and Principal Component Analysis and helped us make various complex algorithms like the Support vector Machine. Simultaneously in the rapid advancement of Digital Computers made progress in novel machine learning models. Artificial Neural Networks (Perceptrons) were implemented in the 1950s [1]. Many Deep Learning Models based on the Neural Networks like the Hopfield network and the Boltzmann Machines and training via backpropagation were implemented. In recent times, by combining power computers and particular purpose intended processors, we were capable of implementing Deep Neural Networks with billions of weight parameters and extensive data, and they helped identify complex patterns in the data.

Classical novel Machine Learning models like deep neural networks are equipped with the feature of recognising the statistical patterns in the data and also produce the data that possess the same statistical patterns. So, by this observation, we can suggest that if a small Quantum Information processor can produce some statistical patterns that are difficult for a classical computer to perform computationally, they can perhaps also recognise patterns that are computationally difficult for a Classical computer to find.This whole observation is the basis for the hope of answering whether an efficient Quantum algorithm can be found for Machine Learning.

A quantum algorithm is a set of instructions solving a problem, such as determining whether two graphs are isomorphic, that can be performed on a quantum computer.

Quantum Machine Learning software uses Quantum Algorithms as a part of their implementation. The analysis of quantum algorithms makes it clear that they have the potential to outperform their classical counterpart for specific problems. This potential to outperform is termed Quantum Speed-Up. The idea of a quantum speedup depends on whether one takes a orderly computer science viewpoint—which necessitates mathematical proofs—or a prospect based on what can be done with realistic, finite size devices—which needs solid statistical evidence of a scaling advantage over some finite range of problem sizes. In quantum machine learning, the best achievable execution of classical algorithms is mostly not known. This can be seen in the case of Shor's polynomial-time quantum algorithm for integer factorisation. We do not know any sub-exponential-time classical algorithm, but the possibility is not ruled out.

Resolution of a scaling advantage distinguishing quantum and classical machine-learning would rely on the continuation of a quantum computer.

Moreover, it is called a 'benchmarking' problem. Such advantages could incorporate refined classification accuracy and sampling of classically inaccessible systems. Subsequently, quantum speedups in machine learning are currently described using idealized measures from complexity theory:

- Query complexity
- Gate complexity

Query complexity measures the number of queries to the information source for the classical or quantum algorithm. We say its a quantum speedup results when the number of queries needed to solve a problem is lower for the quantum algorithm than for the classical algorithm.

Gate Complexity is the number of elementary quantum operations (or gates)needed to obtain the desired result.

Query and gate complexity are standardized models that quantify the necessary resources to solve a problem class. Without knowing how to map this idealization to reality, not much can be said about the necessary resource scaling in a real-world scenario. Therefore, the required resources of classical machine learning algorithms are quantified mainly by numerical experimentation. The resource requirements of quantum machine learning algorithms are likely to be similarly challenging to quantify in practice.

II. QUANTUM SPEED-UP

Quantum computers make use of quantum coherence and entanglement to process information in some ways that even classical computers cannot do. The recent years have seen gradual advancement in the construction of powerful quantum computers.

A quantum algorithm is a stepwise method implemented on a quantum computer to resolve a problem, such as exploring a database. Quantum machine learning software makes use of quantum algorithms to process information. Quantum algorithms can, in principle, outperform the best known classical algorithms while solving specific problems. This is known as a *quantum speedup*[2].

For instance, quantum computers can search an unsorted database with N entries in a time proportional to $\sqrt{N}$, where a classical computer, when given black-box access to the the same database takes time proportional to N. Here, the quantum computer shows a square-root speedup over the classical computer. Likewise, quantum computers can make Fourier transforms over N data points, invert sparse $N \times N$ matrices, and find their eigenvalues and eigenvectors in time proportional to a polynomial in $log2N$, where the best-known algorithms for classical computers take time proportional to $N(log2N)$: thus, the quantum computer exhibits an exponential speedup over the best classical computer algorithms.

III. CLASSICAL MACHINE LEARNING

Classical machine learning and data analysis can be classified into certain categories. First, computers can produce 'classic' data analysis methods such as least-squares regression, polynomial interpolation and data analysis. Machine learning rules can be supervised or unsupervised. In supervised learning, the training data are divided into labelled categories, such as samples of handwritten digits together with the actual number the handwritten digit is supposed to represent, and the job of the machine is to learn how to assign labels to data outside the training set. In unsupervised learning, the training set is unlabelled, and the goal of the machine is to find the natural categories into which the training data falls (for example, different types of photos on the internet) and then categorize data outside the training set. Finally, there are machine-learning tasks that involve combinations of supervised and unsupervised learning, together with training sets that maybe generated by the machine itself.

