scientific reports

Check for updates

OPEN

QUBO formulations for training machine learning models

Prasanna Date[1]✉, Davis Arthur[2] & Lauren Pusey-Nazzaro[3]

Training machine learning models on classical computers is usually a time and compute intensive process. With Moore's law nearing its inevitable end and an ever-increasing demand for large-scale data analysis using machine learning, we must leverage non-conventional computing paradigms like quantum computing to train machine learning models efficiently. Adiabatic quantum computers can approximately solve NP-hard problems, such as the quadratic unconstrained binary optimization (QUBO), faster than classical computers. Since many machine learning problems are also NP-hard, we believe adiabatic quantum computers might be instrumental in training machine learning models efficiently in the post Moore's law era. In order to solve problems on adiabatic quantum computers, they must be formulated as QUBO problems, which is very challenging. In this paper, we formulate the training problems of three machine learning models—linear regression, support vector machine (SVM) and balanced k-means clustering—as QUBO problems, making them conducive to be trained on adiabatic quantum computers. We also analyze the computational complexities of our formulations and compare them to corresponding state-of-the-art classical approaches. We show that the time and space complexities of our formulations are better (in case of SVM and balanced k-means clustering) or equivalent (in case of linear regression) to their classical counterparts.

The importance of machine learning algorithms in scientific advancement cannot be understated. Machine learning algorithms have given us great predictive power in medical science[1], economics[2], agriculture[3] etc. These algorithms can only be implemented and deployed after they have been trained—a process that requires tuning the model parameters of a machine learning model in order to extract meaningful information from data. Training a machine learning model is a time and compute intensive process usually. In such situations, one is often forced to make a trade-off between the accuracy of a trained model and the training time. With the looming end of Moore's law and rapidly increasing demand for large-scale data analysis using machine learning, there is a dire need to explore the applicability of non-conventional computing paradigms like quantum computing to accelerate the training of machine learning models.

Quantum computers are known to bypass classically-difficult computations by performing operations on high-dimensional tensor product spaces[4]. To this extent, we believe that machine learning problems, which often require such manipulation of high-dimensional data sets, can be posed in a manner conducive to efficient quantum computation. Quantum computers have been shown to yield approximate solutions to NP-hard problems, such as the quadratic unconstrained binary optimization (QUBO) problem[5], graph clustering problem[6], protein folding problem[7] etc. In addition to these results, demonstration of quantum supremacy by Google[8] has led us to believe that quantum computers might offer speedup in a much wider range of problems such as accelerating training of machine learning models.

To this extent, the principal contributions of our work are:

1. We formulate the training problems of three machine learning models—linear regression, support vector machine (SVM) and balanced *k*-means clustering—as QUBO problems so that they can be trained on adiabatic quantum computers.
2. For the aforementioned models, we provide a theoretical comparison between state-of-the-art classical training algorithms and our formulations that are conducive to being trained on adiabatic quantum computers. We observe that the time and space complexities of our formulations are better in case of SVM and balanced *k*-means clustering, and equivalent in case of linear regression, to their classical counterparts.

[1]Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA. [2]Auburn University, Auburn, AL 36849, USA. [3]Washington University in St. Louis, St. Louis, MO 63130, USA. ✉email: datepa@ornl.gov

Our formulations provide a promising outlook for training such machine learning models on adiabatic quantum computers. In the future, larger and more robust quantum computers are sought to abate the limitations of current machines and potentially allow machine learning models to be trained faster and more reliably.

Related work

Quantum machine learning algorithms have been proposed for both universal and adiabatic quantum computers. We briefly review a handful of such algorithms that leverage universal quantum computers here. Relevant algorithms leveraging adiabatic quantum computers have been reviewed in the subsequent sections. Quantum machine learning algorithms, and in general, all quantum algorithms will greatly benefit from optimal design of quantum circuits[9,10], optimized quantum states[11], quantum memory[12], improved quantum coherence times[13] and quantum error correction[14]. Today's quantum machine learning algorithms are catered towards quantum computers in the noisy intermediate-scale quantum (NISQ) era. Results presented in this paper are part of our ongoing work to accelerate training of machine learning models using quantum computers[15–17].

Farhi et al. proposed the Quantum Approximate Optimization Algorithm (QAOA), which produces approximate solutions for combinatorial optimization problems[18–20], is computationally universal[21], and has been used to train unsupervised machine learning models[22]. Farhi and Neven also proposed quantum neural networks where a sequence of parameter dependent unitary transformations act on classical or quantum input data and produce classification predictions on the output qubits[23]. Gyongyosi and Imre proposed training optimizations for such gate-based quantum neural network models[24]. Benedetti et al.[25] proposed the use of the variational quantum eigensolver (VQE) algorithm in conjunction with parameterized quantum circuits as quantum machine learning models. QAOA and VQE based quantum machine learning models are widely used in the literature.

Adiabatic quantum computers

The adiabatic theorem states that a quantum physical system remains in its instantaneous eigenstate under a slowly acting perturbation if there is a gap between its eigenvalue and the rest of the Hamiltonian's spectrum[26]. Adiabatic quantum computers leverage the adiabatic theorem to perform computation[27]. Specifically, starting with the global minimum of a simple Hamiltonian, they homotopically connect it to the global minimum of the problem of interest[28]. The D-Wave adiabatic quantum computers, for instance, are adept at approximately solving the quadratic unconstrained binary optimization (QUBO) problem, which is stated as follows:

$$\min_{z \in \mathbb{B}^M} \; z^T A z + z^T b \tag{1}$$

where, M is a natural number; $\mathbb{B} = \{0, 1\}$ is the set of binary numbers; $z \in \mathbb{B}^M$ is the binary decision vector; $A \in \mathbb{R}^{M \times M}$ is the real-valued $M \times M$ QUBO matrix; and, $b \in \mathbb{R}^M$ is the real-valued M-dimensional QUBO vector.

