9

Machine learning with classical data

There has been significant recent interest in exploring the interplay between quantum computing and machine learning. Quantum resources and quantum algorithms have been studied in all major parts of the traditional machine learning pipeline: (i) the dataset, (ii) data processing and analysis, (iii) the machine learning model leading to a hypothesis family, and (iv) the learning algorithm (see [151, 242, 293] for reviews). In this chapter, we predominantly focus on quantum approaches for the latter three categories—that is, here we mostly consider quantum algorithms applied to classical data. These approaches include algorithms hinging on the quantum linear system solver (or quantum linear algebra more generally) as the source for possible quantum speedup over classical learning algorithms. These also include *quantum neural networks* (using the framework of variational quantum algorithms) and *quantum kernels*, where the classical machine learning model is replaced with a quantum model. Additionally, in this chapter, we discuss quantum algorithms that aim to speed up data analysis tasks, namely, *tensor principal component analysis (TPCA)* and *topological data analysis*.

Quantum machine learning is an active area of research. As such, we expect the conclusions made in this chapter to evolve over time, as new results are discovered. At present, our evaluation suggests that few of the considered quantum machine learning algorithms show any promise of quantum advantage in the immediate future. This conclusion stems from a number of factors, including issues of loading classical data into the quantum device and extracting classical data via tomography, and the success of classical "dequantized" algorithms [976]. More specialized tasks such as tensor PCA and topological data analysis may provide larger polynomial speedups (i.e., better than quadratic) in some regimes, but their application scope is less broad. Finally, other techniques such as quantum neural networks and quantum kernel meth-

ods contain heuristic elements which make it challenging to perform concrete analytic end-to-end resource estimates [915].

The authors are grateful to Ewin Tang for reviewing Section 9.1, to Eric Anschuetz for reviewing Section 9.2, to Matthew Hastings and Robin Kothari for reviewing Section 9.3, to Vedran Dunjko for reviewing Sections 9.4 and 9.5, and to Marco Cerezo for reviewing Section 9.5.

9.1 Quantum machine learning via quantum linear algebra

Overview

Linear algebra in high-dimensional spaces with tensor product structure is the workhorse of quantum computation as well as of much of machine learning (ML). It is therefore unsurprising that efforts have been made to find quantum algorithms for various learning tasks, including but not restricted to cluster-finding [707], principal component analysis [708], least-squares fitting [917, 609], recommendation systems [608], binary classification [866], and Gaussian process regression [1089]. One of the main computational bottlenecks in all of these tasks is the manipulation of large matrices. Significant speedup for this class of problems has been argued for via quantum linear algebra, as exemplified by the quantum linear system solver (QLSS). The main question marks for viability are (i) can quantum linear algebra be fully dequantized [977] for ML tasks, (ii) can the classical training data be loaded efficiently into a quantum random access memory (QRAM), and (iii) do the quantum ML algorithms that avoid the above-mentioned pitfalls address relevant machine learning problems? Our current understanding suggests that significant quantum advantage would require an exceptional confluence of (i)–(iii) that has not yet been found in the specific applications analyzed to date, though modest speedups are plausible.

ML applications

The structure of this section differs from other sections in Part I, due to the disparate nature of many of the quantum machine learning proposals and the fact that they are often heuristic. Rather than cover every proposal, we explore three specific applications. Each example explains which end-to-end problem is being solved and roughly how the proposed quantum algorithm solves that problem, arriving at its dominant complexity. In each case, the quantum algorithm assumes access to fast coherent data access (log-depth QRAM) and leverages quantum primitives for solving linear systems (and linear algebra

more generally). Under certain conditions, these primitives can be exponentially faster than classical methods that manipulate all the entries of vectors in the exponentially large vector space. However, for these examples, it is crucial to carefully define the end-to-end problem, as exponential advantages can be lost at the readout step, where the answer to a machine learning question must be retrieved from the quantum state encoding the solution to the linear algebra problem. In the three examples below, this is accomplished with some form of amplitude or overlap estimation, a primitive that brings a multiplicative $O(1/\epsilon)$ factor into the overall complexity when seeking precision ϵ. This $O(1/\epsilon)$ readout cost could be avoided if it were the case that $O(1)$ *samples* from the output state prepared by quantum linear algebra were sufficient for solving the end-to-end problem—situations where this may arise include training large neural networks by solving nonlinear differential equations [701] and determining an optimal set of random features in kernel-based supervised learning [1064]—but we do not cover these examples in detail here.

Furthermore, even if these quantum algorithms are exponentially faster than classical algorithms that manipulate the full state vector, in some cases this speedup has been "dequantized" via classical algorithms that merely sample from the entries of the vector. Specifically, for some end-to-end problems, there exist classical "quantum-inspired" algorithms [977, 271, 924] that solve the problem in time only polynomially slower than the quantum algorithm assuming an analogous data-input model. Namely, the assumption that the quantum algorithm has fast QRAM access to the classical data is analogous to the assumption that the classical algorithm has fast "sample-and-query" (SQ) access to the data—SQ access allows the classical algorithm to sample an entry from the database with probability proportional to its value squared, or to compute the value of any specific entry of the database. The reason that it is fair to compare quantum algorithms relying on QRAM access with classical algorithms relying on SQ access is that both utilize a certain tree-like data structure to enable fast implementation at the circuit level. A large, one-time cost (scaling linearly in the total size of the classical dataset) may be required to "load" the data structure with the classical data, but the data structure is dynamic in the sense that if a single entry in the database is added or changed, updating the data structure has low cost (scaling logarithmically in the total size of the classical dataset). Once the data structure has been set up, one can implement QRAM access (resp. SQ access) using a quantum (resp. classical) circuit with depth only logarithmic in the size of the database. We will not cover the particulars of the quantum-inspired algorithms in more detail, but we note that most of the machine learning tasks based on linear algebra for which quantum

algorithms have been proposed have also been dequantized in some capacity [271].

However, it is worth emphasizing that in some cases it remains possible that there could be an exponential quantum advantage if the quantum algorithm is able to exploit additional structure in the matrices involved, such as sparsity, that the classical algorithm cannot. The three examples below roughly illustrate the spectrum of possibilities: some tasks are fully dequantized, whereas others, to the best of our current knowledge, could still support exponential advantages if certain conditions are met.

Example 1: Gaussian process regression

Actual end-to-end problem: Gaussian process regression (GPR) is a non-parametric, Bayesian method for regression. GPR is closely related to kernel methods [594], as well as to other regression models, including linear regression [858]. Our presentation of the problem follows that of [858, Chapter 2] and [1088]. Given training data $\{x_j, y_j\}_{j=1}^M$, with inputs $x_j \in \mathbb{R}^N$ and noisy outputs $y_j \in \mathbb{R}$, the goal is to model the underlying function $f(x)$ generating the output y

$$y = f(x) + \epsilon_{\text{noise}},$$

where ϵ_{noise} is drawn from i.i.d. Gaussian noise with variance σ^2. Modeling $f(x)$ as a Gaussian process means that for inputs $\{x_j\}_{j=1}^M$, the outputs $\{f(x_j)\}_{j=1}^M$ are treated as random variables with a joint multivariate Gaussian distribution, in such a way that any subset of these values are jointly normally distributed in a manner consistent with the global distribution. While this multivariate Gaussian distribution governing $\{f(x_j)\}_{j=1}^M$ will generally be correlated for different j, the additional additive error ϵ_{noise} on our observations y_j is independent from the choice of $f(x_j)$ and uncorrelated from point to point. The Gaussian process is specified by the distribution $\mathcal{N}(m, K)$ where m is the length-M vector obtained by evaluating a "mean function" $m(x)$ at the points $\{x_j\}_{j=1}^M$, and K is an $M \times M$ covariance kernel matrix obtained by evaluating a covariance kernel function $k(x, x')$ at $x, x' \in \{x_j\}_{j=1}^M$—$\mathcal{N}$ then denotes the multivariate Gaussian distribution with the corresponding mean and covariance. The functional form of the mean and covariance kernel are specified by the user and determine the properties of the Gaussian process, such as its smoothness.[1] These functions typically contain a small number of hyperparameters which can be optimized using the training data. A commonly used covariance kernel function is the squared exponential covariance function $k(x, x') = \exp\left(-\frac{1}{2\ell^2}\|x - x'\|^2\right)$ where

[1] This can be visualized by sampling a function from the distribution, which means sampling a value of $f(x_j)$ from the distribution for each x_j, and plotting the values of $f(x_j)$ as a curve.

ℓ is a hyperparameter controlling the length scale of the Gaussian process, and $\|\cdot\|$ denotes the standard Euclidean norm in the case of vector arguments and the spectral norm in the case of matrix arguments.