IV. LINEAR-ALGEBRA-BASED QUANTUM MACHINE LEARNING

Most of data analysis and the protocols for the Machine Learning models operate by performing some matrix operations on the vectors in the high-dimensional vector space. Quantum Mechanics is all about the matrix operation on the vectors in high-dimensional vector spaces. The basic behind these methods is that a Quantum state of n quantum bits is a vector in 2^n dimensional complex vector space, and performing a gate or any measurement on that qubit corresponds to multyplying that vector to a $2^n \times 2^n$ matrices. By making such matrices, and implementing them as a transformation the Quantum Computers have showed to perform operations like *Fourier Transform* [3], *finding eigenvectors and eigenvalues* [4] and *solving linear set of equations over* 2^n-*dimesnional vector space* in time which is polynomial in N, which make them exponentially faster than their best classical counterparts. The last one is known as Harrow, Hassidim and Lloyd (HHL) algorithms.[5]

The original variant assumed a well-conditioned matrix that is sparse. Sparsity is unlikely in data science, but later improvements relaxed this assumption to include low-rank matrices. We have several quantum algorithms which appear as subroutines when linear algebra techniques are employed in quantum machine learning software.

A. HHL Algorithm

Systems of linear equations arise naturally in many real-life applications in a wide range of areas, such as in the solution of Partial Differential Equations, the calibration of financial models, fluid simulation or numerical field calculation. The problem can be defined as, given a matrix $A \in \mathbb{C}^{N\times N}$ and a vector $b \in \mathbb{C}^N$, find $x \in \mathbb{C}^N$ satisfying $Ax = b$.

A system of linear equations is called s-sparse if A has at most s non-zero entries per row or column. Solving an s-sparse system of size N with a classical computer requires $O(Ns\kappa \; log(1/\epsilon))$ running time using the conjugate gradient method.[5]. Here κ denotes the condition number of the system and ϵ the accuracy of the approximation.

The HHL is a quantum algorithm to estimate a function of the solution with running time complexity of $O(log(N) \; s^2\kappa^2/\epsilon)$ when A is a Hermitian matrix under the assumptions of efficient oracles for loading the data, Hamiltonian simulation and computing a function of the solution. This is an exponential speed up in the size of the system, however one crucial remark to keep in mind is that the classical algorithm returns the full solution, while the HHL can only approximate functions of the solution vector.

1. *Mathematical background*

The first step towards solving a system of linear equations with a quantum computer is to encode the problem in the quantum language. By rescaling the system, we assume b and x to be normalized and map them both to their respective Quantum states $|b\rangle$ and $|x\rangle$. Now, on the rescaled problem

$$A|x\rangle = |b\rangle$$

Since, A is hermitian, it has a Spectral decomposition

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|$$

where $\lambda_j \in \mathbb{R}$ and $|u_j$ is the j^{th} eigenvector of A with respective eigenvalue λ_j. Then

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j|$$

and the right hand side of the system can be written in the eigenbasis of A as

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle$$

where $b_j \in \mathbb{C}$.

2. *Description*

The algorithm uses three quantum registers, all of them set to $|0\rangle$ at the beginning of the algorithm. One register, which we will denote with the subindex n_l, is used to store a binary representation of the eigenvalues of A. A second register, denoted by n_b contains the vector solution, and from now on $N = 2^{n_b}$. There is an extra register, for the auxiliary qubits. These are qubits used as intermediate steps in the individual computations but will be ignored in the following description since they are set to $|0\rangle$ at the beginning of each computation and restored back to the $|0\rangle$ state at the end of the individual operation.