Notation

We use the following notation throughout this paper:

- $\mathbb{R}, \mathbb{N}, \mathbb{B}$: Set of real numbers, natural numbers and binary numbers ($\mathbb{B} = \{0, 1\}$).
- N: Number of data points (number of rows) in the training data set, $N \in \mathbb{N}$.
- d: Number of features (number of columns) in the training data set, $d \in \mathbb{N}$.
- X: Training data set, usually $X \in \mathbb{R}^{N \times d}$, i.e. X contains N data points along its rows, and each data point is a d-dimensional row vector.
- Y: Regression labels of the training data set in case of regression ($Y \in \mathbb{R}^N$); classification labels of the training data set in case of support vector machine ($Y \in \mathbb{B}^N$).

Linear regression

Background. Linear regression is one of the oldest statistical machine learning techniques that is used in a wide range of applications, such as scientific research[29], business[30] and weather forecasting[31]. Linear regression models the relationship between a dependent variable and one or more independent variables.

Adiabatic quantum computing approaches have been proposed in the literature for solving the linear regression problem (Eq. 2). Borle et al. propose a quantum annealing approach for the linear least squares problem[32]. Chang et al. present a quantum annealing approach for solving polynomial systems of equations using least squares[33]. Chang et al. propose a method for solving polynomial equations using quantum annealing and discuss its application to linear regression[34]. While these approaches can only find positive real-valued regression weights, our formulation finds both positive and negative real-valued regression weights.

Here, we denote $X \in \mathbb{R}^{N \times (d+1)}$ as the augmented regression training data matrix, where we have augmented each row of the original $X \in \mathbb{R}^{N \times d}$ with unity for the sake of mathematical convenience. The regression training labels are denoted by $Y \in \mathbb{R}^N$, and the regression weights are denoted by $w \in \mathbb{R}^{d+1}$. Given X and Y, training a linear regression model can be stated as follows:

$$\min_{w \in \mathbb{R}^{d+1}} \; E(w) = ||Xw - Y||^2 \tag{2}$$

Here, $E(w)$ is the Euclidean error function. With reference to Fig. 1, the blue dots represent the data points X and Y, and the green line, characterized by the weights w, is the regression hyperplane which fits the data. The regression problem has an analytical solution, given by

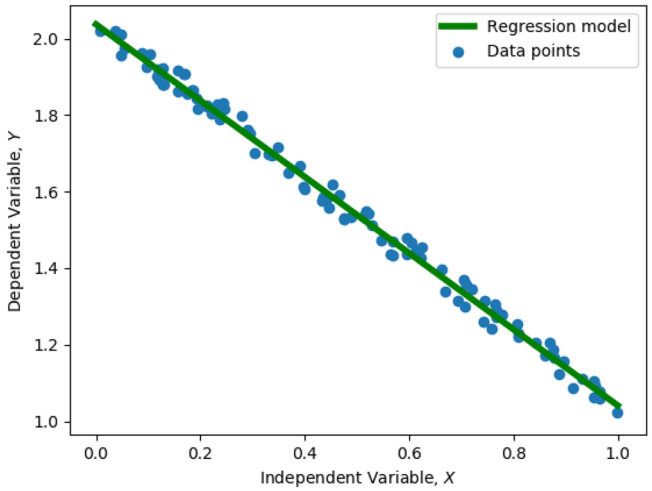

Figure 1. Fitting a linear regression model (green line) to data (blue dots).

$$w = (X^T X)^{-1} X^T Y \tag{3}$$

If $(X^T X)^{-1}$ does not exist, the pseudo inverse is computed. Time complexity of linear regression is known to be $\mathcal{O}(Nd^2)$.

QUBO formulation. We start by rewriting Problem (2) as:

$$\min_{w \in \mathbb{R}^{d+1}} E(w) = w^T X^T X w - 2 w^T X^T Y + Y^T Y \tag{4}$$

Next, we introduce a K-dimensional precision vector $P = [p_1, p_2, \ldots, p_K]^T$. Each entry in P can be an integral power of 2, and can be both positive or negative. We also introduce a K-dimensional vector $\hat{w}_i \in \mathbb{B}^K$ with binary coefficients, such that the inner product $\hat{w}_i^T P$ yields a scalar $w_i \in \mathbb{R}$. This scalar w_i represents the ith entry in our weight vector, where $1 \leq i \leq (d+1)$. The entries of P must be sorted, for instance $P = \left[-2, -1, -\frac{1}{2}, \frac{1}{2}, 1, 2, \right]^T$. $\hat{w}_{ik}$ can be thought of as a binary decision variable that selects or ignores entries in P depending on whether its value is 1 or 0 respectively. With this formulation, we can have up to 2^K unique values for each w_i when P contains only positive values for instance. However, if P contains negative values as well, then the number of unique attainable values for each w_i might be less than 2^K. For example, if $P = [-1, -\frac{1}{2}, \frac{1}{2}, 1]$, then only the following seven distinct values can be attained: $\{-\frac{3}{2}, -1, -\frac{1}{2}, 0, \frac{1}{2}, 1, \frac{3}{2}\}$.

