Quantum Machine Learning for Finance

ICCAD Special Session Paper

Marco Pistoia, Syed Farhan Ahmad, Akshay Ajagekar, Alexander Buts, Shouvanik Chakrabarti, Dylan Herman, Shaohan Hu, Andrew Jena, Pierre Minssen, Pradeep Niroula, Arthur Rattew, Yue Sun, Romina Yalovetzky
Future Lab for Applied Research and Engineering, JPMorgan Chase Bank, N.A.

2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 978-1-6654-4507-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCAD51958.2021.9643469

***Abstract*—Quantum computers are expected to surpass the computational capabilities of classical computers during this decade, and achieve disruptive impact on numerous industry sectors, particularly finance. In fact, finance is estimated to be the first industry sector to benefit from Quantum Computing not only in the medium and long terms, but even in the short term. This review paper presents the state of the art of quantum algorithms for financial applications, with particular focus to those use cases that can be solved via Machine Learning.**

***Index Terms*—Quantum Computing, Machine Learning, Artificial Intelligence, Algorithms, Finance**

I. Introduction

The computational power of a quantum computer grows exponentially with its number of qubits. For this reason, quantum computers are expected to surpass the computational capabilities of classical computers and achieve disruptive impact on numerous industry sectors, such as global energy and materials, pharmaceuticals, telecommunication, travel and logistics, and finance. Finance, in particular, is estimated to be the first industry sector to benefit from quantum computing not only in the medium and long terms, but even in the short term due to the large number of financial use cases that lend themselves to quantum computing and their amenability to be solved effectively even in the presence of approximations [1]. This is especially important for taking advantage of today's Noisy Intermediate-Scale Quantum (or NISQ) devices [2], which are characterized by their low quantum bit (or qubit) counts, short coherence time, and high operation noise.

This review paper presents the state of the art of quantum algorithms for financial applications, focusing in particular on those use cases that can be solved via Machine Learning (ML). The applicability of ML to finance has become increasingly more significant as highly efficient ML algorithms have evolved over time to support different data types and scale to larger data sets. ML operations applicable to finance include regression for asset pricing, classification for portfolio optimization, clustering for portfolio risk analysis and stock selection, generative modeling for market regime identification, feature extraction for fraud detection, reinforcement learning for algorithmic trading, and Natural Language Processing (NLP) for risk assessment, financial forecasting and accounting and auditing. Deep learning is often used for image recognition and text classification, as well as in any use case characterized by large unstructured datasets.

Given the complexity of the algorithms involved, and the size of the data being analyzed, ML has been identified as one of the most important domains of applicability of Quantum Computing. This has become even more evident with the discovery of new quantum algorithms for linear algebra, which offer the potential for executing linear-algebra computations on a quantum computer more efficiently and accurately than their corresponding classical counterparts [3]. Under certain conditions, the quantum speedup can be even exponential [4], modulo some caveats [5]:

- Efficiently loading classical data onto quantum computers and reading out classical outputs resulting from quantum computations is still the field of ongoing research. The majority of the quantum algorithms devised so far is based on the existence of a Quantum Random Access Memory (QRAM) for accessing the classical data [6]. The realization of a QRAM has been theoretically proven, but concrete hardware implementations are still undergoing. Alternatively, classical data can be loaded into a quantum states via specialized circuits [7].
- It is not always possible to apply a quantum linear-algebra algorithm out of the box to solve a specific financial use case; several conditions must be met and customizations are often necessary to address unique use-case-dependent requirements [5]. Furthermore, multiple classical and quantum algorithmic components are usually involved in the end-to-end solution of a financial use case, with the potential for any such component to become the bottleneck and negate the overall quantum advantage. The task of computing the quantum speedup of the solution of a specific use case is, therefore, not always intuitive.

As of today, no end-to-end application of quantum ML with exponential speedup over its classical counterpart has been discovered, but several promising directions have been proposed. Meanwhile, a large body of research and engineering work has been successfully dedicated to the realization of quantum algorithms with significant polynomial speedups in their data-processing subroutines, if not in the data loading and output extraction.

II. Regression

A central task in supervised learning is *regression*, or the problem of training a simple model to approximate real-valued functions. This is an extremely important routine in experimental sciences, and has recently started to be employed on massive datasets. Computationally, this reduces to minimizing a loss function that captures the quality of the fit on training data; the common choices are the L_∞ norm for worst-case error, and the L_1 or L_2 norms for average-case error. The smoothness of the L_2 norm makes it attractive to optimization algorithms, leading to the ubiquity of least squares regressions, where the minimization problem, given N d-dimensional feature vectors, can be expressed as follows:

$$\arg\min_{\vec{\beta}} \sum_{i=1}^{N} w_i \left(y^{(i)} - \vec{\beta}^T \vec{x}^{(i)} \right)^2,$$

which is a convex quadratic minimization problem and reduces to solving an $N \times d$ system of linear equations.

A quantum speedup for this task is first considered by Wiebe *et al.* [8] based on quantum algorithms for solving systems of linear equations. Their algorithm requires access to the entries of the data matrix and weight vector in superposition, and outputs a quantum state that encodes the output in time $\tilde{O}\left(\log(N) s^3 \kappa^6 / \epsilon\right)$, where $\tilde{O}(\cdot)$ neglects polylogarithmic factors in the complexity. While the run time

978-1-6654-4507-8/21/$31.00 ©2021 IEEE

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

of this algorithm is exponentially smaller than classical equivalents, there are two main caveats. First, the construction of a data structure allowing superposition access to the data points may in general require $O(N\log(N))$ time, via the construction of a QRAM [6]. Second, $\sim O(d)$ copies of the output state are required to obtain a full description of the output.

Some attempts have been made to address this issue, albeit with smaller speedups. Wang [9] gives an algorithm for L_2 regression that obtains a classical solution in time $\mathrm{poly}(\log(N), d, \kappa, 1/\epsilon)$. A second approach is to use algorithms that do not output a full description of $\vec{\beta}$, but rather $\vec{\beta}^T\vec{x}$ for a new data point $\vec{x}$. Schuld *et al.* [10] obtain such an algorithm given a quantum state encoding of the training data and $\vec{x}$ that takes time $O(\log(N)\kappa^2/\epsilon^3)$. A third approach is to consider special quantum data structures for accessing the training data. These *efficient* QRAM data structures are different from a general QRAM, in that the cost of inserting, updating or deleting a single entry is $O(\mathrm{poly}(\log(n)))$. Such data structures can be useful when there is a large initial corpus of data, but the regression task must be performed repeatedly, and each updates to the data set is small (or constant). The construction of the original dataset thus has cost $O(N\log(N))$, but future upkeep and regression tasks each cost $O(\mathrm{poly}(\log(n,d)))$. Thus, the cost of the initial construction is effectively amortized over the lifetime of the deployment. This setting can be used to obtain L_2-regression algorithms with complexity $\tilde{O}\left(\kappa\mu/\epsilon(\mathrm{poly}(\log(Nd)))\right)$ [11, 12].

There also exist quantum-inspired classical algorithms [13] based on the last approach that use data structures providing sampling access to the data and obtain algorithms with cost $\tilde{O}\left(\|X\|_F^6\|X\|^2/\epsilon^4\right)$. L_1 and L_∞ regression problems are not smooth and are therefore often solved via smooth relaxations that are problem-specific.

A heuristic approach to regression can be used by simply training parameterized quantum circuits (PQC) as function approximators. A common application of classical regression techniques is in learning time series via Recurrent Neural Networks (RNNs) (often using Long Short Term Memory (LSTM) units). These techniques are commonly used to make predictions about evolving processes from historical data. Quantum versions of RNNs have been proposed that use PQCs as the function model [14–16] that empirically anticipate improvements in convergence or error rate.

With the above discussion on various quantum regression algorithms, we next look at several financial applications that take advantage of these techniques.