Given choices for $m(x)$ and $k(x, x')$ and the observed data $\{x_j, y_j\}_{j=1}^M$, our task is to predict the value $f(x_*)$ of a new test point x_*. Because the Gaussian process assumes that all $M + 1$ values $\{f(x_1), \ldots, f(x_M), f(x_*)\}$ have a jointly Gaussian distribution, it is possible to condition upon the observed data to obtain the distribution for $f(x_*)$ which is $p(f_*|x_*, \{x_j, y_j\}) \sim \mathcal{N}(\bar{f}_*, \mathbb{V}[f_*])$. Our goal is to compute $\bar{f}_*$, the mean (linear predictor) of the distribution for $f(x_*)$, as well as the variance $\mathbb{V}[f_*]$, which gives uncertainty on the prediction. Computing the underlying multivariate Gaussian distribution can be bypassed by exploiting the closure of Gaussians under linear operations, in particular, conditioning. This re-expresses the problem as linear algebra with the kernel matrix. Assuming the common choice of $m(x) = 0$ and defining the length-M vector $k_* \in \mathbb{R}^M$ to have its j-th entry given by $k(x_*, x_j)$, we obtain

$$\bar{f}_* = k_*^\intercal [K + \sigma^2 I]^{-1} y$$
$$\mathbb{V}[f_*] = k(x_*, x_*) - k_*^\intercal [K + \sigma^2 I]^{-1} k_*$$

which characterize the prediction for the test point. The advantages of GPR are a small number of hyperparameters, model interpretability, and that it naturally returns uncertainty estimates for the predictions. Its main disadvantage is the computational cost.

Dominant resource cost/complexity: In classical implementations, the cost is dominated by performing the inversion $[K+\sigma^2 I]^{-1}$, typically via a Cholesky decomposition, resulting in a complexity of $O(M^3)$ (see [858, Chapter 8] and [698] for approximations used to reduce the classical cost). In [1089], a quantum algorithm was proposed that leverages the QLSS to perform this inversion more efficiently. The quantum computer uses the classical data to infer the linear predictor and variance for a test point x_*, and this process must be repeated for the computation of each new test point output. We analyze the complexity of computing $\bar{f}_*$, with a simple extension for $\mathbb{V}[f_*]$. Given classically observed/precomputed values of y and k_*, the quantum algorithm uses state preparation from classical data (based on QRAM) to prepare quantum states representing $|y\rangle$ and $|k_*\rangle$,[2] each with a gate depth of $O(\log(M))$ (though using $O(M)$ gates overall). The algorithm also uses a block-encoding of classical data (also using QRAM) for $A := [K + \sigma^2 I]$, with a normalization factor

[2] For any vector v, the notation $|v\rangle$ denotes the normalized quantum state whose amplitudes in the computational basis are proportional to the entries of v, for example, $|y\rangle = \frac{1}{\|y\|} \sum_j y_j |j\rangle$.

of $\alpha = \|K + \sigma^2 I\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm.[3] The state-of-the-art QLSS has complexity $O(\alpha\kappa\|A\|^{-1}\log(1/\epsilon))$ calls to an α-normalized block-encoding of matrix A with condition number κ (see Eq. (18.2)). In this case, the minimum singular value of A is at least σ^2, so $\kappa/\|A\| \leq \sigma^{-2}$. The QLSS produces the normalized state $|A^{-1}y\rangle$, and a similar approach yields an estimate for the norm $\|A^{-1}y\|$ to relative error ϵ at cost $\widetilde{O}(\alpha\kappa\|A\|^{-1}\epsilon^{-1})$ [248, 327]. Given unitary circuits performing these tasks, we can estimate the quantity $\bar{f}_* = \langle k_*|A^{-1}y\rangle \cdot \|k^*\| \cdot \|A^{-1}y\|$ to precision ϵ using overlap estimation with gate depth upper bounded by

$$\widetilde{O}\left(\log(M) \cdot \|K + \sigma^2 I\|_F \sigma^{-2} \cdot \frac{\|k_*\|\left\|[K + \sigma^2 I]^{-1}y\right\|}{\epsilon}\right),$$

where the three factors come from state preparation (i.e., QRAM), QLSS, and overlap estimation, respectively. Using QRAM for state preparation as described above would use $O(M^2)$ ancilla qubits. Note that classical "quantum-inspired" methods for solving linear systems, based on SQ access, also have $\text{poly}(\|A\|_F, \kappa, \epsilon^{-1}, \log(M))$ complexity [271, 433, 924], and thus the quantum algorithm as stated above offers at most a polynomial speedup in the case of dense matrices.

On the other hand, [1089] considers the case where the vectors and kernels are sparse[4] and uses this to reduce the cost of the quantum algorithm and of QRAM. In this case, using block-encodings of sparse matrices, the factor $\|A\|_F$ in the complexity expression is replaced by a factor $s\|A\|_{\max}$, where s is the sparsity of the matrix A and $\|A\|_{\max}$ is the maximum magnitude of any entry of A—log-depth QRAM with $\Omega(M)$ ancilla qubits would still be necessary to implement the sparse access oracle to the sM arbitrary nonzero entries of A in depth $O(\log(M))$. The upshot is that in the sparse case, because the algorithm assumes the kernel is not low rank, this algorithm is not dequantized by SQ access [271] and may still offer an exponential speedup over quantum-inspired methods. However, we note that the assumption of sparsity in $[K + \sigma^2 I]$ may also enable the use of more efficient classical algorithms for computing the inverse (see Chapter 18 on QLSSs). Moreover, we must include the classical precomputation of evaluating the entries of this matrix. A related, and similarly efficient, quantum algorithm is proposed in [1088] for optimizing the

[3] It may be more efficient to load in the $\{x_j\}$ values and then coherently evaluate the kernel entries using quantum arithmetic. Some ideas in this direction are explored in [264]. One might also consider block-encoding K and $\sigma^2 I$ separately and combining them with linear combination of unitaries.

[4] For the squared exponential covariance function mentioned above, the kernel matrix will not be sparse, but [1089] notes several applications of GPR where sparsity is well justified.

hyperparameters of the GPR kernel by maximizing the marginal likelihood of the observed data given the model.

Example 2: Support vector machines

Actual end-to-end problem: The task for the support vector machine (SVM) is to classify an N-dimensional vector x_* into one of two classes ($y_* = \pm 1$), given M labeled data points of the form $\{(x_j, y_j) : x_j \in \mathbb{R}^N, y_j = \pm 1\}_{j=1,\ldots,M}$ used for training. The training phase solves a continuous optimization problem to find a maximum-margin hyperplane, described by normal vector $w \in \mathbb{R}^M$ and offset $b \in \mathbb{R}$, which separates the training data. That is, data points with $y_j = 1$ lie on one side of the plane, and data points with $y_j = -1$ lie on the other side. Once trained, the classification of x_* is inferred via the formula

$$y_* = \text{sign}(b + \langle w, x_* \rangle), \tag{9.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product between vectors.

In the "hard-margin" version of the problem where all training points must be classified correctly (assuming it is possible to do so, i.e., the data is linearly separable), the solution (w, b) is given by

$$\underset{(w,b)}{\text{argmin}} \, \|w\|^2, \qquad \text{subject to:} \qquad (\langle w, x_j \rangle + b) y_j \geq 1 \qquad \forall j. \tag{9.2}$$

In the "soft-margin" version of the problem, the hyperplane need not correctly classify all training points. The relation $(\langle w, x_j \rangle + b) y_j \geq 1$ is relaxed to $(\langle w, x_j \rangle + b) y_j \geq 1 - \xi_j$, with $\xi_j \geq 0$. Now, (w, b) are determined by

$$\underset{(w,b,\xi)}{\text{argmin}} \, \|w\|^2 + \gamma \|\xi\|_1, \quad \text{subject to:} \quad (\langle w, x_j \rangle + b) y_j \geq 1 - \xi_j \quad \forall j, \tag{9.3}$$

where $\|\cdot\|_1$ denotes the vector 1-norm, and γ is a user-specified hyperparameter related to how much to penalize points that lie within the margin. Both Eqs. (9.2) and (9.3) are convex programs, in particular, quadratic programs, which can also be rewritten as second-order cone programs [612]. Another feature of these formulations is that the solution vectors w and ξ are usually sparse; the j-th entry is only nonzero for values of j where x_j lies on or within the margin near the hyperplane—these x_j are called the "support vectors."

In [972], a "least-squares" version of the SVM problem was proposed, which has no inequality constraints:[5]

$$\underset{(w,b,\xi)}{\text{argmin}} \, \|w\|^2 + \frac{\gamma}{M} \|\xi\|^2, \quad \text{subject to:} \quad (\langle w, x_j \rangle + b) y_j = 1 - \xi_j \quad \forall j. \tag{9.4}$$

[5] Our definition of the least-squares SVM is equivalent to the normal presentation found in [972, 866]; however, we choose slightly different conventions for normalization of certain parameters, such as γ, with respect to M. The goal of our choices is to make the final complexity expression free of any explicit M dependence.

This is an equality-constrained least-squares problem, which is simpler than a quadratic program and can be solved using Lagrange multipliers and inverting a linear system. Specifically, one introduces a vector $\beta \in \mathbb{R}^M$ and solves the $(M+1) \times (M+1)$ linear system $Au = v$, where

$$A = \begin{pmatrix} 0 & \mathbf{1}^\intercal / \sqrt{M} \\ \mathbf{1} / \sqrt{M} & K/M + \gamma^{-1} I \end{pmatrix}, \qquad u = \begin{pmatrix} b \\ \beta \end{pmatrix}, \qquad v = \frac{1}{\sqrt{M}} \begin{pmatrix} 0 \\ y \end{pmatrix}, \tag{9.5}$$

with K the kernel matrix for which $K_{ij} = \langle x_i, x_j \rangle$, $\mathbf{1}$ the all-ones vector, and I the identity matrix. The vector w is inferred from β via the formula $w = \sum_j \beta_j x_j / \sqrt{M}$.