Now, let us define the binary vector $\hat{w} \in \mathbb{B}^{K(d+1)}$, such that

$$\hat{w} = [\hat{w}_{11} \ \ldots \ \hat{w}_{1K} \ \hat{w}_{21} \ \ldots \ \hat{w}_{2K} \ \ldots \ \hat{w}_{(d+1)1} \ \ldots \ \hat{w}_{(d+1)K}]^T \tag{5}$$

Similarly, we can define a precision matrix ($\mathcal{P}$) as follows:

$$\mathcal{P} = I_{d+1} \otimes P^T \tag{6}$$

where I_{d+1} represents the $(d+1)$-dimensional identity matrix, and $\otimes$ represents the Kronecker product. Note that $\mathcal{P}$ has the dimensions $(d+1) \times K(d+1)$. We can now recover our original weight vector by observing that:

$$w = \mathcal{P} \hat{w} \tag{7}$$

We have thus represented our weight vector (to finite precision) in terms of the precision matrix $\mathcal{P}$ and the binary vector $\hat{w} \in \mathbb{B}^{K(d+1)}$. We are now able to pose the minimization problem of Eq. (4) as an equivalent QUBO problem. Let us substitute the expression we obtained for the weight vector w in terms of $\mathcal{P}$ and $\hat{w}$ into Eq. (4), which yields:

$$\min_{\hat{w} \in \mathbb{B}^{(d+1)K}} E(\hat{w}) = \hat{w}^T \mathcal{P}^T X^T X \mathcal{P} \hat{w} - 2 \hat{w}^T \mathcal{P}^T X^T Y \tag{8}$$

Note that we have neglected the term $Y^T Y$ because it is a constant scalar and does not affect the optimal solution to this unconstrained optimization problem. Observe that Eq. (8) now has the form of a QUBO problem, as desired. Hence, we can solve this optimization problem using an adiabatic quantum computer.

Computational complexity. The regression problem (Problem 2) has $\mathcal{O}(Nd)$ data (X and Y) and $\mathcal{O}(d)$ weights (w), which is the same for Problem (8). We introduced K binary variables for each of the $d+1$ weights when converting Problem (2) to Problem (8). So, we have $\mathcal{O}(dK)$ variables in Eq. (8), which translates to quadratic qubit footprint ($\mathcal{O}(K^2 d^2)$) using an efficient embedding algorithm such as the one proposed by Date et al.[5]

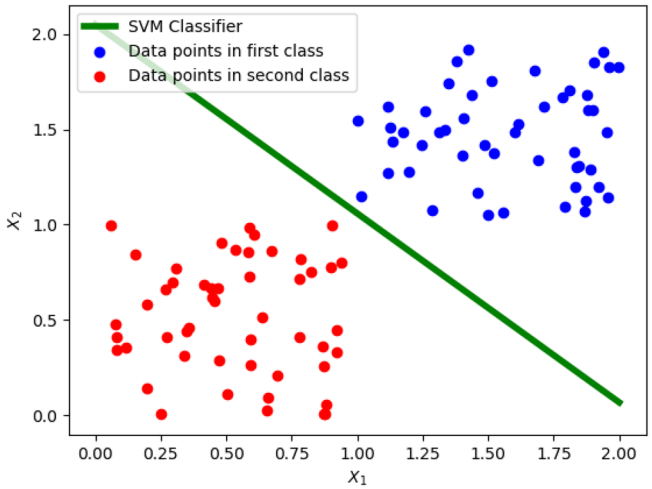

Figure 2. SVM model (green line) correctly classifying training data (red and blue dots).

Embedding is the process of mapping logical QUBO variables to qubits on the hardware, and is challenging because inter-qubit connectivity on the hardware is extremely limited. So, the space complexity of our approach is $\mathcal{O}(K^2d^2)$.

Solving the regression problem takes $\mathcal{O}(Nd^2)$ time classically. We analyze the time complexity of our approach in three parts: (i) Time taken to convert the regression problem into QUBO problem; (ii) Time taken to embed the QUBO problem onto the hardware; and (iii) Time taken to perform quantum annealing. From Eq. (8), we can infer that the conversion takes $\mathcal{O}(Nd^2K^2)$ time. Since we have $\mathcal{O}(dK)$ variables in the QUBO formulation, embedding can be done in $\mathcal{O}(d^2K^2)$ time using the embedding algorithm proposed by Date et al.[5]. While the theoretical time complexity of quantum annealing to obtain an exact solution is known to be exponential ($\mathcal{O}(e^{\sqrt{d}})$)[35], a more realistic estimate of the running time can be made by using measures such as ST99 and ST99(OPT)[36], which give the expected number of iterations to reach a certain level of optimality with 99% certainty. Quantum annealing is known to perform well on problems where the energy barriers between local optima are tall and narrow because such an energy landscape is more conducive to quantum tunneling. In order to estimate ST99 and ST99(OPT) for our approach, details on specific instances of the regression problem are required. It remains out of the scope of this paper to estimate ST99 and ST99(OPT) for generic QUBO formulation of the regression problem.

Having said that, we would like to shed some light on the quantum annealing running times observed in practice. An adiabatic quantum computer can only accommodate finite-sized problems—for example, D-Wave 2000Q can accommodate problems having 64 or fewer binary variables requiring all-to-all connectivity[5]. For problems within this range, a constant annealing time and a constant number of repetitions seem to work well in practice. So, the total time to convert and solve a linear regression problem on adiabatic quantum computer would be $\mathcal{O}(Nd^2K^2)$.

It may seem that this running time is worse than its classical counterpart. However, the above analysis assumes that K is variable. On classical computers, the precision is fixed, for example, 32-bit or 64-bit precision. We can analogously fix the precision for quantum computers, and interpret K as a constant. The resulting qubit footprint would be $\mathcal{O}(d^2)$, and the time complexity would be $\mathcal{O}(Nd^2)$, which is equivalent to the classical approach.

Support vector machine (SVM)

Background. Support vector machine (SVM) is a powerful supervised machine learning model that produces robust classifiers as shown in Fig. 2. The classifier produced by SVM maximizes its distance from the classes of the data points. Although SVM was meant for binary classification originally, several variants of SVM have been proposed over the years that allow multi-class classification[37,38]. SVM has wide ranging applications in multimedia (vision, text, speech etc.)[39], biology[40], and chemistry[41], among many other scientific disciplines.