A. Asset Pricing

Asset Pricing is the task of assigning prices to various categories of financial instruments, such as stocks, bonds, and derivatives. There are several economic models used to assign these prices based on a sequence of instantaneous (or *spot*) prices, including general equilibrium pricing and arbitrage-free pricing [17]. The common approach to predicting these spot prices is to model them as functions of simple underlying stochastic processes, such as Brownian or Geometric Brownian motion. Historical financial data can then be used to determine the parameters of these stochastic models. More generally, however, predicting spot prices (as well as many other time-varying quantities of financial interest) can be modeled as a *time series* learning problem. Specifically, given a sequence of historic prices up to time t, can accurate predictions be made for prices in the future? Since this reduces to predicting real values based on training data, it is best modeled as a supervised-regression problem. Stochastic pricing models with historically calibrated parameters can be viewed as *ad hoc* solutions based on domain knowledge.

The increasing success of deep RNNs for time-series prediction—especially those that leverage LSTM—has led researchers to consider using these general-purpose algorithms for asset pricing. Gu *et al.* [18] and Chen *et al.* [19] investigate the use of LSTM-based deep-learning methods for asset pricing, obtaining promising results. Specifically, Gu *et al.* [18] show that ML forecasts on the S&P 500 achieve an out-of-sample annualized Sharpe Ratio of 0.77 versus the 0.51 of a buy-and-hold investor. They also find that a value-weighted long-short decile spread strategy based on neural network forecasts of stock prices achieve an annualized Sharpe Ratio of 1.35, nearly double the state-of-the-art classical regression approaches. Chen *et al.* further show that refined models based on LSTM forecasts can achieve out-of-sample annualized Sharpe Ratios much larger than naive deep-learning forecasts as well as classical approaches, including the Fama-French five-factor model.

The biggest challenge in deploying deep-learning methods is that training complex neural networks can often be a much more computationally intensive process than the simple parameter calibration required by classical approaches. PQCs may offer advantages over classical variational regression models in terms of expressivity, training complexity and prediction performance. In 2020, the use of PQCs to formulate RNNs has been described [14], along with proposals of quantum LSTM models [15]. Both approaches show potential empirical improvements over classical neural networks for particular functions, although the applicability to asset pricing is yet to be investigated.

B. Multi-Asset Trend Following Strategies

Regression models can be used to predict 1-day returns of a multi-asset class portfolio. Each financial asset class (e.g., equity, bond, cash or commodities) might have different internal dynamics. Nevertheless, a regression model might be able to encompass the global dynamic. An example of global dynamic is the following: if the equity market is bearish, an investor might prefer safer investments, say, bonds, and therefore cause a rise in bond prices. Indeed, there has been a negative correlation between equity and bond since the beginning of the century. However, this correlation was positive between 1970 and 2000 [20]. Therefore, the relationship between asset classes evolves and should not be taken as general truth.

To predict returns daily, one can use historical prices from various time points (e.g., 1 month, 3 months, etc.) to introduce trend information in the input data. However, this causes an increase in the number of features in the data, and could result in over-fitting. Consequently, one would rather use a Lasso regression [21] than a vanilla regression. Indeed, by adding an L_1 penalty term $\lambda\|\vec{\beta}\|_1$ to the cost function of the L_2 regression problem, the model will select a subset of relevant features. As with any regression method, one can treat it as a classification model. The classes here would be buy, hold or sell. The L_1 norm may not be the most attractive regularization term for quantum implementation. Nevertheless, Du *et al.* [22] provide the closest known implementation with a differentially private Lasso estimator.

C. Implied-Volatility Estimation

The *Implied volatility* metric captures the financial market's view of the likelihood of changes in a given security's price. The analysis of volatility is crucial for risk management, portfolio hedging and option pricing. A precise notion of the market's expectation of volatility is required [23]. Portfolios have a sensitivity with respect to volatility

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

changes. For instance, it has been proven that implied volatilities—such as those of oil, gold and the US stock market—play a role on the returns of the equity sector. Particularly, their impact on the prices and returns of the ten most representative US equity sectors has been quantified [24].

A quantum approach for learning implied volatilities has been proposed [25]. This uses the deep quantum neural networks firstly introduced by Beer *et al.* [26]. Given N options, the input data is its strike prices $K_1, K_2, \ldots, K_N$, and the output is the implied volatilities $\sigma_{K_1}, \sigma_{K_2}, \ldots, \sigma_{K_N}$. A sigmoid function is used to convert the strike prices to numbers in $[0, 1]$, which are then represented as quantum states $\left|\phi_n^{in}\right\rangle$ for $n = 1, 2, \ldots, N$. The network consists of one input neuron, one output neuron, and one hidden layer with two neurons. The output of the network is a density matrix $\rho_n^{out} = \left|\phi_n^{out}\right\rangle \left\langle\phi_n^{out}\right|$. The implied volatility of each of the N options is then calculated using its respective element in the density matrix.

III. Classification

The goal of *classification* in ML is to predict the labels for new data points using a model that is fit by a labeled dataset. Well-known traditional classification algorithms include Linear Classification, Nearest Centroid and Support Vector Machines (SVMs). More recently, neural-network-based methods have seen tremendous success. Once a neural network has had its weights trained via a labeled dataset, it can be used to perform inference on unseen data instances. It has been empirically demonstrated that neural networks achieve better performance than traditional methods, especially on large datasets [27]. Neural networks, on the other hand, fall short in terms of transparency and interpretability, which could be desirable when it comes when making decisions or performing tasks that involve handling sensitive information [28].

a) Linear Classification: A *linear classifier* allows for classifying an object in a dataset based on the value of a linear combination of that object's characteristics, known as *feature values* and stored in a *feature vector*. The common algorithm used for this is the *perceptron* method [29], which finds a γ margin classifier given n d-dimensional data points in time $O(nd/\gamma^2)$. Quantum algorithms have been shown to be able to provide speedup, bringing the running time down to $O(\sqrt{n}d/\gamma^2)$ [30] or $O(\sqrt{nd}/\gamma)$ [31]. It was later discovered that the optimal classical algorithm for training classifiers with constant margin runs in $\tilde{O}(n + d)$ [32], and that a corresponding optimal quantum classification algorithm can bring about quadratic speedup, leading to running time $\tilde{O}(\sqrt{n} + \sqrt{d})$ [33].

b) Distance-based Classifiers: Distance-based classifiers predict the label of a new data point based on its distance to reference points according to some metric. Typical examples are the *nearest-centroid* and *k-nearest-neighbors* (k-NN) classifiers [34].

The nearest-centroid algorithm is a good baseline classifier that offers interpretable results. This algorithm takes as input a number of labeled data points, where each data point belongs to a specific class. The model fitting consists of computing the *centroids*, which are the barycenters of data points that cluster together in space. Once the centroids are found, a new data point is classified by finding its closest centroid in terms of Euclidean distance. A quantum version of the nearest centroid algorithm was used to perform classification on the Modified National Institute of Standards and Technology (MNIST) handwritten-digit dataset [28]. The experiments were executed on the IonQ trapped-ion quantum computer. This approach utilizes novel quantum procedures for loading the classical data onto quantum states and estimating distances between these states. Its accuracy matches that of the classical nearest-centroid algorithm. The authors, however, do not claim any quantum speedup in terms of time complexity.

Another distance-based classifier is the k-NN algorithm, which predicts the label of a data point based on a majority vote among k closest training samples according to a metric, such as the Euclidean distance. Several quantum approaches for k-NN have been proposed. In particular, [35] utilizes an algorithm for computing Hamming distances in superposition, and the quantum minimum-finding method [36] to find the neighbors with the smallest Hamming distances to a sample. If feature vectors lie in a low-dimensional space, the algorithm can classify a new sample with worst-case time complexity $O(\sqrt{M}\log(M))$, where M is the training set size. Another quantum k-NN approach achieves a query complexity of $O(\sqrt{kM})$ [37]. The authors design an oracle that encodes the fidelity between two states into a quantum register, which allows for the usage of a quantum algorithm for finding k-minima [38].

c) Support Vector Machines: An *SVM* [39] consists of solving a convex quadratic optimization problem to find the hyperplane that results in the maximum margin between two classes of data. The dual problem

$$\max_{\alpha_i} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k K(x_i, x_k) \mid \alpha_i \geq 0 \qquad (1)$$

is the one usually solved. $K : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ is symmetric and positive semi-definite; a common example is the *Radial Basis Function*. K induces the, potentially *non-linear*, feature map $\phi(x) = K(\cdot, x) = K_x$. The Hilbert Space built from such maps is called the *Reproducing Kernel Hilbert Space* (RKHS) with reproducing kernel K. The optimal classifier in RKHS is one that is a linear combination of K_x's over a subset of the training data [34].