1. Load the data $|b\rangle \in \mathbb{C}^N$. That is, perform the transformation

$$|0\rangle_{nb} \rightarrow |b\rangle_{nb}$$

2. Apply Quantum Phase Estimation (QPE) with

$$U = e^{iAt} = \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle\langle u_j|$$

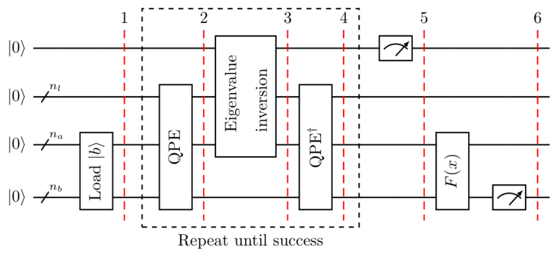

FIG. 1. HHL Circuit

The quantum state of the register expressed in the eigenbasis of A is now

$$\sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{nl} |u_j\rangle_{nb}$$

where $|\lambda_j\rangle_{nl}$ is the nl bit binary representation of λ_j

3. Add an auxiliary qubit and apply a rotation conditioned on $|\lambda_j\rangle$

$$\sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{nl} |u_j\rangle_{nb} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right)$$

where C is a normalization constant, and as expressed in the current form above, should be less than the smallest eigenvalue λ_{min} in magnitude.

4. Apply $QPE^\dagger$. Ignoring possible errors from QPE, this results in

$$\sum_{j=0}^{N-1} b_j |0\rangle_{nl} |u_j\rangle_{nb} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right)$$

5. Measure the auxiliary qubit in the computational basis. if the outcome is 1, the register is in the post-measurement state, which up to a normalization factor corresponds to the solution.

6. Apply an observable M to calculate $F(x) = \langle x|M|x\rangle$.

Check the code for HHL algorithm implementation on Github

V. QUANTUM PRINCIPLE COMPONENT ANALYSIS

In principal component analysis (PCA), suppose that the data are presented in the form of vectors v_j in a d-dimensional vector space, where $d = 2^n = N$.

For example, v_j could be the vector of changes in prices of all stocks in the stock market from time t_j to time t_{j+1}. The covariance matrix of the data is $C = \sum_j v_j v_j^T$, where superscript T denotes the transpose operation: the covariance matrix summarizes the correlations between the different components of the data, for example, correlations between changes in the prices of different stocks. In its simplest form, principal component analysis operates by diagonalizing the covariance matrix:$C = \sum_k e_k c_k c_k^\dagger$, where the c_k are the eigenvectors of C, and e_k are the corresponding eigenvalues. If only a few of the eigenvalues c_k are large, and the remainder are small or zero, then the eigenvectors corresponding to those eigenvalues are called the principal components of C. Each principal component represents an underlying common trend or form of correlation in the data, and decomposing a data vector v in terms of principal components, $v = \sum_k v_k c_k$, allows one both to compress the representation of the data and to predict future behaviour. Classical algorithms for performing PCA scale as $O(d^2)$ in terms of computational complexity and query complexity.

For quantum principal component analysis of classical data, we choose a data vector v_j at random and use a quantum random access memory (qRAM)[6] to map that vector into a quantum state: $v_j \rightarrow |v_j\rangle$. The quantum state that summarizes the vector has $log\ d$ qubits, and the operation of the qRAM requires $O(d)$ operations divided over $O(log\ d)$ steps that can be performed in parallel. Because v_j was chosen at random, the resulting quantum state has a density matrix $\rho = \frac{1}{N}\sum_j |v_j\rangle\langle v_j|$, where N is the number of data vectors. By comparing the covariance matrix C for the classical data, we see that the density matrix for the quantum version of the data is the covariance matrix, up to an overall factor. By repeatedly sampling the data and using a trick called density matrix exponentiation combined with the quantum phase estimation algorithm, which finds eigenvectors and eigenvalues of matrices, we can take the quantum version of any data vector $|v\rangle$ and decompose it into the principal components $|ck\rangle$, revealing the eigenvalue of C at the same time. The properties of the principal components of C can then be probed by making measurements on the quantum representation of the eigenvectors of C. The quantum algorithm scales as $O[(logN)^2]$ in computational complexity and query complexity. That is, quantum PCA is exponentially more efficient than classical PCA.