Some quantum approaches for training SVM using adiabatic quantum computers have been proposed in the literature. Ahmed proposes a formulation for quantum SVM that runs on noisy intermediate-scale quantum (NISQ) processors[42]. Welsh et al. propose a formulation of SVM for the D-Wave quantum computers[43]. Our findings improve upon their formulation, allowing for real-valued learning parameters up to a certain precision.

Given training data $X \in \mathbb{R}^{N\times d}$ and training labels $Y \in \{-1, +1\}^N$, we would like to find a classifier (determined by weights, $w \in \mathbb{R}^d$, and bias, $b \in \mathbb{R}$), that separates the training data. Formally, training SVM is expressed as:

$$\begin{aligned} &\min_{w,\,b} \ \frac{1}{2}||w||^2 \\ &\text{subject to:} \qquad y_i(w^T x_i + b) \geq 1 \qquad \forall i = 1,\ 2,\ \ldots,\ N \end{aligned} \tag{9}$$

Note that x_i is the ith row vector in X and y_i is the ith element in Y. The objective function is convex because its Hessian matrix is positive semidefinite. Furthermore, since the constraints are linear, they are convex as well, which makes Problem (9) a quadratic programming problem. To solve Problem (9), we first compute the Lagrangian as follows:

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \lambda_i \Big[y_i(w^T x_i + b) - 1 \Big] \tag{10}$$

where, λ is the vector containing all the Lagrangian multipliers, i.e. $\lambda = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_N]^T$, with $\lambda_i \geq 0 \quad \forall i$. The non-zero Lagrangian multipliers in the final solution correspond to the support vectors and determine the hyperplanes H_1 and H_2 in Fig. 2. The Lagrangian dual problem (Eq. 10) is solved in $\mathcal{O}(N^3)$ time on classical computers by applying the Karush-Kuhn-Tucker (KKT) conditions[44,45]. As part of the KKT conditions, we set the gradient of $\mathcal{L}(w, b, \lambda)$ with respect to w to zero. We also set the partial derivative of $\mathcal{L}(w, b, \lambda)$ with respect to b to zero. Doing so yields:

$$\nabla_w \mathcal{L}(w, b, \lambda) = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \qquad \Longrightarrow \qquad w = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{11}$$

$$\frac{\partial \mathcal{L}(w, b, \lambda)}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{12}$$

Substituting Eqs. (11) and (12) into Eq. (10):

$$\mathcal{L}(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j x_i x_j y_i y_j \tag{13}$$

Note that Eq. (13) is a function of λ only. We want to maximize Eq. (13) with respect to the Lagrangian multipliers, and also ensure that $\lambda_i, \lambda_j \geq 0 \quad \forall i, j$, while satisfying Eq. (12).

QUBO formulation. In order to convert SVM training into a QUBO problem, we write Eq. (13) as a minimization problem:

$$\min_{\lambda} \ \mathcal{L}(\lambda) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j x_i x_j y_i y_j - \sum_{i=1}^{N} \lambda_i \qquad \lambda_i, \lambda_j \geq 0 \quad \forall i, j \tag{14}$$

This can be written in a matrix form as follows:

$$\min_{\lambda} \mathcal{L}(\lambda) = \frac{1}{2} \lambda^T (XX^T \odot YY^T) \lambda - \lambda^T 1_N \qquad \lambda \geq 0_N \tag{15}$$

where, 1_N and 0_N represent N-dimensional vectors of ones and zeros respectively, and $\odot$ is the element-wise multiplication operation.

We now reintroduce the K-dimensional precision vector $P = [p_1, p_2, \ldots, p_K]^T$ as described in the "Linear regression" section of this paper, but only allow positive powers of 2 in order to impose the non-negativity constraint on λ. We also introduce K binary variables $\hat{\lambda}_{ik}$ for each Lagrangian multiplier such that:

$$\lambda_i = \sum_{k=1}^{K} p_k \hat{\lambda}_{ik} \qquad \forall i = 1, 2, \ldots, N \tag{16}$$

where, p_k denotes the kth entry in the precision vector P. Next, we vertically stack all binary variables:

$$\hat{\lambda} = [\hat{\lambda}_{11} \ \ldots \ \hat{\lambda}_{1K} \ \hat{\lambda}_{21} \ \ldots \ \hat{\lambda}_{2K} \ \ldots \ \hat{\lambda}_{N1} \ \ldots \ \hat{\lambda}_{NK}]^T \tag{17}$$

We now define the precision matrix as follows:

$$\mathcal{P} = I_N \otimes P^T \tag{18}$$

Notice that:

$$\lambda = \mathcal{P} \hat{\lambda} \tag{19}$$

Finally, we substitute the value of λ from Eq. (19) into Eq. (15):

$$\min_{\hat{\lambda} \in \mathbb{B}^{NK}} \ \mathcal{L}(\hat{\lambda}) = \frac{1}{2} \hat{\lambda}^T \mathcal{P}^T (XX^T \odot YY^T) \mathcal{P} \hat{\lambda} - \hat{\lambda}^T \mathcal{P}^T 1_N \tag{20}$$

Equation (20) is identical to Eq. (1) with $z = \hat{\lambda}$, $A = \frac{1}{2} \mathcal{P}^T (XX^T \odot YY^T) \mathcal{P}$, $b = -\mathcal{P}^T 1_N$, and $M = KN$. Hence, we have converted the SVM training problem from Eq. (10) into a QUBO problem in Eq. (20), which can be solved on adiabatic quantum computers.