One of proposed quantum enhancements to the SVM is based on evidence that universal quantum computation, most likely, cannot be efficiently simulated on a classical computer [40]. Thus, one should be able to construct a quantum circuit for the map $\vec{x} \mapsto \mathcal{U}_{\Phi(\vec{x})} |0\rangle$; $\mathcal{U}_{\Phi(\vec{x})}$ is a unitary operation applied to the computational basis state consisting of all qubits in the $|0\rangle$ state, such that this operation is not classically feasible. This is called a *quantum feature map* and maps classical data $\vec{x}$ into a quantum Hilbert Space that is exponentially large in the dimension of $\vec{x}$. A potential quantum kernel is

$$K(x_i, x_j) = \| \langle 0 | \mathcal{U}^\dagger_{\Phi(\vec{x_i})} \mathcal{U}_{\Phi(\vec{x_j})} |0\rangle \|^2$$

which is symmetric and positive semi-definite. In this case the RKHS is spanned by the functionals, $K(\cdot, x)$, that are constructed from quantum circuits. The coefficients α_i of the decision function in RKHS can be computed by a convex optimizer running on a classical computer; the quantum computer is used to evaluate the kernel. This hybrid model is called QSVM [40]. There is potential quantum advantage in the expressability of the feature map, as long as the associated kernel is infeasible for a classical device to compute. This kernel can be computed on a quantum device utilizing either the *destructive SWAP* [41] or the *controlled-SWAP* [42] tests. The latter has better asymptotic complexity, but is not as feasible on NISQ devices. While the former's asymptotic scaling is prohibitive, it can efficiently be implemented on small quantum computers; this allows for experimentation in the near-term.

The class of *Instantaneous Quantum Polynomial* (IQP) circuits has been suggested as a potential candidate for $\mathcal{U}_{\Phi(\vec{x})}$ [40]:

$$\begin{aligned} \mathcal{U}_{\Phi(\vec{x})} &= U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} \\ U_{\Phi(\vec{x})} &= \exp\left(i \textstyle\sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right) \end{aligned} , \qquad (2)$$

where the functions ϕ_S represent classical preprocessing. If the IQP

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

circuit is deep enough, it is believed that computing inner products of states resulting from these embeddings is #P-Hard [40], thereby potentially out of reach of classical devices.

A secondary consideration is whether quantum algorithms can be used to accelerate the training of classical SVMs. Algorithms of complexity $\tilde{O}(\sqrt{n}+\sqrt{d})$ has been proposed for training kernel classifiers and ℓ_2 margin SVMs [33]. These algorithms are optimal and provide a quadratic speedup over corresponding optimal classical algorithms [32]. However these algorithms have complexity polynomial in the inverse of the error $(1/\epsilon)$. There exist classical algorithms with $O(\text{poly}(\log(1/\epsilon)))$ complexity using interior-point methods; Kerenidis *et al.* [43] propose a quantum algorithm to speed up these methods in terms of the dimension d. While the quantum run time depends on terms that are difficult to bound directly, for random instances, the quantum algorithm can indeed provide a speedup, leading to a $O(n^{2.59})$ complexity, compared to the $O(n^{3.11})$ classical algorithm's complexity.

In SVM, an optimal hyperplane is obtained that divides the dataset into multiple classes, with a time complexity of $O(\log(1/\epsilon)\text{poly}(N,M))$, where N represents the feature space dimension, M the number of input points, and ϵ the accuracy. QSVMs have mathematically been proven to have a run time of $O(\log(NM))$ [44].

d) Variational Quantum Classifiers: Variational Quantum Classifiers (VQCs) are hybrid quantum-classical ML architectures meant for classification tasks that utilize the quantum state space as a feature space to potentially obtain a quantum advantage. A VQC circuit mainly consists of a quantum embedding, a PQC for processing the quantum data, a measurement routine, and a classical optimization loop for updating the parameters of the PQC. First, classical input data $\vec{x}$ is mapped to a quantum state non-linearly using the feature-map circuit, $\mathcal{U}_{\Phi(\vec{x})}$, defined in Equation 2. Applying $\mathcal{U}_{\Phi(\vec{x})}$ to $|0\rangle^n$ results in the state $|\Phi(\vec{x})\rangle$.

Next, a PQC, $W(\vec{\theta})$, is constructed with parameters $\vec{\theta}$. An example of such a PQC is one made from compositions of single qubit rotations and entangling gates. PQC architectures have been discussed where descriptors, such as the entangling capability and expressibility, are used to characterize the performance of the PQCs [45].

In case of a binary-classification problem, a measurement routine is used to get a binary output. This is accomplished by measuring state $W(\vec{\theta})\mathcal{U}_{\Phi(\vec{x})}|0\rangle^n$ in the Pauli Z-basis and mapping the output bit-string to a function with binary outcome $f:\{0,1\}^n \rightarrow \{+1,-1\}$. The probability of obtaining an outcome, $y=\pm 1$, is

$$p_y(\vec{x}) = \sum_{i \in f^{-1}(y)} \| \langle i | W(\vec{\theta}) | \Phi(\vec{x}) \rangle \|^2.$$

We repeat this step for R measurement shots, which gives an empirical distribution, $\hat{p}_y(\vec{x})$.

Then, a classical cost function is formulated to enable optimizing the parameters $(\vec{\theta}, b)$, where $b \in [-1,+1]$ is an added bias parameter. Once the classifier is trained on the training data set using a classical optimizer, the trained circuit can now be used to assign labels to unlabelled data. Several optimizers have been proposed and used, both gradient based, such as ADAM and SPSA [46, 47], and gradient-free ones, such as COBYLA [48].

VQCs have some limitations, and solving these drawbacks is an active area of research. Barren plateaus occur in optimization algorithms of quantum ML when the parameter search space turns flat once the optimizer is run [49, 50]. Architecture design problems, such as choosing the correct cost functions and initializing the parameters, is a very complex process that has not been completely understood yet [51]. Additionally, a given variational quantum circuit with fixed form may not be able to capture all of the necessary states in the Hilbert space in its parameterization, and as a result, work on adaptive variational quantum algorithms, such as the Evolutionary Variational Quantum Eigensolver (EVQE), may be applicable to VQC [52].

There is a connection between the QSVM and VQC formulations [40, 53, 54] similar to the connection between classical Neural Networks and SVMs [55]. There are various discussions on how data encoding affects VQCs [54, 56], such as repeatedly encoding the inputs [57]. In addition, efficient methods were presented for encoding categorical features [58]. Lastly, there has been research into the expressiveness of PQCs [56, 59].

Support vector machines have been used to predict stock prices for over two decades, but also to predict financial distress and company's credit rating [60].

Next we look at a few example financial applications where the aforementioned quantum classifications techniques have been applied.

A. Prediction of Binary Options

SVM can be used to predict the outcome of exotic options. The *double no-touch* is a binary option [61] with a constant payout and is earned if and only if the underlying asset price remains between a predefined lower and upper bound until expiration. Unlike other options, such as a vanilla call, the payoff is not continuous, but all-or-nothing. Therefore, one can use SVM to separate the two classes corresponding to the binary option outcome. As these classes are not linearly separable, one needs a kernel to predict the outcome. This type of exotic option is often used in foreign exchange. The features selected to train the model could be the average directional index and the ratio between realized volatility over implied volatility.

B. Financial Forecasting

Financial forecasting is a planning tool that helps businesses to adapt to uncertainty based on predictions. Particularly, an algorithm to forecast annual earnings is of interest to any company. Such an algorithm has been proposed [62] that leverages k-NN. It matches a company's recent trend in annual earnings to historical earning sequences of other firms that are similar—known as *neighbor firms*. Some of the features taken into account to find such neighbors include matches based on industry, size and past accruals.