VI. QUANTUM SUPPORT VECTOR MACHINES AND KERNEL METHODS

The simplest examples of supervised machine learning algorithms are linear support vector machines and perceptrons. These methods seek to find an optimal separating hyperplane between two classes of data in a dataset such that, with high probability, all training examples of one class are found only on one side of the hyperplane. The most robust classifier for the data is given when the hyperplane margin and the data are maximized. Here the 'weights' learned in training are the parameters of the hyperplane. One of the most extraordinary powers of the support vector machine lies in its generalization to nonlinear hypersurfaces via kernel functions. Such classifiers have found great success in image segmentation as well as in the biological sciences. Like its classical counterpart, the quantum support vector machine is a paradigmatic example of a quantum machine learning algorithm. The first quantum support vector machine was discussed in the early 2000s. , Using a variant of Grover's search for function minimization. Finding s support vectors out of N vectors consequently takes$\sqrt{N/s}$ iterations. Recently, a least-squares quantum support vector machine was developed that harnesses the full power of the qBLAS subroutines. The data input can come from various sources, such as qRAM accessing classical data or a quantum subroutine preparing quantum states. Once the data are made available to the quantum computing device, they are processed with quantum phase estimation and matrix inversion (the HHL algorithm). All the operations required to construct the optimal separating hyperplane and to test whether a vector lies on one side or the other can in principle be performed in time that is polynomial in $logN$, where N is the dimension of the matrix required to prepare a quantum version of the hyperplane vector. Polynomial and radial basis function kernels are discussed, and another kernel-based method called the Gaussian process regression. This approach to quantum support machines has been experimentally demonstrated in a nuclear magnetic resonance testbed for a handwritten digit recognition task.

Check Code for Quantum Support Vector Machine on Github

VII. SUPERVISED LEARNING WITH QUANTUM COMPUTERS

We use a typology introduced by Aimeur, Brassard and Gambs. It distinguishes four approaches of how to combine quantum computing and machine learning, depending on whether one assumes the data to be generated by a quantum (Q) or classical (C) system, and if the information processing device is quantum (Q) or classical (C).

CC

The case CC refers to classical data being processed classically. This is of course the conventional approach to machine learning, but in this context it relates to machine learning based on methods borrowed from quantum information research. An example is the application of tensor networks, which have been developed for quantum many-body-systems, to neural network training.

QC

The case QC investigates how machine learning can help with quantum computing. For example, when we want to get a comprehensive description of the internal state of a quantum computer from as few measurements as possible we can use machine learning to analyze the measurement data [12]. Another idea is to learn phase transitions in many-body quantum systems, a fundamental physical problem with applications in the development of quantum computers.

CQ

The CQ setting, which uses quantum computing to process classical datasets. The datasets consist of observations from classical systems, such as text, images or time series of macroeconomic variables, which are fed into a quantum computer for analysis. This requires a quantum-classical interface, which is a challenge.

QQ

QQ, looks at 'quantum data' being processed by a quantum computer. This can have two different meanings. First, the data could be derived from measuring a quantum system in a physical experiment and feeding the values back into a separate quantum processing device. A much more natural setting however arises where a quantum computer is first used to simulate the dynamics of a quantum system (as investigated in the discipline of quantum simulation and with fruitful applications to modeling physical and chemical properties that are otherwise computationally intractable), and consequently takes the state of the quantum system as an input to a quantum machine learning algorithm executed on the very same device. The advantage of such an approach is that while measuring all information of a quantum state may require a number of measurements that is exponential in the system size, the quantum computer has immediate access to all this information and can produce the result, for example a yes/no decision, directly—an exponential speedup by design.

A. Quantum Computing for Supervised Learning

Two strategies for designing a Quantum Machine Learning Algorithm:

1. The first strategy aims at translating classical models into the language of quantum mechanics in the hope to harvest algorithmic speedups. The sole goal is to reproduce the results of a given model, say a neural net or a Gaussian process, but to 'outsource' the computation or parts of the computation to a quantum device. The translational approach requires significant expertise in quantum algorithmic design. The challenge is to assemble quantum routines that imitate the results of the classical algorithm while keeping the computational resources as low as possible.
2. The second strategy, whose many potential directions are still widely unexplored, leaves the boundaries of known classical machine learning models. Instead of starting with a classical algorithm, one starts with a quantum computer and asks what type of machine learning model might fit its physical characteristics and constraints, its formal language and its proposed advantages. Hence the name the "Exploratory Approach".

B. How do Quantum Computers classify Data?

We will look at how to implement a type of nearest neighbour method with quantum interference induced by a Hadamard gate.

1. *The Squared-Distance Classifier*

Machine learning always starts with a dataset. Inspired by the kaggle Titanic dataset, let us consider a set of 2-dimensional input vectors $\{x^m = (x_0^m, x_1^m)^T\}, m = 1, ..., M$. Each vector represents a passenger who was on the Titanic when the ship sank in the tragic accident of 1912, and specifies two features of the passenger: The price in dollars which she or he paid for the ticket (feature 0) and the passenger's cabin number (feature 1). Assume the ticket price is between $0 and 10{,}000$, and the cabin numbers range from 1 to 2,500. Each input vector x_m is also assigned a label y_m that indicates if the passenger survived ($y_m = 1$) or died ($y_m = 0$).