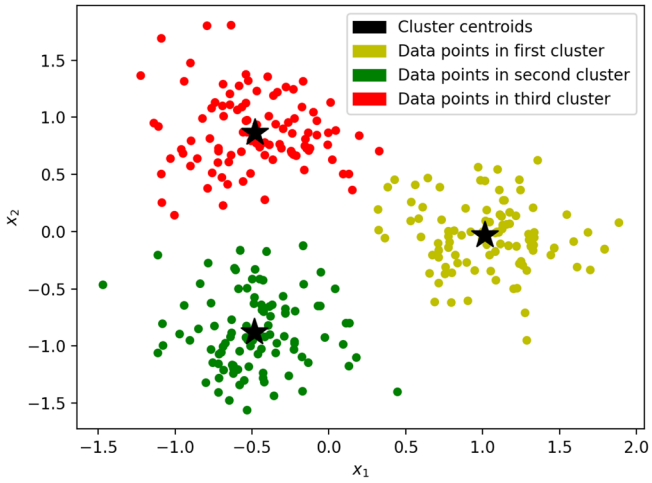

Figure 3. Training a balanced k-means clustering model ($k = 3$) on training data (yellow, green, and red dots).

Computational complexity. We begin our theoretical analysis by defining the space complexity with respect to the number of qubits needed to solve the QUBO. The SVM training problem stated in Eq. (15) contains $\mathcal{O}(N)$ variables (λ) and $\mathcal{O}(Nd)$ data (X and Y). The QUBO formulation of the SVM training problem stated in Eq. (20) consists of the same amount of data. However, as part of the QUBO formulation, we introduced K binary variables for each Lagrangian multiplier in the original problem (Eq. 15). So, the total number of variables in Eq. (20) is $\mathcal{O}(KN)$. So, the qubit footprint (or space complexity) of this formulation would be $\mathcal{O}(N^2K^2)$ after embedding onto the hardware.

The time complexity of classical SVM algorithms is known to be $\mathcal{O}(N^3)$[46]. We analyze the time complexity for training an SVM model in three parts as outlined in "Linear regression" section. Firstly, the time complexity for converting Problem (9) into a QUBO problem can be inferred from Eqs. (14) and (16) as $\mathcal{O}(N^2K^2)$. Secondly, the time taken to embed the (NK)-sized QUBO problem on the quantum computer is $\mathcal{O}(N^2K^2)$ (see "Linear regression" section for more details). Lastly, for the reasons mentioned in the "Linear regression" section, it is not straight forward to get a realistic estimate of the time complexity of the quantum annealing process. However, a constant annealing time in conjunction with a constant number of repetitions seems to work well in practice on an adiabatic quantum computer of fixed and finite size as explained in "Regression" section. So, the total time complexity is $\mathcal{O}(N^2K^2)$.

Note that the qubit footprint $\mathcal{O}(N^2K^2)$ and time complexity $\mathcal{O}(N^2K^2)$ assume that K is a variable. If the precision for all parameters ($\hat{\lambda}$) is fixed (e.g. limited to 32-bit or 64-bit precision), then K becomes a constant factor. The resulting qubit footprint would be $\mathcal{O}(N^2)$, and time complexity would also be be $\mathcal{O}(N^2)$. This time complexity is an order of magnitude better than the classical algorithm ($\mathcal{O}(N^3)$).

Balanced *k*-means clustering

Background. k-Means clustering is an unsupervised machine learning model that partitions training data into k clusters such that each point belongs to the cluster with the nearest centroid. The optimal cluster assignments of the training data minimizes within-cluster variance. Balanced k-means clustering is a special case of the k-means model where each cluster contains approximately N/k points as shown in Fig. 3. Balanced clustering models have applications in a variety of domains including network design[47], marketing[48], and document clustering[49].

Quantum approaches to training clustering models have been discussed in the literature. Ushijima-Mwesigwa et al. demonstrate partitioning a graph into k parts concurrently using quantum annealing on the D-Wave 2X machine[50]. Kumar et al. present a QUBO formulation for k-clustering that differs from the k-means model[51]. Bauckhage et al. propose a QUBO formulation for binary clustering ($k = 2$)[52] and k-medoids clustering[53]. Our QUBO formulation for balanced k-means clustering synthesizes a number of ideas proposed in the literature.

Given training data $X \in \mathbb{R}^{N\times d}$, we would like to partition the N data points into k clusters $\Phi = \{\phi_1, ..., \phi_k\}$. Let the centroid of cluster ϕ_i be denoted as μ_i. Formally, training the generic k-means clustering model is expressed as:

$$\min_{\Phi} \sum_{i=1}^{k} \frac{1}{2|\phi_i|} \sum_{x,y \in \phi_i} ||x - y||^2 \tag{21}$$

In the case that each cluster is of equal size, $|\phi_i|$ is constant, and Problem (21) reduces to:

$$\min_{\Phi} \sum_{i=1}^{k} \sum_{x,y \in \phi_i} ||x - y||^2 \tag{22}$$

Note that for most applications of balanced clustering, the cluster sizes are only approximately equal to one another. In these cases, the solution to Problem (22) may not be the exact solution to Problem (21). Classically, the k-means clustering problem is solved heuristically through an iterative approach known as Lloyd's algorithm. A modified version of this algorithm is used for balanced k-means clustering to uphold the constraint that no cluster contains more than N/k points[54]. This modified version of Lloyd's algorithm runs in $\mathcal{O}(N^{3.5}k^{3.5})$ time on classical computers[55].