C. Credit Scoring

Credit scoring is a method to evaluate the credit risk of loan applications. It helps credit analysts to decide whether the applicants are worthy of credit. Based on past experience, credit scoring is the prediction of future behavior. An algorithm for this has been proposed using weighted k-NN [63]. The credit applicants are classified into one of two groups: a group whose members are likely to repay their debts and another group that should be denied credit because of high likelihood of defaulting.

IV. Clustering

Clustering consists of identifying groups of data points that are close to each other according to certain metrics. The feature space in which the data is encoded and the grouping metric are proxies for the actual similarities and differences of the data points. Inspired by quantum mechanics and suitable for high-dimensional data, *Quantum Clustering* (QC) [64] is an algorithm that belongs to the family of density-based clustering algorithms, where clusters are defined by regions of higher density of data points. The basic idea of QC is to map each data point to a Gaussian distribution centered at that sample.

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

An analytical form computed from the Schödinger equation is used to determine the potential that gives rise to a mixture of these Gaussian as its ground state. The minima in the system's potential energy function are used to identify clusters and are found via gradient-descent methods. Other points are assigned to clusters in a similar way. *Dynamic Quantum Clustering* (DQC) [65], an improvement of QC, adopts the time-dependent Schrödinger Equation in order to study evolution of quantum states associated with data points and the structure of the potential energy function. Being data-agnostic, DQC can be applied in a wide range of fields, especially finance, for example on S&P 500 data [66].

Classical-algorithm-inspired quantum-clustering techniques have also been proposed. For example, *k-means* is a well-known classical clustering algorithm that identifies, among all data points, the k most significant clusters and their representative centroids. Inspired by k-means, and providing the same robustness guarantees against some level δ of noise as the classical δ-k-means algorithm, the quantum *q-means* algorithm [67] has time complexity that is poly-logarithmic in the size of the dataset, and can be implemented using distance estimation and quantum matrix multiplication. A quantum spectral clustering algorithm for data represented as a graph has also been proposed [43]. To overcome the potentially huge time/space overhead of loading large datasets onto a quantum device, *coresets* have been proposed [68], which are small datasets combined with weight functions to sufficiently summarize original datasets. If small enough and still a faithful representation of the original dataset, a coreset could be used to enable execution on a NISQ computer [68–71].

Next we briefly discuss several use cases of these quantum clustering algorithms in the financial sector.

A. Fraud Detection

Clustering techniques can be used to perform *anomaly detection* by learning, from existing data, the *normal* mode(s), and then using this information to identify if a new data point is normal or otherwise *anomalous* [72]. Clustering can improve learning from imbalanced datasets, which oftentimes is the case for fraud data [73]. Clustering can also be combined with additional feature-selection and extraction techniques. For example, in time series data, a series could be hiking abnormally fast but still stay in a normal value range. Adding derivatives into the clustering algorithm can help detecting such an anomaly [74].

B. Stock Selection

Cluster analysis has also been used by investors for maximizing profit and minimizing loss. Stock returns are likely to be similar in a region thanks to geographic and macroeconomic features. Identification of stock clusters allows one to track those with similar returns but different risks. Once stocks are grouped by cluster analysis, informed investors can use the output for guidance. They will, for instance, look for same-return stocks and then choose to minimize risks. Alternatively, they will pick a cluster of same-risk stocks and high return [75].

C. Exchange Rate Regimes

In 1999, Levy-Yeyati and Sturzenegger [76] wanted to exhibit the inconsistency between the self-reported *de jure* classification from the International Monetary Fund (IMF) and the actual behavior shown in the data. In order to overcome bias, the authors proposed to use k-means to perform cluster analysis for exchange rate regimes. This led to a *de facto* classification, that has then been widely used as well as tested against prior methodologies [77].

D. Hedge Fund Clustering

Due to the variety of hedge fund—and, therefore, investing strategies—it can be hard for investors to classify such investment vehicles. Moreover, hedge funds tend to reveal less information than other type of funds as they do not fall under the same disclosure requirements. To classify hedge funds, predefined classes would not be able to manage correctly future type of hedge funds. Hence, clustering methods, such as k-means, have been used to overcome this issue [78]. The features considered are based on available characteristics of hedge funds, such as asset classes, size, fees, leverage and liquidity.

V. Generative Modeling

A *Generative Model* learns a probability distribution over data [79]. In supervised learning, where the model is provided as a set of input/label pairs $\{(x_i, y_i)\}$, the model learns $P(X, Y)$, the joint probability distribution of inputs and labels [80]. In unsupervised learning, these models can be used to generate new data given only samples [81]. Since measuring a quantum state naturally results in a probability distribution over the outcomes, it makes sense to see if quantum computation can be utilized for generative modeling.

a) Boltzmann Machine: The *Boltzmann Machine* [82] is defined by a collection of *visible* (observed) and *hidden* (marginalized out) random variables, and an undirected graph of conditional dependencies among them. It originates from thermodynamics where the nodes represent a system of correlated classical spins, s_i, under an external magnetic field. The classical *Ising Hamiltonian*

$$\mathcal{H} = -\sum_{i,j} J_{ij} s_i s_j - \sum_i h_i s_i$$

represents the energy of the system. Probabilistic inference is performed by sampling from the steady-state distribution—a Gibbs state—over the visible nodes. This is usually done utilizing Markov Chain Monte Carlo (MCMC) methods [82]. In most cases, the graph is restricted to being bipartite to make sampling feasible, resulting in the *Restricted Boltzmann Machine* (RBM) [83].

To formulate the *quantum* Boltzmann Machine, we quantize the Ising Hamiltonian by making the replacements $s_i \mapsto \sigma_i^z$, where σ_i^z is the Pauli Z spin operator for the i-th qubit. This results in a quantum Hamiltonian, and thus nodes are associated with qubits, and sampling is performed by projective measurements on the visible qubits.

One potential quantum method to sample from the visible nodes of the Gibbs state is to utilize *Quantum Annealing* (QA) [83–85]. For example, QA can be performed using the D-Wave devices [86].

Alternatively, we can prepare the quantum Gibbs state for this system by performing *Imaginary Time Evolution* (ITE) [87]. If the initial state is maximally mixed, performing ITE according to a quantum Hamiltonian will result in the associated Gibbs State. ITE can be performed variationaly, via McLachlan's principle, on a gate-based quantum computer [88]. Interestingly, the model introduced by Zoufal *et al.* [87] can be utilized to formulate a Boltzmann Machine without restricted connections that is tractable on a quantum device.

b) Generative Adversarial Learning: As another prominent architecture for modeling probability distributions, *Generative Adversarial Networks* (GANs) [89], operate by simultaneously training a *generator network* $\mathcal{G}_\theta$ and a *discriminator network* $\mathcal{D}_\phi$ against each other through adversarial games, for which $\mathcal{G}_\theta$ tries to fool $\mathcal{D}_\phi$ by generating fake data samples that are non-distinguishable from the ones drawn from the real distribution, whereas $\mathcal{D}_\phi$ tries to

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

tell them apart and not be fooled by $\mathcal{G}_\theta$. Quantum GANs (qGANs) have since been proposed [90, 91] and experimentally tested, for example, on superconducting quantum computers [92]. Either of qGAN's generator or discriminator, or both, can be in the form of quantum circuits. In addition to the original GAN's cross entropy, other distance metrics, such as Wasserstein [93], have also been proposed to improve the adversarial training on NISQ devices.

c) Quantum Born Machine: Closely related to quantum Boltzmann Machines and qGANs, *Quantum Born Machines* [94, 95] are another class of methods based on PQCs that have been studied for performing distributed-learning tasks. For example, Coyle *et al.* [95] propose using maximum mean discrepancy, the Stein discrepancy, and the Sinkhorn divergence, to improve the training of a subclass of quantum circuit Born machines.

Having discussed several quantum generative modeling techniques, we next look at sample use cases in the finance domain where these techniques can be applied.