We consider a dataset of only 2 passengers, one who died and one who survived the event. The task is to find the probability of a third passenger to die or survive the crash. We preprocess the data in order to project it onto roughly the same scales. Oftentimes, this is done by imposing zero mean and unit variance, but here we will simply rescale the range of possible ticket prices and cabin numbers to the interval [0, 1] and round the values to two decimal digits.

Possibly the simplest supervised machine learning method, which is still surprisingly successful in many cases, is known as nearest neighbour.

Closeness has to be defined by a distance measure, for example the Euclidean distance between data points.

We will measure the weight for each data point in the dataset, such as the point nearer to the given data point , has much more importance than the others.

$$\gamma_m = 1 - \frac{1}{c}|\tilde{x} - x^m|^2$$

Here, c is some constant, $\tilde{x}$ is the given data point for us to predict, and x^m is the data from the dataset

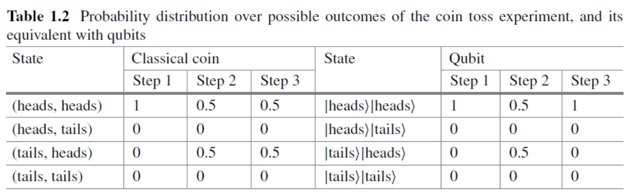

Table 1.2 Probability distribution over possible outcomes of the coin toss experiment, and its equivalent with qubits

State	Classical coin			State	Qubit		
	Step 1	Step 2	Step 3		Step 1	Step 2	Step 3
(heads, heads)	1	0.5	0.5	|heads⟩|heads⟩	1	0.5	1
(heads, tails)	0	0	0	|heads⟩|tails⟩	0	0	0
(tails, heads)	0	0.5	0.5	|tails⟩|heads⟩	0	0.5	0
(tails, tails)	0	0	0	|tails⟩|tails⟩	0	0	0

FIG. 2. The Data Table (Source : www.kaggle.com)

We define the probability of assigning label $\tilde{y}$ to the new input $\tilde{x}$ as the sum over the weights of all M_1 training inputs which are labeled with $y_m = 1$

$$p(\tilde{y} = 1) = \frac{1}{\chi}\frac{1}{M_1}\sum_{m|y^m=1}(1 - \frac{1}{c}|\tilde{x} - x^m|^2)$$

The probability of predicting label 0 for the new input is the same sum, but over the weights of all inputs labeled with 0. The $\frac{1}{\chi}$ factor is for making sure the probabilities sum remain 1.

2. *Interference with the Hadamard Transformation*

Most quantum computers are based on a mathematical model called a qubit, which can be understood as a random bit (a Bernoulli random variable) whose description is not governed by classical probability theory but by quantum mechanics.

We will refer to the two random bits as two coins that can be tossed, and the quantum bits can be imagined as quantum versions of these coins.

Step 1, turn both coins to 'heads'. In Step 2 toss the first coin only and check the result. In Step 3 toss the first coin a second time and check the result again. Consider repeating this experiment from scratch a sufficiently large number of times to count the statistics, which in the limiting case of infinite repetitions can be interpreted as probabilities

The first three columns of Table show these probabilities for our little experiment. After the preparation step 1 the state is by definition (heads, heads). After the first toss in step 2we observe the states (heads, heads) and (tails, heads)with equal probability. After the second toss in step 3, we observe the same two states with equal probability, and the probability distribution hence does not change between step 2 and 3.

Compare this with two qubits q_1 and q_2. Again, performing a measurement called a projective z-measurement a qubit can be found to be in two different states (let us stick with calling them heads and tails, but later it will be $|0\rangle$ and $|1\rangle$). Start again with both qubits being in state heads, heads. This means that repeated measurements would always return the result heads, heads, just as in the classical case.

Now we apply an operation called the Hadamard transform on the first qubit, which is sometimes considered as the quantum equivalent of a fair coin toss. Measuring the qubits after this operation will reveal the same probability distribution as in the classical case, namely that the probability of heads, heads and tails, heads is both 0.5. However, if we apply the 'Hadamard coin toss twice without intermediate observation of the state, one will measure the qubits always in state (heads, heads), no matter how often one repeats the experiment

In linear algebra, a transformation between probability vectors can always be described by a special matrix called a stochastic matrix, in which rows add up to 1.