QUBO formulation. To formulate Problem (22) as a QUBO problem, it will be useful to define a matrix $D \in \mathbb{R}^{N\times N}$ where each element is given by:

$$d_{ij} = ||x_i - x_j||^2 \tag{23}$$

where x_i and x_j are the ith and jth data points in X. We also define a binary matrix $\hat{W} \in \mathbb{B}^{N\times k}$ such that $\hat{w}_{ij} = 1$ if and only if point x_i belongs to cluster ϕ_j. Since we are assuming clusters of the same size, each column in $\hat{W}$ should have approximately N/k entries equal to 1. Additionally, since each data point belongs to exactly one cluster, each row in $\hat{W}$ must contain exactly one entry equal to 1. Using this notation, the inner sum in Problem (22) can be rewritten:

$$\sum_{x,y\in\phi_j} ||x - y||^2 = \hat{w}_j'^T D \hat{w}_j' \tag{24}$$

where $\hat{w}_j'$ is the jth column in $\hat{W}$. From this relation, we can cast Problem (22) into a constrained binary optimization problem. First, we vertically stack the Nk binary variables in $\hat{W}$ as follows:

$$\hat{w} = [\hat{w}_{11} \ \dots \ \hat{w}_{N1} \ \hat{w}_{12} \ \dots \ \hat{w}_{N2} \ \dots \ \hat{w}_{1k} \ \dots \ \hat{w}_{Nk}]^T \tag{25}$$

Provided the constraints on $\hat{w}$ are upheld, Problem (22) is equivalent to:

$$\min_{\hat{w}} \ \hat{w}^T (I_k \otimes D)\hat{w} \tag{26}$$

where I_k is the k-dimensional identity matrix.

We can add the constraints on $\hat{w}$ by including penalty terms that are minimized when all conditions are satisfied. First, we account for the constraint that each cluster must contain approximately N/k points. For a given column $\hat{w}_j'$ in $\hat{W}$, this can be enforced by including a penalty of the form:

$$\alpha(\hat{w}_j'^T \hat{w}_j' - N/k)^2 \tag{27}$$

where α is a constant factor intended to make the penalty large enough that the constraint is always upheld. Dropping the constant term $\alpha(N/k)^2$, this penalty is equivalent to $\hat{w}_j'^T \alpha F \hat{w}_j'$ where F is defined as:

$$F = 1_N - \frac{2N}{k} I_N \tag{28}$$

Using this formulation, the sum of all column constraint penalties is:

$$\hat{w}^T (I_k \otimes \alpha F)\hat{w} \tag{29}$$

Next, we account for the constraint that each point belongs to exactly 1 cluster. For a given row $\hat{w}_i$, this can be enforced by including a penalty of the form:

$$\beta(\hat{w}_i^T \hat{w}_i - 1)^2 \tag{30}$$

where β is a constant with the same purpose as α in Eq. (27). Dropping the constant term, this penalty is equivalent to $\hat{w}_i^T \beta G \hat{w}_i$ where G is defined as:

$$G = 1_k - 2I_k \tag{31}$$

To find the sum of all row constraint penalties, we first convert the binary vector $\hat{w}$ into the form $\hat{v}$ shown below:

$$\hat{v} = [w_{11} \dots w_{1k} \ w_{21} \dots w_{2k} \dots w_{N1} \dots w_{Nk}]^T \tag{32}$$

This can be accomplished through a linear transformation $Q\hat{w}$ where each element in $Q \in \mathbb{B}^{Nk\times Nk}$ is defined as:

$$q_{ij} = \begin{cases} 1 & j = N\text{mod}(i-1,k) + \lfloor \frac{i-1}{k} \rfloor + 1 \\ 0 & \text{else} \end{cases} \tag{33}$$

After the transformation, the sum of all row constraint penalties is given by $\hat{v}^T (I_N \otimes \beta G)\hat{v}$. This can be equivalently expressed as:

$$\hat{w}^T Q^T (I_N \otimes \beta G) Q \hat{w} \tag{34}$$

Combining the penalties from Eqs. (29) and (34) with the constrained binary optimization problem from Eq. (26), Problem (22) can be rewritten as:

$$\min_{\hat{w}} \ \hat{w}^T(I_k \otimes (D + \alpha F) + Q^T(I_N \otimes \beta G)Q)\hat{w} \tag{35}$$

Equation (35) is identical to Eq. (1) with $z = \hat{w}$, $A = (I_k \otimes (D + \alpha F) + Q^T(I_N \otimes \beta G)Q)$, and $b = 0$. Thus, we have converted Eq. (22) into a QUBO problem which can be solved on adiabatic quantum computers.

Computational complexity. The balanced k-means clustering problem stated in Eq. (22) contains $\mathcal{O}(Nd)$ data and $\mathcal{O}(N)$ variables. In our QUBO formulation, we introduce k binary variables for each variable in the original problem. Thus, the total number of variables in Eq. (35) is $\mathcal{O}(Nk)$. This translates to a quadratic qubit footprint of $\mathcal{O}(N^2k^2)$.

While an exact solution to the generic k-means clustering model (Problem 21) requires $\mathcal{O}(N^{kd+1})$ time[56], a classical algorithm for balanced k-means clustering will converge to a locally optimal solution in $\mathcal{O}(N^{3.5}k^{3.5})$ time[55]. To compute the time complexity for converting Eq. (22) into a QUBO problem, we can rewrite Eq. (35) as follows:

$$\min_{W} \sum_{l=1}^{k}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{m=1}^{d} w_{il}(x_{im} - x_{jm})^2 w_{jl} + \alpha \sum_{l=1}^{k}\sum_{j=1}^{N}\sum_{i=1}^{N} w_{il} f_{ij} w_{jl} + \beta \sum_{l=1}^{N}\sum_{j=1}^{k}\sum_{i=1}^{k} w_{li} g_{ij} w_{lj} \tag{36}$$