However, unlike the first two formulations, the least-squares formulation does not generally have sparse solution vectors (w, b) (see [971]). Additionally, its solution can be qualitatively different, due to the fact that correctly classified data points can lead to negative ξ_j that apply penalties to the objective function through the appearance of $\|\xi\|^2$.

Dominant resource cost/complexity: The hard-margin and soft-margin formulations of SVM are quadratic programs, which can be mapped to second-order cone programs and solved with quantum interior point methods (QIPMs). This solution was proposed in [612], and, assuming access to log-depth QRAM it can find ϵ-accurate estimates for the solution (w, b) in time scaling as $\widetilde{O}(M^{0.5}(M+N)\kappa_{\text{IPM}}\zeta \log(1/\epsilon)/\xi')$, where κ_{IPM}, ζ, and ξ' are instance-specific parameters related to the QIPM. This compares to $O(M^{0.5}(M+N)^3 \log(1/\epsilon))$ for naively implemented classical interior point methods. In [612], numerical simulations on random SVM instances were performed to compute these instance-specific parameters, and the results were consistent with a small polynomial speedup. However, the resource estimate of [328] for a related problem suggests a practical advantage may be difficult to realize with this approach.

The least-squares formulation can be solved directly with the QLSS, as pursued in [866]. This can be compared to classically solving the linear system via Gaussian elimination, with cost $O(M^3)$. The QLSS requires the ability to prepare the state $|v\rangle$, which can be accomplished in $O(\log(M))$ depth through methods for preparation of states from classical data, although requiring $O(M)$ total gates and ancilla qubits. One also needs a block-encoding of the matrix A. One method is through block-encodings from classical data, which requires classical precomputation of the $O(M^2)$ entries of K (incurring classical cost $O(M^2 N)$) and producing a block-encoding with normalization factor $\alpha = \|A\|_F$ (Frobenius norm). Henceforth, we assume that $\|x_j\| \leq 1$ for all j, which can always be achieved by scaling down the training data (inducing a

scaling up of w and $\sqrt{\gamma}$ by an equal factor). This implies $\|K/M\|_F \leq 1$ and hence $\|A\|_F \leq \sqrt{2} + 1 + \sqrt{M}\gamma^{-1}$. A better block-encoding can be obtained by block-encoding K/M via the method for Gram matrices[6] and $\gamma^{-1}I$ via the trivial method, and then combining these with the rest of A via linear combination of block-encodings. This avoids the need to classically calculate the inner products $\langle x_i, x_j\rangle$, and has a better normalization $\alpha \leq \sqrt{2} + 1 + \gamma^{-1}$.

Given these constructions, the QLSS outputs the state $|u\rangle = (b|0\rangle + \sum_{j=1}^{M} \beta_j |j\rangle)/\sqrt{b^2 + \|\beta\|^2}$; the cost is $\widetilde{O}(\alpha\kappa_A/\|A\|)$ queries to the block-encoding of A, where κ_A is the condition number of A. We may assert that $\|A\| \geq 1$. This follows by noting that the lower-right block of A (as defined in Eq. (9.5)) is positive semidefinite, and that 1 is an eigenvalue of A when the lower-right block is set to zero. The condition number should be upper bounded by an M-independent function of γ due to the appearance of the regularizing $\gamma^{-1}I$.

Reading out all $M + 1$ entries of $|u\rangle$ via tomography would multiply the cost by $\Omega(M)$. However, in [866], it was observed that to classify a test point x_* via Eq. (9.1), one can use overlap estimation rather than classically learning the solution vector. In our notation and normalization, this can be carried out as follows. Let $|x_j\rangle = \sum_{i=1}^{M} x_{ji}|i\rangle/\|x_j\|$, with x_{ji} denoting the i-th entry of the vector x_j. Starting with $|u\rangle$, we prepare $|x_j\rangle$ into an ancilla register, using methods for controlled state preparation from classical data, forming

$$|\tilde{u}\rangle = \frac{b|0\rangle|0\rangle + \sum_{j=1}^{M} \beta_j |j\rangle\Big(\|x_j\||x_j\rangle + \sqrt{1 - \|x_j\|^2}|M+1\rangle\Big)}{\sqrt{b^2 + \|\beta\|^2}}.$$

One also creates a reference state $|\tilde{x}_*\rangle$ encoding x_*, defined as

$$|\tilde{x}_*\rangle = \frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2M}} \sum_{j=1}^{M} |j\rangle\Big(\|x_*\||x_*\rangle + \sqrt{1 - \|x_*\|^2}|M+2\rangle\Big).$$

The right-hand side of Eq. (9.1) is then given by $\sqrt{2}\sqrt{b^2 + \|\beta\|^2}\langle\tilde{u}|\tilde{x}_*\rangle$. Thus, the overlap $\langle\tilde{u}|\tilde{x}_*\rangle$ must be estimated to precision $\epsilon = 1/\sqrt{2(b^2 + \|\beta\|^2)}$ in order to distinguish ± 1 and classify x_*. Additionally, the norm $\|u\| = \sqrt{b^2 + \|\beta\|^2}$ must be calculated; this can separately be done to relative error ϵ' at cost

[6] We sketch a possible instantiation of this method here. Define $|x_i\rangle = \|x_i\|^{-1} \sum_{k=1}^{M} x_{ik}|k\rangle$ where x_{ik} is the k-th entry of x_i. Suppose $M = 2^m$ is a power of 2. Following the setup in block-encodings and [431, Lemma 47], we must define sets of M orthonormal states $\{|\psi_i\rangle\}$ and $\{|\phi_j\rangle\}$. We choose $|\psi_i\rangle = (\|x_i\||x_i\rangle + \sqrt{1 - \|x_i\|^2}|M+1\rangle)(H^{\otimes m}|i\rangle)|0^m\rangle$, where H denotes the Hadamard transform. We choose $|\phi_j\rangle = (\|x_j\||x_j\rangle + \sqrt{1 - \|x_j\|^2}|M+2\rangle)|0^m\rangle(H^{\otimes m}|j\rangle)$. These states can be prepared in $O(\log(M))$ depth using $O(M)$ total gates and ancilla qubits with methods for controlled state preparation from classical data. It can be verified that these sets are orthonormal, and that $\langle\psi_i|\phi_j\rangle = \langle x_i, x_j\rangle/M$. Hence, the Gram matrix construction yields a block-encoding of K/M with normalization factor 1.

$\widetilde{O}(\alpha\kappa_A/\epsilon')$ (see Chapter 18 on QLSSs). We may also note that as $u = A^{-1}v$ and $\|v\| = 1$, we have $\|u\| \leq \kappa_A/\|A\|$. Thus, the overall circuit depth required to classify x_* is

$$\widetilde{O}\left(\frac{\alpha\kappa_A^2}{\|A\|^2}\right).$$

There is no explicit poly(N, M) dependence. However, for certain datasets and parameter choices, such dependence could be hidden in κ_A or α, making an apples-to-apples comparison with Gaussian elimination less clear.

Furthermore, this task has been dequantized under the assumption of SQ access [350, 271, 924]. In time scaling as poly$(\|A\|_F, \epsilon^{-1}, \log(NM))$, one can classically sample from the solution vector $|u\rangle$ to error ϵ, and furthermore, one can estimate inner products $\langle\tilde{u}|\tilde{v}\rangle$ in time $O(1/\epsilon^2)$ [976].[7] However, the cost can be reduced through a trick that is analogous to how the quantum algorithm can block-encode the $\gamma^{-1}I$ part of A separately to avoid the dependence on a large $\|A\|_F$. In particular, [271, Corollary 6.18] gives a classical complexity that would be polynomially related to the quantum complexity above under appropriate matching of parameters, but the power of this polynomial speedup could still be significant. In any case, such a speedup crucially requires log-depth QRAM access to the training data, which requires total gate complexity $\Omega(NM)$ and $O(NM)$ ancilla qubits.

Example 3: Supervised cluster assignment

Actual end-to-end problem: Suppose we are given access to a vector $x \in \mathbb{C}^N$ and a set of M samples $\{y_j \in \mathbb{C}^N\}_{j=1,\ldots,M}$. We want to estimate the distance between x and the centroid of the set $\{y_j\}$ to judge whether x was drawn from the same set as $\{y_j\}$. If we have multiple sets $\{y_j\}$, we can infer that x belongs to the one for which the distance is shortest; as a result, this is also called the "nearest-centroid problem." Specifically, the computational task is to estimate $\|x - \frac{1}{M}Y\mathbf{1}\|$ to additive constant error ϵ with probability $1 - \delta$, where $Y \in \mathbb{C}^{N\times M}$ is the matrix whose columns are y_j, and $\mathbf{1}$ is the vector of M ones—the vector $Y\mathbf{1}/M$ is the centroid of the set.