C. Quantum Embedding

A quantum embedding represents classical data as quantum states in a Hilbert space via a quantum feature map. It takes a classical data point x

and translates it into a set of gate parameters in a quantum circuit, creating a quantum state $|\psi_x\rangle$

This process is a crucial part of designing quantum algorithms and affects their computational power.

Let's consider classical input data consisting of M examples, with N features each,

$D = \{x^{(1)}, x^{(2)}, ..., x^{(m)}........, x^{(N)}\}$

where $x^{(m)}$ is a N-dimensional vector for m=1,. . . ,M. To embed this data into n quantum subsystems (n qubits for discrete- and continuous-variable quantum computing, respectively), we can use various embedding techniques, some of which are explained briefly below.

1. *Basis Embedding*

Basis embedding associates each input with a computational basis state of a qubit system. Therefore, classical data has to be in the form of binary strings. The embedded quantum state is the bit-wise translation of a binary string to the corresponding states of the quantum subsystems. For example, $x = 1001$ is represented by the 4-qubit quantum state $|1001\rangle$

Hence, one bit of classical information is represented by one quantum subsystem.

Let's consider the classical Dataset D mentioned above. For basis embedding, each example has to be a N-bit binary string:

$x^{(m)} = (b_1, ..., b_N)$ with $b_i \in \{0, 1\}$ for $i = 1, ..., N$. Assuming all features are represented with unit binary precision, each input example $x^{(m)}$ can be directly mapped to the quantum state $|x^{(m)}\rangle$.

This means that the number of Quantum Subsystems, n , must be equal to N

For Eg: let's say we have a classical dataset containing two examples $x^{(1)} = 01$ and $x^{(2)} = 11$. The corresponding basis encoding uses two qubits to represent $|x^{(1)}\rangle = |01\rangle$ and $|x^{(2)}\rangle = |11\rangle$ resulting in

$$|D\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

2. *Amplitude Embedding*

In the amplitude-embedding technique, data is encoded into the amplitudes of a quantum state.

A normalized classical $N-$dimensional data point x is represented by the amplitudes of a $n-$qubit quantum state $|\psi_x\rangle$ as

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i|i\rangle$$

where $N = 2^n$, x_i is the i-th element of x , and $|i\rangle$ is the i-th computational basis state. In this case, however x_i can have different numeric data types, e.g integer or floating point.

For Example: Let's say we want to encode the four-dimensional floating point array $x = (1.0, 0.0, -5.5, 0.0)$ using amplitude embedding.

The first step is to normalize it, i.e

$$x_{norm} = \frac{1}{\sqrt{31.25}}(1.0, 0.0, -5.5, 0.0)$$

The corresponding amplitude encoding uses two qubits to represent x_{norm} as

$$|\psi_{x_{norm}}\rangle = \frac{1}{\sqrt{31.25}}[|00\rangle - 5.5|10\rangle]$$

VIII. PERSPECTIVE ON FUTURE WORK

As we have seen in this overview, small quantum computers and large quantum simulators anneal and more seem to be used in machine learning and data analysis. However, the development of quantum algorithms requires quantum hardware that is not yet available. Quantum machine learning provides a list of possible applications for small quantum computers compiled and developed by specialized processes of quantum data processing, digital quantum processors and sensors. Hardware challenges are technically inherent, and straightforward ways are available to overcome them. However, they must be overcome if quantum machine learning is going to be a 'killer app' for quantum computers. As mentioned earlier, most of the quantum algorithms identified are subject to several limitations.

[1] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65, 386 (1958).

[2] Rønnow, T. F. et al. Defining and detecting quantum speedup. Science 345, 420–424 (2014).

[3] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26, 1484–1509 (1997

[4] Nielsen, M. A. and Chuang, I. L. Quantum Computation and Quantum Information (Cambridge Univ. Press, 2000).

[5] Harrow, Aram W; Hassidim, Avinatan; Lloyd, Seth (2008). "Quantum algorithm for solving linear systems of equations". Physical Review Letters. 103 (15): 150502. arXiv:0811.3171. Bibcode:2009PhRvL.103o0502H.

[5] Hestenes, Magnus R.; Stiefel, Eduard (December 1952). "Methods of Conjugate Gradients for Solving Linear Systems" (PDF). Journal of Research of the National Bureau of Standards. 49 (6): 409. doi:10.6028/jres.049.044.

[6] Giovannetti, V., Lloyd, S. and Maccone, L. Quantum random access memory. Phys. Rev. Lett. 100, 160501 (2008).