From Eq. (36), the time complexity is $\mathcal{O}(N^2kd)$, which is dominated by the first term. Embedding a QUBO problem having $\mathcal{O}(Nk)$ variables takes $\mathcal{O}(N^2k^2)$ time using the embedding algorithm proposed by Date et al.[5]. For the reasons mentioned in the "Linear regression" section, it is not straight forward to get a realistic estimate of the time complexity of the quantum annealing process. However, a constant annealing time in conjunction with a constant number of repetitions seems to work well in practice on an adiabatic quantum computer of fixed and finite size as explained in the "Linear regression" section. Therefore, the total time complexity for the quantum algorithm is $\mathcal{O}(N^2k(d + k))$. This time complexity is better than the worst case time complexity of the classical algorithm ($\mathcal{O}(N^{3.5}k^{3.5})$). However, the number of iterations in the classical algorithm varies greatly depending on the quality of the initial guess at the cluster centroids. In some cases, the classical algorithm may converge in much less than $\mathcal{O}(N^{3.5}k^{3.5})$ time and outperform its quantum counterpart.

Conclusion

As the task of training machine learning models becomes more computationally intensive, devising new methods for efficient training has become a crucial pursuit in machine learning. The process of training a given model can often be formulated as a problem of minimizing a well-defined error function for a given machine learning model. Given the power of quantum computers to approximately solve certain hard optimization problems with great efficiency as well as the demonstration of quantum supremacy by Google, we believe quantum computers can accelerate training of machine learning models. In this paper, we posed the training problems for three machine learning models (linear regression, support vector machine, and balanced k-means clustering) as QUBO problems to be solved on adiabatic quantum computers like D-Wave 2000Q. Furthermore, we analyzed the associated time and space complexity of our formulations and provided a theoretical comparison to the state-of-the-art classical methods for training these models. Our results are promising for training machine learning models on quantum computers in the future.

In the future, we would like to empirically evaluate the performance of our quantum approaches on real quantum computers. We would also like to compare the performance of our quantum approaches to state-of-the-art classical approaches. Finally, we would like to formulate other machine learning models such as logistic regression, restricted Boltzmann machines, deep belief networks, Bayesian learning and deep learning as QUBO problems that could potentially be trained on adiabatic quantum computers.

Received: 16 July 2020; Accepted: 15 April 2021
Published online: 11 May 2021

References

1. Obermeyer, Z. & Emanuel, E. J. Predicting the future—Big data, machine learning, and clinical medicine. *N. Engl. J. Med.* **375**, 1216 (2016).
2. Yatchew, A. Nonparametric regression techniques in economics. *J. Econ. Lit.* **36**, 669–721 (1998).
3. McQueen, R. J., Garner, S. R., Nevill-Manning, C. G. & Witten, I. H. Applying machine learning to agricultural data. *Comput. Electron. Agric.* **12**, 275–293 (1995).
4. Gyongyosi, L. & Imre, S. A survey on quantum computing technology. *Comput. Sci. Rev.* **31**, 51–71 (2019).
5. Date, P., Patton, R., Schuman, C. & Potok, T. Efficiently embedding qubo problems on adiabatic quantum computers. *Quantum Inf. Process.* **18**, 117 (2019).
6. Schaeffer, S. E. Graph clustering. *Comput. Sci. Rev.* **1**, 27–64 (2007).
7. Dill, K. A., Ozkan, S. B., Shell, M. S. & Weikl, T. R. The protein folding problem. *Ann. Rev. Biophys.* **37**, 289–316. https://doi.org/10.1146/annurev.biophys.37.092707.153558 (2008).
8. Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
9. Gyongyosi, L. & Imre, S. Quantum circuit design for objective function maximization in gate-model quantum computers. *Quantum Inf. Process.* **18**, 225 (2019).
10. Gyongyosi, L. & Imre, S. Circuit depth reduction for gate-model quantum computers. *Sci. Rep.* **10**, 1–17 (2020).
11. Gyongyosi, L. Quantum state optimization and computational pathway evaluation for gate-model quantum computers. *Sci. Rep.* **10**, 1–12 (2020).
12. Gyongyosi, L. & Imre, S. Optimizing high-efficiency quantum memory with quantum machine learning for near-term quantum devices. *Sci. Rep.* **10**, 1–24 (2020).
13. Unruh, W. G. Maintaining coherence in quantum computers. *Phys. Rev. A* **51**, 992 (1995).