Dominant resource cost/complexity: Naively computing the centroid incurs classical cost $O(NM)$. In [707], a quantum solution to this problem was proposed. Let $\bar{x} = x/\|x\|$ and let $\bar{Y}$ be normalized so that all columns have unit

[7] The method of doing so is succinct to describe (see, e.g., [978]). First, one uses sample access to the vector $\tilde{u}$ to generate an index i at random, with probability $|\tilde{u}_i|^2/\|\tilde{u}\|^2$. Then, one uses query access to $\tilde{u}$ and $\tilde{v}$ to compute the quantity $\mathcal{R} = (\tilde{v}_i\|\tilde{u}\|)/(\tilde{u}_i\|\tilde{v}\|)$. The expectation value of $\mathcal{R}$ is precisely $\langle\tilde{u}|\tilde{v}\rangle$, and the variance is upper bounded by 1. Thus, an estimate of $\langle\tilde{u}|\tilde{v}\rangle$ to ϵ precision is obtained by averaging $O(1/\epsilon^2)$ samples of the random variable $\mathcal{R}$.

norm. Define the $N \times (M+1)$ matrix R and length-$(M+1)$ vector w as follows:

$$R = \begin{pmatrix} \bar{x} & \bar{Y}/\sqrt{M} \end{pmatrix}, \qquad w = \begin{pmatrix} \|x\| \\ -1_Y/\sqrt{M} \end{pmatrix},$$

where 1_Y is the length-M vector containing the norms of the columns of Y, defined such that $\bar{Y}1_Y = Y\mathbf{1}$. Then, $Rw = x - \frac{1}{M}Y\mathbf{1}$. Using methods for block-encoding and state preparation from classical data, one constructs $O(\log(NM))$-depth circuits that block-encode R (with normalization factor $\|R\|_F = 2$) and prepare the state $|w\rangle$. If we apply the block-encoding of R to $|w\rangle$ and measure the block-encoding ancillas, the probability that we obtain $|0\rangle$ is precisely $(\|Rw\|/2\|w\|)^2$. Thus, using amplitude estimation, one learns $\|Rw\|$ to precision ϵ with probability at least $1-\delta$ at cost $O(\|w\| \log(1/\delta)/\epsilon)$ calls to the log-depth block-encoding and state preparation routines.

The advantage over naive classical methods essentially boils down to the assumption of efficient classical data loading for a specific dataset. Subsequently, this quantum algorithm was dequantized, and it was understood that a similar feat is possible classically in the SQ access model [977]. Specifically, the classical algorithm runs in time $\widetilde{O}(\|w\|^2 \log(1/\delta)/\epsilon^2)$, reducing the exponential speedup to merely quadratic.

Caveats

The overwhelming caveat in these and other proposals is access to the classical data in quantum superposition. These quantum machine learning algorithms assume that we can load a vector of N entries or a matrix of N^2 entries in polylog(N) time. While efficient quantum data structures, that is, QRAM, have been proposed for this task, they introduce a number of caveats. In order to coherently load N pieces of data in $\log(N)$ time, QRAM uses a number of ancilla qubits, arranged in a tree structure. To load data of size N, the QRAM data structure requires $O(N)$ qubits, which is exponentially larger than the $O(\log(N))$ data qubits used in the algorithms above. This spatial complexity does not yet include the overheads of quantum error correction and fault-tolerant computation, in particular the large spatial resources required to distill magic states in parallel. As such, we do not yet know if it is possible to build a QRAM that can load the data sufficiently quickly, while maintaining moderate spatial resources.

In addition, achieving speedups by efficiently representing the data as a quantum state may suggest that classical methods based on tensor networks could achieve similar performance, in some settings. Taking this line of reasoning to the extreme, a number of efficient classical algorithms have been developed by "dequantizing" the quantum algorithms. That is, by assuming an

analogous access model (the SQ access model) to the training data, as well as some assumptions on sparsity and/or rank of the inputs, there exist approximate classical sampling algorithms with polynomial overhead as compared to the quantum algorithms [977, 976]. This means that any apparent exponential speedup must be an artifact of the data loading/data access assumptions.

A further caveat is inherited from the QLSS subroutine, which is that the complexity is large when the matrices involved are ill conditioned. This caveat is somewhat mitigated in the Gaussian process regression and support vector machine examples above, where the matrix to be inverted is regularized by adding the identity matrix.

End-to-end resource analysis

To the best of our knowledge, full end-to-end resource estimation has not been performed for any specific quantum machine learning tasks.

Outlook

Much of the promise of quantum speedup for classical machine learning based on linear algebra hinges on the extent to which quantum algorithms can be dequantized. At present, the results of [977] seem to prohibit an exponential speedup for many of the problems proposed, but there is still the possibility of a large polynomial speedup. The most recent asymptotic scaling analysis [271] for dequantization methods still allows for a power 4 speedup in the Frobenius norm of the "data matrix" and a power 11 speedup in the polynomial approximation degree (see [86] for more details). However, the classical algorithms are steadily improving, and their scaling might be further reduced.

It is also worth noting that the classical probabilistic algorithms based on the SQ access model are not currently used in practice. This could be due to a number of reasons, including the poor polynomial scaling, the fact that the access model might not be well suited to many practical scenarios, or simply because the method is new and has not been tested in practice (see [61, 270] for some work in this direction).

On the other hand, some machine learning tasks based on quantum linear algebra are not known to be dequantized, such as Gaussian process regression under the assumption that the kernel matrix is sparse. In particular, avoiding dequantization and achieving an exponential quantum speedup appears to require that the matrices involved are simultaneously sparse, high rank, and well conditioned.[8] In this situation, quantum algorithm can leverage block-encodings

[8] Dequantization can also be avoided even when the matrices involved are dense, provided that they are given by a product of a small number of sparse matrices. For example, it was described in [1064] how an exponential speedup may be possible for a certain ML task, where

for which the normalization factor is equal to the sparsity, rather than general block-encodings of classical data for which the normalization factor is the Frobenius norm. The complexity of quantum-inspired classical algorithms based on SQ access will still grow polynomially with the Frobenius norm even when the matrices are sparse,[9] although other classical algorithms may be able to exploit the sparsity more directly. Perhaps unsurprisingly, the prototypical matrices that satisfy these criteria are sparse unitary matrices, such as those naturally implemented by a local quantum gate. For unitary matrices, the condition number is 1, and the Frobenius norm is equal to the square root of the Hilbert space dimension—exponentially large in the system size. (As a simple example, consider the identity matrix on n qubits.) A central question is whether situations like this occur in interesting end-to-end machine learning problems. Even if they do, an exponential speedup is not guaranteed. An additional hurdle arises in the quantum readout step, which incurs a cost scaling as the inverse in the precision target. To avoid exponential overhead, the end-to-end problem must not require exponentially small precision.

Further reading

For further reading on specific machine learning tasks where quantum algorithms have been proposed, we refer the reader to [151, 293, 916]. We have not covered dequantization techniques in great detail; for an accessible summary and perspective, see [978], and for a more detailed overview, see [979].

9.2 Quantum machine learning via energy-based models

Overview

An important class of models in machine learning is *energy-based models*, which are heavily inspired by statistical mechanics. The goal of energy-based

the matrix to be inverted is neither sparse nor low rank; rather, it is related to a sparse matrix via the discrete Fourier transform (a dense unitary matrix). A block-encoding for the relevant matrix is constructed by leveraging the quantum Fourier transform (QFT). Note that the QFT can be decomposed into a product of sparse matrices corresponding to the local unitary gates in the quantum circuit for QFT.

[9] For example, consider the complexity of algorithms for pseudoinversion of an s-sparse matrix A. Let the rank of A be r, which may be as large as the matrix dimension, and suppose all nonzero singular values of the matrix lie in the interval $[1/\kappa, 1]$. This implies that $\sqrt{r}/\kappa \leq \|A\|_F \leq \sqrt{r}$, and thus $\kappa\|A\|_F \geq \sqrt{r}$. As a consequence, the complexity of quantum-inspired algorithms with SQ access in [271, 433, 924]—scaling as $\mathrm{poly}(\|A\|_F, \kappa, 1/\epsilon)$—will necessarily grow as a polynomial of the matrix rank, even for fixed sparsity. On the other hand, the complexity of the quantum algorithm with QRAM that applies the QLSS and amplitude estimation—scaling as $\mathrm{poly}(s\|A\|_{\max}, \kappa, 1/\epsilon)$—is independent of the matrix rank for well-conditioned A (i.e., $\kappa = O(1)$).

models is to train a physical model (i.e., tune the interaction strengths between a set of particles) such that the model closely matches the training set when the model is in thermal equilibrium (made more precise below). Energy-based models are an example of *generative* models since, once they are trained, they can then be used to form new examples that are similar to the training set by sampling from the model's thermal distribution.

Due to their deep connection to physics, energy-based models are prime candidates for various forms of quantization. However, one challenge faced by quantum approaches is that the statistical mechanical nature of the learning problem also often lends itself to efficient, approximate classical methods. As a result, the best quantum algorithms may also be heuristic in nature, which prevents an end-to-end complexity analysis. While energy-based models are less widely used than deep neural networks today, they were an important conceptual development in machine learning [893] and continue to foster interest due to their sound theoretical basis, and their connection to statistical mechanics.

There are a number of proposals for generalizing energy-based models to quantum machine learning. The starting point is a graph where the vertices are divided into *visible* $\{v\}$ and *hidden* $\{h\}$ nodes. When each node is assigned a value in some discrete or continuous set, this constitutes a "configuration" (h, v) of the model. A training set $\mathcal{D}$ is provided as input, containing a list of configurations of the visible vertices. The hidden nodes are not part of the training set, but including them is essential for the model to be able to capture latent variables in the data.