A. Fraud Detection

Quantum versions of the Boltzmann Machine have be utilized for generative-learning and discriminative-learning tasks [83, 85]. Specifically for fraud detection, a Variational ITE Boltzmann Machine methodology has been utilized to classify anomalous credit-card transactions [87]. The system Hamiltonian is represented by a sum of Pauli strings whose coefficients are functions of trained parameters and input features. As mentioned earlier, this formulation is not restricted to the Ising Hamiltonian typically utilized by Boltzmann Machines. Predictions are performed by sampling from a single visible qubit indicating whether the transaction was fraudulent.

qGANs were combined with a framework for Generative Adversarial Anomaly detection, AnoGAN [96]. The generator was a PQC; the continuous output from the expectation of Pauli Z operators on each qubit was fed into a classical affine upscaling layer to achieve the full input feature dimension. The goal of the generator was to model the distribution of non-fraudulent transactions.

B. Probability Distribution Preparation

One crucial step for achieving quantum advantage in many financial applications is the efficient preparation of input probability distributions. qGANs [97, 98] and quantum Born Machines [94, 95] have both been utilized to learn PQCs for loading probability distributions. Upon convergence, the quantum circuit, as an efficient representation of the underlying distribution, can for example be used in amplitude estimation to perform derivative pricing tasks [99], with a theoretical quadratic speedup compared to classical Monte Carlo simulations. Additional techniques have been explored for the general creation of continuous distributions [100, 101]. Additional techniques exploring the creation of certain families of continuous distributions include the work of Rattew *et al.* [102] for the preparation of normal distributions.

VI. Quantum-Assisted Feature Extraction

Feature extraction refers to the set of techniques used to identify attributes of a dataset potentially helpful in ML tasks such as classification and regression. A quantum algorithm may help in feature extraction by computing properties of the dataset that a classical computer would fail to identify, or would take a very long time to do so. By encoding a data onto a quantum state, we can map a low-dimensional classical data to a much higher dimension in the Hilbert space. The expanded dimensionality of the quantum representation may be used to identify features invisible to a classical algorithm [103]. The growing interest in quantum kernels [104, 105], used in conjunction with Support Vector Machines, has also culminated an experimental demonstration [106].

A widely used algorithm to extract low-dimensional features out of a high-dimensional data is the Principal Component Analysis (PCA). In PCA, a large feature space is analyzed to identify attributes with the highest variance. Classical PCA takes time that is polynomial in the dimension or number of features in the original dataset. If such a classical data is mapped to a quantum density matrix, the quantum version of the algorithm can perform PCA exponentially faster, that is in time polynomial in the logarithm of the dimension [107].

Extracting features is particularly challenging while analyzing images where a large number of pixels have to be analyzed to identify image attributes. For these applications, a quantum computer may help in edge detection in images [108].

In finance, feature extraction may be used in detecting anomalies in transactions. As an example use case, graph theoretic tools are used to study bidding markets to identify colluding communities or cartels [109]. Quantum-aided graph kernel methods [103, 110] have been proposed to detect non-trivial features, such as *communities* [111], in a graph, which may represent, for instance, a network of financial parties that frequently transact with each other. When working with graph representations of data, we often want to measure the similarity between two graphs. In fact, Gaussian Boson Sampling can be used to check if two graphs are isomorphic to each other [112]. Moreover, Gaussian Boson Sampling can be used to construct kernel vectors representing the similarity between any two graphs [56].

Feature selection consists of choosing from a subset of the available features to pass to the model [34]. This contrasts with methods, such as PCA, that perform a transformation on the features. Feature selection can be formulated as a combinatorial minimization problem with binary decision variables designating whether to select a feature or not. Such binary optimization problems can be solved utilizing QA [84].

Below we present examples of how these techniques can be applied to financial use cases.

A. Model Reduction

PCA is a widely used method for dimensionality reduction that can be seen from the perspective of singular value decomposition. With the matrix decomposition $A = U\Sigma V$, where Σ is a rectangular diagonal matrix, the k-principal components are the first k columns of $U\Sigma$.

In 2014, Lloyd *et al.* [107] described a quantum PCA with exponential speedup over its classical counterpart. This theoretical speedup is realizable under certain conditions as it is based on HHL [3]. The algorithm can be used in finance to ease *model tuning*: as market conditions evolves, models needs to be tuned in order to match the implied volatility—volatility estimated by the model—with the market volatility. By using PCA, one reduces the number of components and, consequently, the number of parameters, thereby easing the model tuning.

For example, in a product based on foreign exchange, the input parameters are various and can range from global market data, such as risk-free interest rate, to asset specifics parameters, such as the spot price. As a consequence, the model tuning becomes computationally expensive due to the high number of inputs. However, as just the top three principal components can oftentimes explain over 95% of the

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

output variations, one can tune the model faster and still accurately by using only these three components.

A variation of quantum PCA has been implemented on hardware [113] to solve a similar problem by reducing the volatility factor dimension of the Heath-Jarrow-Morton model [114] in order to estimate forward rates.

B. Combinatorial Feature Selection for Credit Score Classification

As mentioned earlier, feature selection can be cast to a combinatorial optimization problem. In the case of supervised learning, it important to select features that are independent and relevant to the learning task. More specifically, for classification, the correlation coefficients between label and features can represent the relevance. The correlation matrix of the features can be used to represent the dependence between features. This can be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem, where the quadratic terms are the entries of the correlation coefficients between features, and the linear terms are correlations between the features and the label. QA [84] can be used to solve the QUBO utilizing heuristics provided by quantum mechanics. This exact formulation, solved with a Quantum Annealer, was applied to reduce the number of features used for assessing the credit worthiness of applicants [115].

VII. Reinforcement Learning

Reinforcement learning (RL) [116] is a ML technique where an agent attempts to learn through interactions with the environment. Classical RL has demonstrated remarkable capabilities in areas such as video games [117], board games [118, 119], robotics [120] and self-driving vehicles [121].

Classical RL is often formulated as a Markov Decision Process (MDP). MDPs enable the modeling of environments where actions are non-deterministic—that is, where taking a given action may probabilistically lead to one of multiple possible outcomes. As such, MDPs are useful for modeling many real-world problems where RL agents are exposed to inherent uncertainty. An MDP is characterized by a set of states $s \in S$, a set of actions $a \in A(s)$ available at each state s, transition dynamics specifying the probability of obtaining state s_j upon taking action a at state s_i, and a reward function $R(s_i, a, s_j)$. Of importance, an agent selects actions according to a *policy* which is maintained as a probability distribution over the actions available at any given state. The objective of an RL agent is to learn an optimal policy (one which selects actions maximizing the expected cumulative rewards) given that both the transition dynamics and the reward function of the environment are unknown *a priori*.

Utilizing quantum computers to perform RL was first discussed by Dong *et al.* in 2005 [122], with a follow-up in 2008 [123]. In their approach, the possible actions at any given state in the environment are maintained in a quantum superposition, and amplitude amplification is used to increase the probability of measuring a *good* action at any given state. In 2017, Dunjko *et al.* published a framework for quantum RL, where they expand upon the amplitude-amplification approach, which assumes access to an oracle representing the environment [124]. Furthermore, they introduce more general techniques for learning model meta-parameters, and additionally observe that there is significant potential for quantum advantage in luck-favoring task environments (i.e., environments where a lucky agent finds good sequences of actions much sooner than an unlucky agent) following from quantum search-based speedups. In a 2021 paper, Wang *et al.* derive a quantum RL algorithm with quadratic performance improvements in various parameters over corresponding classical algorithms for the evaluation of an optimal policy, state-values, and state-action pair values (q-values) in an MDP [125]. They explain that this work is applicable to any RL problem where the environment may be classically simulated, as a classical circuit implementing the simulator may be efficiently turned into a quantum circuit. Additionally, recent studies have explored the use of variational PQCs to implement both RL and Deep RL (DRL) in continuous action spaces [126, 127].

Next, we present some use cases showing how quantum RL techniques can be utilized in the finance domain.