14. Bennett, C. H., DiVincenzo, D. P., Smolin, J. A. & Wootters, W. K. Mixed-state entanglement and quantum error correction. *Phys. Rev. A* **54**, 3824 (1996).
15. Date, P. & Potok, T. Adiabatic quantum linear regression. arXiv preprint arXiv:2008.02355 (2020).
16. Date, P., Schuman, C., Patton, R. & Potok, T. A classical-quantum hybrid approach for unsupervised probabilistic machine learning. In Future of Information and Communication Conference, 98–117 (Springer, 2019).
17. Arthur, D. & Date, P. Balanced k-means clustering on an adiabatic quantum computer. arXiv preprint arXiv:2008.04419 (2020).
18. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014).
19. Farhi, E., Goldstone, J., Gutmann, S. & Zhou, L. The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size. arXiv preprint arXiv:1910.08187 (2019).
20. Farhi, E., Gamarnik, D. & Gutmann, S. The quantum approximate optimization algorithm needs to see the whole graph: A typical case. arXiv preprint arXiv:2004.09002 (2020).
21. Lloyd, S. Quantum approximate optimization is computationally universal. arXiv preprint arXiv:1812.11075 (2018).
22. Otterbach, J. et al. Unsupervised machine learning on a hybrid quantum computer. arXiv preprint arXiv:1712.05771 (2017).
23. Farhi, E. & Neven, H. Classification with quantum neural networks on near term processors. arXiv preprint arXiv:1802.06002 (2018).
24. Gyongyosi, L. & Imre, S. Training optimization for gate-model quantum neural networks. *Sci. Rep.* **9**, 1–19 (2019).
25. Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **4**, 043001 (2019).
26. Born, M. & Fock, V. Beweis des adiabatensatzes. *Zeitschrift für Physik* **51**, 165–180 (1928).
27. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. arXiv preprint quant-ph/0001106 (2000).
28. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **58**, 5355 (1998).
29. Leatherbarrow, R. J. Using linear and non-linear regression to fit biochemical data. *Trends Biochem. Sci.* **15**, 455–458 (1990).
30. Dielman, T. E. *Applied Regression Analysis for Business and Economics* (Duxbury/Thomson Learning Pacific Grove, 2001).
31. Paras, S. M. *et al.* A simple weather forecasting model using mathematical regression. *Indian Res. J. Ext. Educ.* **12**, 161–168 (2016).
32. Borle, A. & Lomonaco, S. J. Analyzing the quantum annealing approach for solving linear least squares problems. *In International Workshop on Algorithms and Computation*, 289–301 (Springer, 2019).
33. Chang, T. H., Lux, T. C. & Tipirneni, S. S. Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Inf. Process.* **18**, 374 (2019).
34. Chang, C. C., Gambhir, A., Humble, T. S. & Sota, S. Quantum annealing for systems of polynomial equations. *Sci. Rep.* **9**, 1–9 (2019).
35. Mukherjee, S. & Chakrabarti, B. K. Multivariable optimization: Quantum annealing and computation. *Eur. Phys. J. Special Top.* **224**, 17–24 (2015).
36. Wang, C. & Jonckheere, E. Simulated versus reduced noise quantum annealing in maximum independent set solution to wireless network scheduling. *Quantum Inf. Process.* **18**, 1–25 (2019).
37. Bo, G. & Xianwu, H. Svm multi-class classification. *J. Data Acquis. Process.* **21**, 334–339 (2006).
38. Cheong, S., Oh, S. H. & Lee, S.-Y. Support vector machines with binary tree architecture for multi-class classification. *Neural Inf. Process. Lett. Rev.* **2**, 47–51 (2004).
39. Moreno, P. J., Ho, P. P. & Vasconcelos, N. A Kullback-Leibler divergence based kernel for svm classification in multimedia applications. *Advances in Neural Information Processing Systems* 1385–1392, (2004).
40. Byvatov, E. & Schneider, G. Support vector machine applications in bioinformatics. *Appl. Bioinform.* **2**, 67–77 (2003).
41. Ivanciuc, O. *et al.* Applications of support vector machines in chemistry. *Rev. Comput. Chem.* **23**, 291 (2007).
42. Ahmed, S. Pattern recognition with Quantum Support Vector Machine (QSVM) on near term quantum processors. Ph.D. thesis, Brac University (2019).
43. Willsch, D., Willsch, M., De Raedt, H. & Michielsen, K. Support vector machines on the d-wave quantum annealer. *Comput. Phys. Commun.* **248**, 107006 (2020).
44. Karush, W. Minima of functions of several variables with inequalities as side constraints. M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago (1939).
45. Kuhn, H. W. & Tucker, A. W. Nonlinear programming. In *Traces and Emergence of Nonlinear Programming*, 247–258 (Springer, 2014).
46. Bottou, L. & Lin, C.-J. Support vector machine solvers. *Large Scale Kernel Mach.* **3**, 301–320 (2007).
47. Gupta, G. & Younis, M. Load-balanced clustering of wireless sensor networks. In *IEEE International Conference on Communications, 2003. ICC '03*, Vol. 3, 1848–1852 (2003).
48. Ghosh, J. & Strehl, A. *Clustering and Visualization of Retail Market Baskets* 75–102 (Springer, 2005).
49. Banerjee, A. & Ghosh, J. Competitive learning mechanisms for scalable, incremental and balanced clustering of streaming texts. In *Proceedings of the International Joint Conference on Neural Networks, 2003* Vol. 4, 2697–2702 (2003).
50. Ushijima-Mwesigwa, H., Negre, C. F. A. & Mniszewski, S. M. Graph partitioning using quantum annealing on the d-wave system. ArXiv arXiv:1705.03082 (2017).
51. Kumar, V., Bass, G., Tomlin, C. & Dulny, J. Quantum annealing for combinatorial clustering. *Quantum Inf. Process.* **17**, 1–14 (2018).
52. Bauckhage, C., Ojeda, C., Sifa, R. & Wrobel, S. Adiabatic quantum computing for kernel k= 2 means clustering. In *LWDA* 21–32, (2018).
53. Bauckhage, C., Piatkowski, N., Sifa, R., Hecker, D. & Wrobel, S. A qubo formulation of the k-medoids problem. In *LWDA* 54–63, (2019).
54. Ganganath, N., Cheng, C. & Tse, C. K. Data clustering with cluster size constraints using a modified k-means algorithm. In *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* 158–161 (2014).
55. Malinen, M. I. & Fränti, P. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition* (eds Fränti, P. *et al.*) 32–41 (Springer, 2014).
56. Inaba, M., Katoh, N. & Imai, H. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, SCG '94*, 332–339, https://doi.org/10.1145/177424.178042 (Association for Computing Machinery, 1994).

Acknowledgements

This manuscript has been authored in part by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan). This research used resources of the Oak Ridge Leadership Computing Facility, which

is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work was funded in part by the DOE Office of Science, High-energy Physics Quantised program. This work was funded in part by the DOE Office of Science, Advanced Scientific Computing Research (ASCR) program.

Author contributions

P.D. contributed to formulating the linear regression and support vector machine training problems. D.A. contributed to formulating the balanced k-means clustering training problem. L.P. contributed in writing the introduction and conclusion of the paper.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to P.D.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.