A graphical model is then built on the vertices—each vertex is a physical system (such as a spin-1/2 particle) and edges between vertices represent physical interactions. The model is described by an energy functional $H(h, v)$, which assigns an energy value to each possible configuration (h, v) of the vertices. For example, in Boltzmann machines (BMs), the vertices are assigned binary variables, and the interactions are Ising interactions. The model can be used to generate samples (e.g., via Markov chain Monte Carlo methods) from the thermal distribution (also known as the Boltzmann distribution or the Gibbs distribution) at unit temperature, that is, the distribution where each configuration (h, v) is sampled with probability proportional to $\mathrm{e}^{-H(h,v)}$. In unsupervised learning tasks, provided a set of training samples of configurations of the visible units v, the goal is to tune the interaction weights of the model such that the model's thermal distribution best matches the distribution that generated the training set.

Quantum algorithms can potentially be helpful for training classical graphical models. One can also generalize the model itself by allowing the physical

systems on each vertex to be quantum, and interactions between systems to be noncommuting.

Actual end-to-end problem(s) solved

Classical graphical models: Let $G = (V, E)$ denote a graph with vertices V and edges E. For classical models, each vertex j is assigned a binary variable $z_j = \pm 1$. The variables are split into visible and hidden nodes, $z \in \{v\} \cup \{h\}$. For classical BMs, the energy functional is often taken to be quadratic[10] with weights $\{b_i, w_{ij}\}$:

$$H(z) = \sum_{i \in V} b_i z_i + \sum_{(i,j) \in E} w_{ij} z_i z_j. \tag{9.6}$$

Note that interactions can occur between any pair of nodes (hidden or visible). In the special case of a restricted Boltzmann machine (RBM), each edge must pair up a hidden node with a visible node (i.e., the graph is bipartite). This restriction makes the model less expressive than graphs with edges between hidden nodes, but it leads to simplifications for certain training approaches.

The thermal distribution corresponding to the energy functional (at unit temperature) associates each configuration v of visible nodes with a probability $p(v)$ such that

$$p(v) = \sum_h p(h, v), \qquad p(h, v) = \frac{\mathrm{e}^{-H(h,v)}}{\mathcal{Z}}, \qquad \mathcal{Z} = \sum_{h,v} \mathrm{e}^{-H(h,v)},$$

where $\mathcal{Z}$, the partition function, is the normalization to ensure probabilities sum to 1. Even though hidden nodes are integrated out in the calculation of $p(v)$, they impact the distribution of $p(v)$ through their interactions with the visible nodes.

Given a training set $\mathcal{D} = \{v_1, v_2, \ldots, v_{|\mathcal{D}|}\}$ of sample configurations of the visible nodes, the goal of the training phase is to modify the weights $\theta \in \{b_i\} \cup \{w_{ij}\}$ such that samples from the thermal distribution of the model most closely match the training samples. Ideally, this is done by finding the set of weights that maximizes the likelihood of observing the samples, that is, $\prod_{v \in \mathcal{D}} p(v)$, or, equivalently, minimizing the (normalized) log-likelihood loss function, defined as

$$L(b, w) = -\frac{1}{|\mathcal{D}|} \sum_{v \in \mathcal{D}} \log(p(v)). \tag{9.7}$$

[10] This quadratic energy functional is related to the Sherrington–Kirkpatrick (SK) model [933] with an external field, which is a model for spin glasses in the statistical mechanics literature. For the SK model, the couplings w_{ij} are chosen randomly for each pair of nodes, and it is typically computationally hard to find configurations with optimal energy (see Section 4.2 on beyond quadratic speedups in combinatorial optimization for additional information).

The loss function can be minimized using some variant of gradient descent, which requires the evaluation of the derivatives $\partial_\theta L$ for $\theta \in \{b_i\} \cup \{w_{ij}\}$. For the energy functional above, these derivatives can be readily calculated from ensemble averages (see, e.g., [1041]). For example,

$$\frac{\partial L}{\partial w_{ij}} = \langle z_i z_j \rangle_{v \in \mathcal{D}} - \langle z_i z_j \rangle \,, \tag{9.8}$$

where $\langle \cdot \rangle$ denotes an average over samples from the thermal distribution $p(h, v)$, while $\langle \cdot \rangle_{v \in \mathcal{D}}$ denotes an average where v is drawn at random from the training set $\mathcal{D}$, and h is sampled from the thermal distribution conditioned on that choice of v. Without any further restrictions, the gradients will typically be difficult to evaluate (or even to estimate accurately). An exact computation requires computing a sum over the exponential number of configurations of the vertices.

In some cases, good estimates of the gradients can be obtained by repeatedly drawing samples from the thermal distribution and computing averages. Samples can be generated with Markov chain Monte Carlo (MCMC) methods such as the Metropolis–Hastings algorithm or simulated annealing; however, the time required to sample from a distribution close to the thermal distribution depends on the mixing time of the Markov chain, which is generally unknown and can also be exponential in the graph size. Additionally, many samples need to be generated to produce a robust average, with precision ϵ requiring $O(1/\epsilon^2)$ samples. Approximate classical methods, such as contrastive divergence [531], avoid this issue by initializing the Markov chain at one of the training samples and deliberately taking a small number of steps—this does not exactly correspond to optimizing the log-likelihood but in some cases has empirical success [916]. Indeed, here we see the benefit of restricting to bipartite graphs in RBMs: since there are no edges between hidden nodes, the Gibbs distribution over the hidden nodes is independent from node to node, conditioned on a fixed setting of the visible nodes. This enables a simple exact calculation for the first term of Eq. (9.8), and it is also key to the success of estimating the second term with contrastive divergence, where the hidden layer and the visible layer are conditionally sampled in alternating fashion.

Once the model has been trained, new samples can also be generated via the same MCMC methods. The end-to-end tasks are (i) training the model, and then, (ii) generating samples from the trained model to accomplish some larger machine learning goal.

Quantum graphical models: A separate end-to-end problem is found by generalizing the model itself to be quantum. For example, one can start with a clas-

sical BM and promote the binary variables to qubits. The energy functional is promoted to a quantum Hamiltonian and augmented with a transverse field, which does not commute with the Ising interactions. The result is a quantum Boltzmann machine (QBM), described by a transverse-field Ising (TFI) Hamiltonian [30] (cf. Eq. (1.4)):

$$H_{\text{QBM}} = -\sum_{i\in V}(\kappa_i X_i + b_i Z_i) - \sum_{(i,j)\in E} w_{ij} Z_i Z_j \,, \tag{9.9}$$

where X_i and Z_i are the Pauli-X and Pauli-Z operators on qubit i, and b_i, κ_i, w_{ij} are real variational parameters of the model. The ground or Gibbs state of H_{QBM} can be prepared in a variety of ways, including the adiabatic algorithm, Gibbs sampling, or as a variational quantum algorithm. These states can be measured (in the Z basis or in the X basis), yielding samples of the variables v, h drawn from different distributions than the thermal distribution for the classical BM. Alternatively, one can trace out the hidden nodes, viewing the leftover quantum state on the visible nodes as the output of the QBM; this may be suitable if the input data is also quantum. As in the classical case, the training phase for a QBM consists of varying the weights via gradient descent to maximize a likelihood function. However, the noncommutativity of the Hamiltonian leads to complications: the gradients of the loss function are no longer directly given by sample expectation values as was the case in Eq. (9.8), although workarounds have been proposed [30, 614, 1037, 40, 1093]. For example, in the case of classical input data, sample expectation values can be used to optimize function that is not equal to the loss function, but can be shown to be an upper bound on it using the Golden–Thompson inequality [30]. Alternatively, in the case of quantum input data, and assuming the QBM has no hidden nodes, the relevant loss metric is the relative entropy and its gradients can again be related to sample averages [614]—this scenario is closely related to the Hamiltonian learning problem. In any case, the end-to-end problem is to train these models and generate samples.

Dominant resource cost/complexity

Complexity of classical graphical models: Recall that for classical BMs, one wishes to produce samples from the thermal distribution corresponding to the energy functional in Eq. (9.6), that is, Gibbs sampling (of diagonal Hamiltonians), either to assist in training the model or, if it has already been trained, to make inferences or generate new data. Specifically, given $H(h, v)$, one wishes to draw samples of (h, v) with probability proportional to $e^{-H(h,v)}$, either with v free or with v fixed (sometimes referred to as "clamped") to a particular value from the training set $\mathcal{D}$. Classically, one approach is simulated annealing or

other MCMC algorithms. Quantumly, one can take one of several analogous approaches, including "quantum simulated annealing" [944] and quantum annealing, discussed as follows.

For quantum simulated annealing, one prepares the coherent Gibbs state $\sum_{v,h} \sqrt{p(h,v)}|v,h\rangle$, and a quadratic speedup is obtained over classical simulated annealing. The method is to construct a Hamiltonian whose ground state is the coherent Gibbs state at temperature T (i.e., for which probabilities $p(h,v)$ are proportional to $e^{-H(h,v)/T}$), and follow an adiabatic path from $T = \infty$ to $T = 1$. Following the path is accomplished by repeatedly performing quantum phase estimation (QPE) to project onto the ground state of the Hamiltonian at a given temperature. As is typical for the adiabatic algorithm, the cost of this procedure is dominated by the inverse of the spectral gap—this is the precision required for QPE to succeed. Specifically, for a graphical model with $|V|$ vertices, the runtime will be $\text{poly}(|V|)/\Delta$, where Δ is the minimum spectral gap. Importantly, Δ can be related to the maximum mixing time t_{mix} of the simulated annealing Markov chain, as $1/\Delta = O(\sqrt{t_{\text{mix}}})$, which leads to the quadratic speedup, although it is possible that Δ is exponentially small in $|V|$.