A. Algorithmic Trading

The process of executing trades of financial instruments systematically by accounting for market variables with limited or no human intervention is referred to as *algorithmic* or *automated trading*. Generally, algorithmic trading is performed by predictions in a supervised manner followed by obtaining optimal trading decisions under uncertainty associated with the corresponding predictions and market volatility. RL bypasses the need for predictions by casting algorithmic trading as a sequential decision-making problem wherein trading decisions are obtained directly that maximize the cumulative returns over a finite time horizon [128]. The domain of RL, and more specifically DRL, has demonstrated huge applicability for algorithmic trading [129]. However, such RL approaches for automated trading operate under certain strong assumptions and may benefit from quantum ML techniques for improved time and model complexity.

Algorithmic trading can be cast to a multi-period portfolio-selection problem that involves re-balancing the portions of capital invested in selected assets at each stage. There have been attempts to solve this multi-stage optimization problem with a QA device to obtain an optimal trading trajectory [130]. However, this approach does not adopt any RL technique based on policy or value function approximation. Due to the hardware limitations of the current quantum devices, quantum RL approaches have not been directly applied yet to automated trading. Nevertheless, components of algorithmic trading can certainly benefit from quantum advantages offered by quantum RL. For instance, the LSTM neural network architecture used as q-value estimator [131] could be potentially replaced with quantum LSTM [15] for improved performance. Also, variational quantum circuits [127] can be used for different DRL components applied to decision-making in algorithmic trading.

B. Market Making

Market makers have an important role in financial markets as they increase the liquidity of exchanges, thereby facilitating transactions and investment [132, 133]. A market maker is responsible for maintaining a set of sell orders (*asks*) and buy orders (*bids*) at various quantities and prices. When incoming market orders are made on a security held by the market maker, they are required to transact. As such, they inherently assume risk, as a position they are forced to acquire can subsequently depreciate. Market makers profit by taking advantage of the gap, called *spread*, between the lowest ask and highest bid. For instance, assuming an incoming market order is made to sell security X, the market maker will fulfill the order purchasing it at their bid price. If another market order is immediately made to purchase security X, the market maker fulfills the order by selling it at their ask thereby price, obtaining a profit equal to the spread.

Market making is amenable to quantum RL, where the problem can be modelled with an agent state, taking into account attributes such as inventory and risk-tolerance, and an environment state where the agent only has partial information, which may not necessarily be Markovian [134].

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

VIII. Natural Language Processing

Natural Language Processing (NLP) is the field concerned with automated text and language analysis. A drawback with most search engines that use classical NLP is that they understand separate words and not a grammatical structure. This has triggered research in distributional compositional semantics (DisCo). A particular DisCo model is the Coecke, Sadrzadeh and Clark (CSC) model [135, 136], based on tensor-product composition inspired by quantum theory.

In modern classical NLP, the vector space model [137] is used to compute the meaning of individual words. Given an individual word w in a text, its meaning is computed by first setting up basis words (i.e., the most common words in the text) and then, for each of w's nearby basis words, counting its frequency through the text. The proximity of two words is measured by the similarity between them and it is calculated, for example, using the inner product of their normalized representative vectors. These are called *distributional methods* and cannot be extended to find the meaning of long sentences as two sentences are not typically repeated. In contrast, algorithms based on compositional semantics derive the meaning of a sentence from known meanings of component words. The DisCo model combines both approaches to introduce grammatical understanding to the composition of word vectors.

In the CSC model, each grammatical type in the text is assigned a tensor product space based on some grammar (e.g., the Lambek's pregroup grammar [138]). For instance, a transitive verb takes a subject noun as a left argument and an object noun as right argument. The meaning of a noun is calculated as in the distributional model; its vector space is denoted as $\mathscr{N}$. Therefore, the meaning of a transitive verb is a tensor in the space $\mathscr{N} \otimes \mathscr{L} \otimes \mathscr{N}$, where $\mathscr{L}$ is the meaning space for the sentences. An important feature of this model is the use of diagrammatic notation for vectors, tensors and linear maps. This model has the computational challenge of large tensor product spaces. Even thought there exist classical approaches—such as dimensionality reduction [139]—to avoid the calculation of the full tensor product, they make certain assumptions that are not always necessarily met.

The recent development of encoding classical data on quantum hardware using variational PQCs enables quantum NLP to be particularly suitable for NISQ devices. In particular, the quantum CSC model can encode linguistic structures faster in comparison to its classical counterpart. Its quantum speedup stems from the quantum nearest-neighbor algorithm that is employed for the sentence similarity calculations in the DisCo framework. If certain conditions are met, for the N-dimensional noun meaning space there is a quantum algorithm capable of classifying any CSC model sentence composed of n tensors into M classes with time $O(\sqrt{MN}\log(M))$, an improvement over classical methods' $O(NM)$ complexity [140].

Below are a few potential applications of the discussed quantum NLP techniques in the financial sector.

A. Risk Assessment

Banks can quantify the chances of a successful loan payment based on a *credit risk assessment*. Usually, the payment capacity is calculated based on previous spending patterns and past loan payment history. However, this information is not always available, especially for underbanked applicants. NLP techniques can be applied to solve this problem, by using multiple data points to assess credit risk. For instance, NLP can measure attitude and an entrepreneurial mindset in business loans. Similarly, it can also point out incoherent data and take it up for more scrutiny. Even more, the subtle aspects, such as the lender's and borrower's emotions during a loan process, can be incorporated with the help of NLP [141, 142].

B. Financial Forecasting

Financial forecasting is based on many macroeconomic factors, which are unstructured and scattered across different sources. This is the reason why NLP techniques are frequently employed [143]. For example, NLP has been proposed for classification of news articles as significant or non-significant from the financial point of view [144]. In addition, *sentiment analysis*, which plays an important role in decision-making by traders, has also been carried out with the help of NLP techniques [145].

C. Accounting and Auditing

Another application of NLP is *accounting and auditing* [142], whose objective is the detection and prevention of fraud via evaluation of accounting systems, monitoring of internal controls, assessment of fraud risk, and interpretation of financial data for anomalous trends. NLP has been proposed for the creation of semantic knowledge bases or trees for financial accounting standards. Also auditors can detect anomalies in financial statements by applying NLP techniques.

IX. Conclusion

In this paper, we presented an introduction of quantum ML techniques and their applications in the financial services sector. We identified seven machine learning tasks, for which several quantum algorithms have been previously proposed in the literature: regression, classification, clustering, generative learning, feature extraction, sequential decision-making, and Natural Language Processing. We analyzed the speedups offered by various quantum ML techniques, and discussed the financial applications that could benefit from such quantum acceleration. Moreover, where the literature for finance-specific quantum ML techniques remains sparse, we provide insights into applying state-of-the-art general quantum ML techniques to specific financial use cases. Additionally, we consider the realities of implementing quantum computing techniques in the financial sector, for example, by considering the challenges imposed by hardware limitations. In summary, this article serves as a road map towards enriching the finance industry with quantum ML techniques in the NISQ era and beyond.

Disclaimer

This paper was prepared for information purposes by the Future Lab for Applied Research and Engineering (FLARE) group of JPMorgan Chase Bank, N.A.. This paper is not a product of the Research Department of JPMorgan Chase & Co. or its affiliates. Neither JPMorgan Chase & Co. nor any of its affiliates make any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, but limited to, the completeness, accuracy, reliability of information contained herein and the potential legal, compliance, tax or accounting effects thereof. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction.