An alternative method for preparing (and sampling from) the coherent Gibbs state was proposed in [1041]. There, one begins in an easy-to-prepare coherent mean-field state approximating the coherent Gibbs state. Then, one performs rejection sampling with amplitude amplification to gain a quadratic speedup over the analogous classical method. Additionally, it was proposed to use amplitude estimation to gain a quadratic improvement in the number of samples needed to achieve precision ϵ, from $O(1/\epsilon^2)$ to $O(1/\epsilon)$, mirroring later analyses that work for general quantum-accelerated Monte Carlo methods [773]. If these $O(1/\epsilon)$ quantum samples are each for the same training sample $v \in \mathcal{D}$, this is straightforward; however, if the samples are drawn randomly from $v \in \mathcal{D}$, achieving the quadratic speedup from amplitude estimation requires accessing the data in $\mathcal{D}$ coherently and quickly. Such data access is provided by the quantum random access memory (QRAM) primitive, for which the circuit *depth* can be logarithmic in the size of the training data, at the expense of a number of ancilla qubits (and total gates) that is linear in the size of the training data.

For quantum annealing, the idea is to add a uniform transverse field, as in the QBM of Eq. (9.9) with $\kappa_i = \kappa_j$ for all i, j. The transverse field is initially strong, and slowly turned off. This is similar to the adiabatic algorithm, but differs in that it is specifically carried out at finite ambient temperature. Thus, the system-bath interaction of the device naturally drives the state to the Gibbs state, which coincides with the classical thermal distribution once the transverse field is turned off. This is a heuristic method; it is efficient but there are

few success guarantees. The hope is that the inclusion of an initial transverse field induces nonclassical fluctuations that help the system avoid becoming trapped in local minima as the transverse field is turned off.

Overall, computing the gradient of the loss function with respect to one parameter, up to precision ϵ, will require complexity $O(S/\epsilon)$, where S is the complexity of sampling from the Gibbs state. The above assumes log-depth QRAM to be able to estimate the $\langle z_i z_j \rangle_{v\in\mathcal{D}}$ term of Eq. (9.8). The complexity of S will be poly($|V|$) $\sqrt{t_{\text{mix}}}$ if a quantum simulated annealing approach is used, or some hard-to-analyze quantity if the quantum annealing approach is used. If the number of training samples is small, one can also sequentially compute the sum over $v \in \mathcal{D}$ and avoid the assumption of log-depth QRAM, leading to complexity $O(S|\mathcal{D}|/\epsilon')$ (where $\epsilon' \geq \epsilon$ may be order-1). This must be carried out for all $|E| + |V|$ weights in the model, although these could be simultaneously estimated to precision ϵ at cost $\widetilde{O}(\sqrt{|E| + |V|}/\epsilon)$ samples, using methods from [549], which leverage the quantum gradient estimation primitive. It is not clear what value of ϵ is required in practice. Reference [1041] takes $\epsilon \sim 1/\sqrt{|\mathcal{D}|}$, to match the natural uncertainty coming from a finite number of training samples. In this case, the overall complexity is dominated by

$$\widetilde{O}\left(S \cdot \sqrt{|V| + |E|} \cdot \sqrt{|\mathcal{D}|}\right) \tag{9.10}$$

assuming log-depth QRAM, and

$$\widetilde{O}\left(S \cdot \sqrt{|V| + |E|} \cdot |\mathcal{D}|\right) \tag{9.11}$$

without log-depth QRAM (the precision for each training sample can be taken as $\epsilon' = O(1)$). The linear dependence on $|\mathcal{D}|$ could potentially be mitigated by first classically computing a "core set" $\mathcal{D}'$ satisfying the requirements that $|\mathcal{D}'| \ll |\mathcal{D}|$ and that replacing $\mathcal{D}$ with $\mathcal{D}'$ causes minimal change to the loss function in Eq. (9.7) [498].

Complexity of quantum graphical models: For QBMs, the dominant cost comes from producing samples from the quantum Gibbs state for the system in Eq. (9.9), that is, the state $\rho \propto \mathrm{e}^{-H_{\text{QBM}}}$, which can be accomplished through methods for Gibbs sampling. Rigorous methods for Gibbs sampling may scale exponentially in the size of the graph, without further assumptions. Such scaling would likely not be tolerable in practice. However, Monte Carlo–style methods for Gibbs sampling, which follow a similar approach as MCMC, but in an inherently quantum way, may be more effective in this case. These could have poly($|V|$) scaling for some parameter settings, but must also have exponential scaling in the worst case, as sampling low-energy Ising-model configurations is known to be NP-hard.

One can also heuristically apply quantum annealing, beginning from a large transverse field and reducing its strength slowly to some final nonzero value. However, some hardware platforms may only admit global control over the transverse field, preventing one from tuning the transverse-field strengths κ_i individually. In any of these approaches, it is difficult to make any rigorous statements about the Gibbs sampling complexity.

Existing resource estimates

There are no logical resource estimates for quantum annealing. However, [7, 116] discuss in detail how to embed the fully connected architecture of a RBM into the 2D lattice architecture available on planar quantum annealers. Reference [116] reports an embedding ratio scaling which is roughly quadratic—that is, a graphical model with $|V|$ vertices requires $O(|V|^2)$ qubits to accommodate the architectural limitations of the device. A proper resource estimation has not been performed for the fault-tolerant algorithm of [1041].

Caveats

There are two main caveats to quantum approaches to training classical models, which apply to both the annealing and to the fault-tolerant setting. First, classical heuristic algorithms, such as greedy methods or contrastive divergence, often perform well in practice and are the method of choice for existing classical analyses. These methods are also often highly parallelizable. If the quantum algorithm offers a speedup over a slower, exact classical method, this may not be relevant if the faster approximate classical methods are already sufficient. Second, the situations where one might hope for the heuristic quantum annealing approach to perform better might not be relevant problems, for instance, in highly regular lattice-based problems.

A caveat of the QBM is that the gradients of the loss function are not exactly related to sample averages, and imperfect workarounds, such as those proposed in [30], must be pursued. Like many other situations in machine learning, the resulting end-to-end solution is heuristic and evidence of its efficacy requires empirical demonstration.

Comparable classical complexity and challenging instance sizes

For classical models, an exact computation of the gradients would scale exponentially in the size of the graph, as $O(|\mathcal{D}|2^{|V|})$ for the gradient of a single parameter. Approximate methods based on simulated annealing or other MCMC methods would scale as $O(S_c/\epsilon^2)$, where S_c is the classical sample time, scaling as $S_c = \text{poly}(|V|)t_{\text{mix}}$. On the other hand, these methods can also be implemented heuristically at reduced cost (e.g., contrastive divergence, where

one does not wait for the chain to mix), and they can also be implemented on parallel architectures. For instance, in [618], an architecture was proposed to train deep BMs efficiently. Experiments demonstrated that heuristic training methods could be carried out for graphs of size 1 million in 100 seconds on field-programmable gate arrays available in 2010. Much larger sizes would be accessible to a scaled-up version of the same architecture on modern hardware. It is unlikely that any exact method, quantum or classical, could match this efficiency.

For the quantum models, the classical complexity of sampling from the Gibbs state of the model would be exponential in the graph size $|V|$. Thus, training these models would likely not be pursued classically.

Speedup

For the classical models, the speedup can be quadratic in most of the parameters: producing a sample can in some cases be sped up quadratically, and the number of samples required to achieve a certain precision also enjoys a quadratic speedup (e.g., t_{mix} to $\sqrt{t_{\text{mix}}}$ and $O(1/\epsilon^2)$ to $O(1/\epsilon)$). The methods that give these provable quadratic speedups are based on primitives such as amplitude amplification, where superquadratic speedups are not possible without exploiting additional structure. Larger superpolynomial speedups are only possible under optimistic assumptions about the success of heuristic quantum annealing approaches at producing samples faster than classical approaches.

For the quantum models, the speedup is technically exponential, assuming that for the models considered, quantum algorithms for Gibbs sampling scale efficiently while approximate classical methods (e.g., tensor networks) scale exponentially. Indeed, it was shown in [1042] that certain QBMs are BQP-complete, in the sense that its ground state can be efficiently prepared on a quantum computer, and any quantum computation (including those with exponential speedup) can be encoded into its ground state. However, this construction is artificial, and it has yet to be demonstrated that there are specific real-world machine learning tasks where these models offer a speedup over the best available classical machine learning model for the same task.

Outlook

While energy-based models are naturally in a form that can readily be extended to the quantum domain, there still lacks decisive evidence of quantum advantage for a specific end-to-end classical machine learning problem. There remains some uncertainty on the outlook of these approaches due to the centrality of heuristic quantum approaches. One may hold out hope that these heuristics could outperform classical heuristics in some specific settings, but

the success of classical heuristics and effectiveness of approximate classical approaches present a formidable barrier to achieving any quantum advantage in this area.

Further reading

We refer the reader to [916] for more information on quantum approaches to energy-based models.

9.3 Tensor PCA

Overview

Inference problems play an important role in machine learning. One of the most widespread methods is principal component analysis (PCA), a technique that extracts the most significant information from a stream of potentially noisy data. In the special case where the data is generated from a rank-1 vector plus Gaussian noise—the spiked matrix model—it is known that there is a phase transition in the signal-to-noise ratio [536]: above the transition point, the principal component can be recovered efficiently, while below the transition point, the principal component cannot be recovered at all. In the tensor extension of the problem, there are two transitions. One information theoretical, below which the principal component cannot be recovered, and another computational, below which the principal component can be recovered, but only inefficiently, and above which it can be recovered efficiently. Thus, the tensor PCA problem offers a much richer mathematical setting, which has connections to optimization and spin glass theory; however, it is yet unclear if the tensor PCA framework has natural practical applications. A quantum algorithm [509] for tensor PCA was proposed which has provable runtime guarantees for the spiked tensor model; it offers a potentially *quartic* speedup over its classical counterpart and also efficiently recovers the signal from the noise at a smaller signal-to-noise ratio than other classical methods. This algorithm was further developed in [904] and extended to also give a quartic speedup for a related discrete optimization problem called "planted noisy kXOR," which is argued to have possible relevance in cryptography.

Actual end-to-end problem(s) solved

Consider the spiked tensor problem. Let $v \in \mathbb{R}^N$ (or $\in \mathbb{C}^N$)[11] be an unknown signal vector, and let $p \in \mathbb{N}$ be a positive integer. Construct the tensor

$$T = \lambda v^{\otimes p} + V,$$

[11] Reference [509] provides reductions between real and complex cases.

where V is a random tensor in $\mathbb{R}^{N^p}$ (or $\mathbb{C}^{N^p}$), with each entry drawn from a normal distribution with mean 0 and variance 1. The vector v is assumed to have norm $\sum_j v_j^* v_j = \sqrt{N}$ and can be identified with a quantum state. The quantity λ is the signal-to-noise ratio.

The main question we are interested in is for what values of λ can we detect or reconstruct v from (full) access to T, and how efficiently can this be done? In [874], it was shown that the maximum likelihood solution w^{ML} to the objective function

$$w^{\mathrm{ML}} = \underset{w\in\mathbb{C}^n}{\operatorname{argmax}} \langle T, w^{\otimes p}\rangle$$

will have high correlation with v as long as $\lambda \gg N^{(1-p)/2}$, where $\langle\cdot,\cdot\rangle$ denotes the standard dot product after writing the N^p entries of the tensor as a vector. However, the best known *efficient* classical algorithm [1033] requires $\lambda \gg N^{-p/4}$ to recover an approximation of v. Using the spectral method, that is, mapping the tensor T to a $N^{p/2}\times N^{p/2}$ matrix and extracting the maximal eigenvalue, recovery can be done in time complexity $O(N^p)$, ignoring logarithmic prefactors.

Hastings [509] proposes classical and quantum algorithms to solve the spiked tensor model by first mapping T to a bosonic quantum Hamiltonian with N modes, n_{bos} bosons, and p-body interactions, where n_{bos} is a tunable integer parameter satisfying $n_{\mathrm{bos}} > p/2$

$$H_{\mathrm{PCA}}(T) = \frac{1}{2}\left(\sum_{\mu_1,\ldots,\mu_p=1}^{N} T_{\mu_1,\ldots,\mu_p}\left(\prod_{i=1}^{p/2} a_{\mu_i}^\dagger\right)\left(\prod_{j=1+p/2}^{p} a_{\mu_j}\right) + \mathrm{h.c.}\right), \tag{9.12}$$

where h.c. stands for Hermitian conjugate. Here, the operators a_μ and $a_\mu^\dagger$ are annihilation and creation operators of a boson in mode μ, and we restrict to the sector for which $\sum_\mu a_\mu^\dagger a_\mu = n_{\mathrm{bos}}$. As n_{bos} increases, the number of particles increases and the complexity of the algorithm grows, but the threshold for λ above which recovery is possible also decreases.

Hastings shows that the vector v can be efficiently recovered from a vector in the large energy subspace of $H_{\mathrm{PCA}}(T)$ when the largest eigenvalue of $H_{\mathrm{PCA}}(T)$ is at least a constant factor larger than $E_{\max}$, where $E_{\max}$ corresponds to the case where there is no signal. It is shown that, roughly,

$$E_{\max} \sim n_{\mathrm{bos}}^{p/4+1/2} N^{p/4},$$
$$E_0 \approx \lambda (p/2)! \binom{n_{\mathrm{bos}}}{p/2} N^{p/2} \approx \lambda n_{\mathrm{bos}}^{p/2} N^{p/2},$$

where E_0 is the maximum eigenvalue of $H_{\mathrm{PCA}}(T)$. Thus, if $\lambda \gg N^{-p/4}$, there will be a gap between E_0 and $E_{\max}$, and this gap grows as n_{bos} increases. This

enables the quantum algorithm to recover the signal even for signal strength as weak as $\lambda \gg N^{-p/4}$.

Hastings considers the case where p is constant and N grows, and assumes that $n_{\text{bos}} = O(N^\theta)$ for some p-dependent constant $\theta > 0$ chosen sufficiently small. In fact, ultimately, it is determined that in the recovery regime $\lambda \gg N^{-p/4}$, the parameter n_{bos} need only scale as $\text{polylog}(N)(N^{-p/4}/\lambda)^{4/(p-2)}$. In any case, terms in the complexity $O(N^p)$ are dominated by terms $O(N^{n_{\text{bos}}})$.

The idea of mapping the order-p tensor T to an order-2 tensor (i.e., a matrix) H_{PCA}, and then solving the planted inference problem by (classically) extracting spectral information from H_{PCA}, was also independently discovered in [1033], where it is called the "Kikuchi method." There, a tunable parameter ℓ plays the role of n_{bos}, although the two formulations offer different intuitions to motivate the method; their relationship is discussed in detail in [904].

Dominant resource cost/complexity

Hastings shows that the dominant eigenvector of H_{PCA} can be classically extracted in $\widetilde{O}(N^{n_{\text{bos}}})$ time via the power method, where the tilde indicates that we ignore polylogarithmic factors.

He proposes three quantum algorithms for the same problem. The first runs quantum phase estimation on a random state. Since the random state will have squared overlap $\Omega(N^{-n_{\text{bos}}})$ with the high-energy subspace, the expected number of repetitions of phase estimation is $O(N^{n_{\text{bos}}})$. The second algorithm proposes to further use amplitude amplification, reducing the complexity to $O(N^{n_{\text{bos}}/2})$. The third algorithm further improves the complexity by choosing a specific initial high-energy state, and showing that the overlap with the state scales as $\Omega(N^{-n_{\text{bos}}/2})$, which combined with amplitude amplification, leads to a $O(N^{n_{\text{bos}}/4})$ complexity. As discussed above, the estimates assume that factors of $O(N^p)$ can be ignored, since they are negligible with respect to the query complexity of $N^{O(n_{\text{bos}})}$.

This constitutes a quartic speedup over the classical spectral algorithm acting on H_{PCA} for the same choice of n_{bos} that is also presented in [509]. Since the ansatz state is a product state, it can be prepared efficiently.

Hastings further argues that the Hamiltonian simulation of H_{PCA} within the phase estimation subroutine can be accomplished by viewing H_{PCA} as a sparse or local matrix. Specifically, we can view H_{PCA} as a second-quantized Hamiltonian with N registers storing a $O(\log(n_{\text{bos}}))$-bit number representing the occupancy of each mode. The total number of qubits needed is $O(N \log(n_{\text{bos}}))$. Each of the $O(N^p)$ terms in Eq. (9.12) corresponds to a single nonzero entry of one row of H_{PCA} in this basis, so the $O(n_{\text{bos}}^N) \times O(n_{\text{bos}}^N)$ matrix H_{PCA} is $O(N^p)$-sparse. Alternatively, a more compact representation would be to view H_{PCA} in

a first-quantized picture, allocating n_{bos} registers each storing a $O(\log(N))$ bit number corresponding to which mode each of the n_{bos} bosons are in—the total number of qubits needed is $O(n_{\text{bos}} \log(N))$. In order to enforce permutational symmetry, each term in Eq. (9.12) would decompose into $(p/2)!\binom{n_{\text{bos}}}{p/2}$ separate terms that act locally on $p/2$ of the registers. This is closer to the approach taken in the quantum algorithm of [904].

Either way, we can efficiently perform Hamiltonian simulation (e.g., by first constructing a block-encoding of the sparse or local Hamiltonian) to perform quantum phase estimation at total gate complexity $\text{poly}(N, n_{\text{bos}})$ (here interpreting $p = O(1)$), which is negligible compared to $N^{n_{\text{bos}}}$.

Caveats

The spiked tensor model does not immediately appear to be related to any practical problems. Additionally, efficient recovery requires that the signal-to-noise ratio be rather high, which may not occur in real-world settings, and when it does, it is not clear that formulating the problem as a tensor PCA problem will be the most efficient path forward. Relatedly, while the runtime of the algorithm scales subexponentially in N, for large values of n_{bos}, its N dependence of $O(N^{n_{\text{bos}}})$ may still lead to a practically intractable algorithm.

Comparable classical complexity and challenging instance sizes

The algorithms proposed in [509] (see also [1033, 904]) improve on other spectral methods for the spiked tensor model, whenever $n_{\text{bos}} > p/2$ for sufficiently large p. The threshold for which the new algorithms beat the older ones decreases as n_{bos} increases, although the complexity of the algorithm increases with n_{bos}.