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. Menard *et al.*, "A game plan for quantum computing," *McKinsey & Co.*, 2020.
[2] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, 2018.
[3] A. W. Harrow *et al.*, "Quantum algorithm for linear systems of equations," *PRL*, 2009.
[4] A. Gilyén *et al.*, "Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics," *STOC*, 2019.
[5] S. Aaronson, "Read the fine print," *Nature Phys*, 2015.
[6] V. Giovannetti *et al.*, "Quantum random access memory," *PRL*, 2008.
[7] J. Cortese *et al.*, "Loading classical data into a quantum computer," *arXiv:1803.01958*, 2018.
[8] N. Wiebe *et al.*, "Quantum algorithm for data fitting," *PRL*, 2012.
[9] G. Wang, "Quantum algorithm for linear regression," *PRA*, 2017.
[10] M. Schuld *et al.*, "Prediction by linear regression on a quantum computer," *PRA*, 2016.
[11] S. Chakraborty *et al.*, "The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation," *arXiv:1804.01973*, 2018.
[12] I. Kerenidis *et al.*, "Quantum gradient descent for linear systems and least squares," *PRA*, 2020.
[13] N.-H. Chia *et al.*, "Quantum-inspired sublinear classical algorithms for solving low-rank linear systems," *arXiv:1811.04852*, 2018.
[14] J. Bausch, "Recurrent quantum neural networks," in *NeurIPS*, 2020.
[15] S. Y.-C. Chen *et al.*, "Quantum long short-term memory," *arXiv:2009.01783*, 2020.
[16] Y. Takaki *et al.*, "Learning temporal data with a variational quantum recurrent neural network," *PRA*, 2021.
[17] J. H. Cochrane, *Asset pricing: Revised edition*. Princeton university press, 2009.
[18] S. Gu *et al.*, "Empirical asset pricing via machine learning," *The Review of Financial Studies*, 2020.
[19] L. Chen *et al.*, "Deep learning in asset pricing," *arXiv:1904.00745*, 2021.
[20] J. Fan *et al.*, "Equity-bond correlation: A historical perspective," *GCM Research Note*, 2017.
[21] R. Tibshirani, "Regression shrinkage and selection via the lasso," *JRSSB*, 1996.
[22] Y. Du *et al.*, "Quantum differentially private sparse regression learning," *arXiv:2007.11921*, 2020.
[23] M. R. Fengler *et al.*, "The analysis of implied volatilities," in *Applied Quantitative Finance*, Springer, 2002.
[24] W. Ahmad *et al.*, "The us equity sectors, implied volatilities, and covid-19: What does the spillover analysis reveal?" *Resources Policy*, 2021.
[25] T. Sakuma, "Application of deep quantum neural networks to finance," *arXiv:2011.07319*, 2020.
[26] K. Beer *et al.*, "Training deep quantum neural networks," *Nat. Comm.*, 2020.
[27] G. P. Zhang, "Neural networks for classification: A survey," *TSMCPC*, 2000.
[28] S. Johri *et al.*, "Nearest centroid classification on a trapped ion quantum computer," *arXiv:2012.04145*, 2020.
[29] M. Minsky *et al.*, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.
[30] S. Lloyd, "Quantum machine learning for data classification," *Physics*, 2021.
[31] A. Kapoor *et al.*, "Quantum perceptron models," *NIPS*, 2016.
[32] K. L. Clarkson *et al.*, "Sublinear optimization for machine learning," *JACM*, 2012.
[33] T. Li *et al.*, "Sublinear quantum algorithms for training linear and kernel-based classifiers," *ICML*, 2019.
[34] J. Friedman *et al.*, *The elements of statistical learning*. Springer, 2001.
[35] J. Li *et al.*, "Quantum k-nearest neighbor classification algorithm based on hamming distance," *arXiv:2103.04253*, 2021.
[36] C. Durr *et al.*, "A quantum algorithm for finding the minimum," *arXiv:9607014*, 1996.
[37] A. Basheer *et al.*, "Quantum k-nearest neighbors algorithm," *arXiv:2003.09187*, 2020.
[38] K. Miyamoto *et al.*, "A quantum algorithm for finding k-minima," *arXiv:1907.03315*, 2019.
[39] B. E. Boser *et al.*, "A training algorithm for optimal margin classifiers," *COLT*, 1992.
[40] V. Havlíček *et al.*, "Supervised learning with quantum-enhanced feature spaces," *Nature*, 2019.
[41] K. Mitarai *et al.*, "Methodology for replacing indirect measurements with direct measurements," *Phys. Rev. Research*, 2019.
[42] H. Buhrman *et al.*, "Quantum fingerprinting," *PRL*, 2001.
[43] I. Kerenidis *et al.*, "Quantum algorithms for second-order cone programming and support vector machines," *Quantum*, 2021.
[44] P. Rebentrost *et al.*, "Quantum support vector machine for big data classification," *PRL*, 2014.
[45] S. Sim *et al.*, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *QUTE*, 2019.
[46] D. P. Kingma *et al.*, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.
[47] Q. Li *et al.*, "Simultaneous perturbation stochastic approximation algorithm for automated image registration optimization," *IGARSS*, 2006.
[48] M. Powell, "A view of algorithms for optimization without derivatives," *Mathematics TODAY*, 2007.
[49] M. Cerezo *et al.*, "Barren plateau issues for variational quantum-classical algorithms," *APS*, 2020.
[50] K. Sharma *et al.*, "Trainability of dissipative perceptron-based quantum neural networks," *arXiv:2005.12458*, 2020.
[51] M. Cerezo *et al.*, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nat. Comm.*, 2021.
[52] A. G. Rattew *et al.*, "A domain-agnostic, noise-resistant, hardware-efficient evolutionary variational quantum eigensolver," *arXiv:1910.09694*, 2019.
[53] M. Schuld, "Supervised quantum machine learning models are kernel methods," *arXiv:2101.11020*, 2021.
[54] H.-Y. Huang *et al.*, "Power of data in quantum machine learning," *Nat. Comm.*, 2021.
[55] A. Jacot *et al.*, "Neural tangent kernel: Convergence and generalization in neural networks," *arXiv:1806.07572*, 2020.
[56] M. Schuld *et al.*, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *PRA*, 2021.
[57] A. Pérez-Salinas *et al.*, "Data re-uploading for a universal quantum classifier," *Quantum*, 2020.
[58] H. Yano *et al.*, "Efficient discrete feature encoding for variational quantum classifier," *QCE*, 2020.
[59] A. Abbas *et al.*, "The power of quantum neural networks," *Nat. Comp. Sci.*, 2021.
[60] B. Cao *et al.*, "Application of svm in financial research," in *CSO*, 2009.
[61] A. Nekritin, *Binary Options: Strategies for Directional and Volatility Trading*. John Wiley & Sons, 2012.
[62] P. D. Easton *et al.*, "Forecasting earnings using k-nearest neighbor matching," *SSRN*, 2020.
[63] M. Mukid *et al.*, "Credit scoring analysis using weighted k nearest neighbor," in *Journal of Physics: Conference Series*, 2018.
[64] D. Horn *et al.*, "Algorithm for data clustering in pattern recognition problems based on quantum mechanics," *PRL*, 2001.
[65] M. Weinstein *et al.*, "Dynamic quantum clustering: A method for visual exploration of structures in data," *Phys. Rev. E*, 2009.
[66] M. Weinstein *et al.*, "Analyzing big data with dynamic quantum clustering," *arXiv:1310.2700*, 2013.
[67] I. Kerenidis *et al.*, "Q-means: A quantum algorithm for unsupervised machine learning," *arXiv:1812.03584*, 2018.
[68] T. Tomesh *et al.*, "Coreset clustering on small quantum computers," *arXiv:2004.14970*, 2020.
[69] S. U. Khan *et al.*, "K-means clustering on noisy intermediate scale quantum computers," *arXiv:1909.12183*, 2019.
[70] S. S. Mendelson *et al.*, "Quantum-assisted clustering algorithms for nisq-era devices," *arXiv:1904.08992*, 2019.
[71] E. Aïmeur *et al.*, "Quantum clustering algorithms," in *ICML*, 2007.
[72] C. C. Aggarwal *et al.*, "An effective and efficient algorithm for high-dimensional outlier detection," *VLDB*, 2005.
[73] N. D. Singh *et al.*, "Clustering and learning from imbalanced data," *arXiv:1811.00972*, 2018.
[74] M. Sathyapriya *et al.*, "A cluster based approach for credit card fraud detection system using hmm with the implementation of big data technology," *IJAER*, 2019.
[75] N. Da Costa Jr *et al.*, "Stock selection based on cluster analysis," *Economics Bulletin*, 2005.
[76] E. Levy-Yeyati *et al.*, "Classifying exchange rate regimes: Deeds vs. words," *European Economic Review*, 2005.
[77] B. Eichengreen *et al.*, "How reliable are de facto exchange rate regime classifications?" *IJFE*, 2013.
[78] N. Das, "Hedge fund classification using k-means clustering method," *CEF*, 2003.
[79] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016.
[80] A. Y. Ng *et al.*, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *NIPS*, 2002.
[81] A. Radford *et al.*, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv:1511.06434*, 2016.
[82] D. Koller *et al.*, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[83] M. H. Amin *et al.*, "Quantum boltzmann machine," *PRX*, 2018.
[84] E. Farhi *et al.*, "Quantum computation by adiabatic evolution," *arXiv:0001106*, 2000.
[85] V. Dixit *et al.*, "Training restricted boltzmann machines with a d-wave quantum annealer," *Front. Phys.*, 2021.
[86] R. Harris *et al.*, "Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor," *Phys. Rev. B*, 2010.
[87] C. Zoufal *et al.*, "Variational quantum boltzmann machines," *QMI*, 2021.
[88] X. Yuan *et al.*, "Theory of variational quantum simulation," *Quantum*, 2019.
[89] I. J. Goodfellow *et al.*, "Generative adversarial networks," *arXiv:1406.2661*, 2014.
[90] S. Lloyd *et al.*, "Quantum generative adversarial learning," *PRL*, 2018.
[91] P.-L. Dallaire-Demers *et al.*, "Quantum generative adversarial networks," *PRA*, 2018.
[92] L. Hu *et al.*, "Quantum generative adversarial learning in a superconducting quantum circuit," *Science Advances*, 2019.
[93] S. Chakrabarti *et al.*, "Quantum wasserstein gans," *arXiv:1911.00111*, 2019.
[94] S. Cheng *et al.*, "Information perspective to probabilistic modeling: Boltzmann machines versus born machines," *Entropy*, 2018.
[95] B. Coyle *et al.*, "The born supremacy: Quantum advantage and training of an ising born machine," *npj Quantum Information*, 2020.
[96] D. Herr *et al.*, "Anomaly detection with variational quantum generative adversarial networks," *Quantum Science and Technology*, 2021.
[97] C. Zoufal *et al.*, "Quantum generative adversarial networks for learning and loading random distributions," *npj Quantum Information*, 2019.
[98] H. Situ *et al.*, "Quantum generative adversarial network for generating discrete distribution," *Information Sciences*, 2020.
[99] N. Stamatopoulos *et al.*, "Option pricing using quantum computers," *Quantum*, 2020.
[100] T. Häner *et al.*, "Optimizing quantum circuits for arithmetic," *arXiv:1805.12445*, 2018.
[101] L. Grover *et al.*, "Creating superpositions that correspond to efficiently integrable probability distributions," *arXiv:0208712*, 2002.
[102] A. G. Rattew *et al.*, "The efficient preparation of normal distributions in quantum registers," *arXiv:2009.06601*, 2020.
[103] M. Schuld *et al.*, "A quantum hardware-induced graph kernel based on gaussian boson sampling," *arXiv:1905.12646*, 2019.
[104] R. Chatterjee *et al.*, "Generalized coherent states, reproducing kernels, and quantum support vector machines," *arXiv:1612.03713*, 2016.
[105] X. Wang *et al.*, "Towards understanding the power of quantum kernels in the nisq era," *arXiv:2103.16774*, 2021.
[106] K. Bartkiewicz *et al.*, "Experimental kernel-based quantum machine learning in finite feature space," *Scientific Reports*, 2020.
[107] S. Lloyd *et al.*, "Quantum principal component analysis," *Nature Physics*, 2014.
[108] R.-G. Zhou *et al.*, "Quantum image edge extraction based on improved prewitt operator," *QIP*, 2019.
[109] J. Wachs *et al.*, "A network approach to cartel detection in public auction markets," *Scientific Reports*, 2019.
[110] L. Bai *et al.*, "Quantum kernels for unattributed graphs using discrete-time quantum walks," *Pattern Recognition Letters*, 2017.
[111] R. Shaydulin *et al.*, "Network community detection on small quantum computers," *QUTE*, 2019.
[112] K. Brádler *et al.*, "Graph isomorphism and gaussian boson sampling," *Special Matrices*, 2021.
[113] A. Martin *et al.*, "Toward pricing financial derivatives with an ibm quantum computer," *Phys. Rev. Research*, 2021.
[114] D. Heath *et al.*, "Bond pricing and the term structure of interest rates: A new methodology for contingent claims valuation," *Econometrica*, 1992.
[115] A. Milne *et al.*, "Optimal feature selection in credit scoring and classification using a quantum annealer," *White Paper 1Qbit*, 2017.
[116] R. S. Sutton *et al.*, *Reinforcement learning: An introduction*. MIT press, 2018.
[117] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
[118] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.
[119] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, 2017.
[120] J. Kober *et al.*, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, 2013.
[121] A. E. Sallab *et al.*, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, 2017.
[122] D. Dong *et al.*, "Quantum reinforcement learning," in *ICNC*, 2005.
[123] D. Dong *et al.*, "Quantum reinforcement learning," *IEEE SMC*, 2008.
[124] V. Dunjko *et al.*, "Advances in quantum reinforcement learning," in *IEEE SMC*, 2017.
[125] D. Wang *et al.*, "Quantum algorithms for reinforcement learning with a generative model," in *ICML*, 2021.
[126] S. Y.-C. Chen *et al.*, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, 2020.
[127] S. Wu *et al.*, "Quantum reinforcement learning in continuous action space," *arXiv:2012.10711*, 2020.
[128] Z. Zhang *et al.*, "Deep reinforcement learning for trading," *JFDS*, 2020.
[129] T.-V. Pricope, "Deep reinforcement learning in quantitative algorithmic trading: A review," *arXiv:2106.00123*, 2021.
[130] G. Rosenberg *et al.*, "Solving the optimal trading trajectory problem using a quantum annealer," *IEEE JSTSP*, 2016.
[131] Y. Li *et al.*, "Deep robust reinforcement learning for practical algorithmic trading," *IEEE Access*, 2019.
[132] M. Avellaneda *et al.*, "High-frequency trading in a limit order book," *Quantitative Finance*, 2008.
[133] O. Guéant *et al.*, "Dealing with the inventory risk: A solution to the market making problem," *Mathematics and Financial Economics*, 2013.
[134] T. Spooner *et al.*, "Market making via reinforcement learning," *arXiv:1804.04216*, 2018.
[135] S. Clark *et al.*, "A compositional distributional model of meaning," in *QI*, 2008.
[136] B. Coecke *et al.*, "Mathematical foundations for a compositional distributional model of meaning," *arXiv:1003.4394*, 2010.
[137] H. Schütze, "Automatic word sense discrimination," *Computational Linguistics*, 1998.
[138] J. Lambek, *From Word to Sentence: a computational algebraic approach to grammar*. Polimetrica sas, 2008.
[139] T. Polajnar *et al.*, "Learning type-driven tensor-based meaning representations," *arXiv:1312.5985*, 2013.
[140] W. Zeng *et al.*, "Quantum algorithms for compositional natural language processing," *arXiv:1608.01406*, 2016.
[141] L. Purda *et al.*, "Accounting variables, deception, and a bag of words: Assessing the tools of fraud detection," *Contemporary Accounting Research*, 2015.
[142] I. E. Fisher *et al.*, "Natural language processing in accounting, auditing and finance: A synthesis of the literature with a roadmap for future research," *ISAFM*, 2016.
[143] F. Z. Xing *et al.*, "Natural language based financial forecasting: A survey," *AI Review*, 2018.
[144] S. Yıldırım *et al.*, "Classification of "hot news" for financial forecast using nlp techniques," in *Big Data*, 2018.
[145] K. Mishev *et al.*, "Evaluation of sentiment analysis in finance: From lexicons to transformers," *IEEE Access*, 2020.

Authorized licensed use limited to: University of Illinois. Downloaded on January 24,2022 at 19:26:59 UTC from IEEE Xplore. Restrictions apply.