Speedup

The quartic speedup over the classical power method is achieved by combining a quadratic speedup from amplitude amplification with a quadratic speedup related to choosing a clever initial state for phase estimation. The existence of the clever initial state is essential for the beyond-quadratic speedup, and it has been related [904] to the BQP-hardness of the guided local/sparse Hamiltonian problem [419].

As discussed above, there is no readout issue, as the vector v can be efficiently recovered from the single particle density matrix obtained from the eigenvector of $H_{\text{PCA}}(T)$. The quantum algorithm has $O(N \log(n_{\text{bos}}))$ space complexity, which is an exponential improvement over the classical spectral algorithm presented in [509] for the same problem. Furthermore, the quartic speedup in time and exponential speedup in space is possible even in the

absence of a large-scale quantum random access memory (QRAM), since the overall runtime of the algorithm is much larger than the size of the input dataset ($O(N^p)$), and linear (rather than logarithmic) data access cost is tolerable.

In [904], the quartic speedup was extended to apply generally to instances where the "Kikuchi method" is used; improvements to the classical algorithm within this framework would imply commensurate improvements to the quantum algorithm to maintain the quartic relationship.

Outlook

The quartic speedup is very compelling, as beyond-quadratic speedups are rare in quantum algorithms. It is also appealing that the speedup also does not necessarily rely upon an assumption of large-scale QRAM. However, it is currently unclear how applicable this quartic speedup can be in real-world situations.

Further reading

We refer the reader to [904] for a discussion on the intuition and scope of the quartic speedup and additional technical details. We also note that [1091] has studied the quantum approximate optimization algorithm (QAOA) applied to the spiked tensor model, although the result is not directly comparable as it is only shown to succeed for larger values of λ, where additional classical algorithms are also successful.

9.4 Topological data analysis

Overview

In topological data analysis (TDA), we aim to compute the dominant topological features (connected components and k-dimensional holes), known as Betti numbers, of N data points sampled from an underlying topological manifold or of a graph with N vertices. These features may be of independent interest (e.g., the number of connected components in the matter distribution in the universe) or can be used as generic features to compare datasets. We refer to [519] for a recent survey of applications of TDA.

Quantum algorithms for TDA leverage the ability of a register of qubits to efficiently represent a quantum state that stores all cliques in the clique complex built on the topological manifold. The textbook classical algorithm exactly computes the Betti numbers, but its complexity scales polynomially with the number of cliques in the complex, which may grow combinatorially with

(N, k). In contrast, quantum algorithms naturally estimate Betti numbers normalized by the number of cliques in the complex, and their complexity scales polynomially in (N, k) for clique-dense complexes. If the error is rescaled so as to be a constant additive error estimate for the Betti number, currently known quantum algorithms provide a quadratic speedup over the best classical algorithms for the equivalent problem. For relative-error estimates of the Betti number, quartic and superpolynomial speedups have been shown for certain families of graphs. In addition, a number of complexity-theoretic results have been shown that provide evidence that estimating normalized Betti numbers is efficient for quantum computers, but classically hard. Nevertheless, finding practical applications for relative error estimates of high-dimensional Betti numbers is an open research question.

Actual end-to-end problem(s) solved

We construct a simplicial complex built from N data points sampled from an underlying manifold. The simplicial complex is a higher-dimensional generalization of a graph, constructed by connecting data points within a given distance of each other. A simplicial complex constructed in this way is known as a clique complex or a Vietoris–Rips complex. We can consider a sequence, known as a filtration, of complexes constructed by connecting points at increasingly large distances (the distance is referred to as the length scale of the clique complex). We denote the number of k-simplices in the complex at length scale i as $|S^i_k|$.

We then compute the Betti numbers β^i_k (the number of k-dimensional holes at a given length scale i) or the persistent Betti numbers $\beta^{i,j}_k$ (the number of k-dimensional holes present at both scale i and scale j) of the simplicial complex. The persistent Betti numbers $\beta^{i,j}_k$ are used to infer the dominant topological features, considered to be those with the longest persistence as the length scale is increased. The births and deaths of features are typically plotted on a "persistence diagram." Different datasets can be compared by using stable distance measures between their diagrams, or by vectorizing the diagrams and using kernel methods or neural networks. For graphs, there is only a single length scale i, and so β^i_k is the quantity of interest. For statements common to both $\beta^{i,j}_k$ and β^i_k, we will use the notation β^*_k. Typical classical applications consider low values of k, motivated primarily by computational cost and interpretability of the resulting topological features.

Dominant resource cost/complexity

Quantum algorithms naturally estimate $\beta^*_k/|S^i_k|$ to additive error ϵ [709, 469, 514, 755, 143]. For a complex built from N data points, we can either use N

qubits to encode the simplicial complex, or $O(k\log(N))$ qubits when $k \ll N$. Quantum algorithms have two subroutines:

(i) Preparing a state that encodes the simplices present in the simplicial complex. This reduces to finding k-simplices present in the complex at the given length scale (using either classical rejection sampling or Grover's algorithm / amplitude amplification). Using Grover's algorithm this scales as $\left(\binom{N}{k+1}/|S_k^i|\right)^{1/2}$. More efficient clique-finding methods can be used for special classes of graphs [474].

(ii) Projecting onto the eigenspace of an operator that encodes the topological features of the complex in the amplitude of the quantum state (using either quantum phase estimation or quantum singular value transformation). This introduces a dependence on the gap(s) Λ of the operator(s) used to encode the topology.

The most efficient approaches use amplitude estimation to compute the normalized (persistent) Betti number. The most expensive subroutines within the quantum algorithms are the membership oracles that determine if a given simplex is present in the complex, the cost of which we denote by m_k. In the classically challenging clique-dense regime, the overall cost of the most efficient known quantum algorithms for computing $\beta_k^*/|S_k^i|$ to error ϵ is approximately

$$\widetilde{O}\left(\frac{m_k}{\epsilon}\sqrt{\frac{\beta_k^*}{|S_k^i|}}\left(\sqrt{\frac{\binom{N}{k+1}}{|S_k^i|}}+\frac{\mathrm{poly}(N,k)}{\Lambda}\right)\right).$$

When we choose $\epsilon = \Delta/|S_k^i|$ (i.e., computing β_k^* to constant additive error Δ) the complexity is

$$\widetilde{O}\left(\frac{m_k\sqrt{\beta_k^*}}{\Delta}\left(\sqrt{\binom{N}{k+1}}+\frac{\sqrt{|S_k^i|}\cdot\mathrm{poly}(N,k)}{\Lambda}\right)\right).$$

It is clear that regardless of how $|S_k^i|$ and β_k^* scale, the runtime is polynomial in $\binom{N}{k+1}$.

When we choose $\epsilon = r\beta_k^*/|S_k^i|$ (i.e., computing β_k^* to relative error r) the complexity is

$$\widetilde{O}\left(\frac{m_k}{r}\left(\sqrt{\frac{\binom{N}{k+1}}{\beta_k^*}}+\sqrt{\frac{|S_k^i|}{\beta_k^*}}\cdot\frac{\mathrm{poly}(N,k)}{\Lambda}\right)\right).$$

For clique-dense complexes with large Betti numbers, where β_k^* and $|S_k^i|$ are only polynomially smaller than $\binom{N}{k+1}$, the runtime is polynomial in N and k.

Existing resource estimates

In [755], the gate depth (and non-Clifford gate depth) of all subroutines (including explicit implementations of the membership oracles) was established for computing $\beta_k^{i,j}$ and β_k^i. However, that reference did not consider a final compilation to T/Toffoli gates for concrete problems of interest.

In [143], the Toffoli complexity of estimating β_k^i was determined. The Toffoli complexity for estimating β_k^i to relative error for a family of graphs with large β_k^i was determined for $k = 4, 8, 16, 32$ and $N \leq 10^4$. The resulting Toffoli counts ranged from 10^8 ($N = 100, k = 4$) to 10^{17} ($N = 10^4, k = 32$), using $O(N)$ logical qubits.

Caveats

Quantum algorithms naturally solve a different problem than the textbook classical algorithm solves. Namely, the quantum algorithm estimates $\beta_k^*/|S_k^i|$ to error ϵ, with runtime poly(ϵ^{-1}), for a single length scale (pair of length scales). The algorithm must be repeated for all length scales to compute the persistence diagram. In contrast, the textbook classical algorithm for dimension k and length scale j exactly computes the full persistence diagram for all $\beta_{k' \leq k}^{i,j' \leq j}$.

Quantum algorithms (and classical algorithms based on the power method discussed below) depend on the eigenvalue gap(s) Λ of the operator(s) that encode the topology. The scaling of these gaps has not been studied for typical applications.

Finally, typical classical applications consider dimension $k \leq 3$, and applications of high-dimensional Betti numbers are not yet known.

Comparable classical complexity and challenging instance sizes

While classical algorithms are technically efficient for constant dimension k, they are limited in practice. For a number of benchmark calculations on systems with up to 10^9 simplices, we refer to [819].

The textbook classical algorithm, which for dimension k and length scale j exactly computes the full persistence diagram for all $\beta_{k' \leq k}^{i,j' \leq j}$ (rather than a single Betti number), has a worst-case scaling of $O(|S_{k,k+1}^j|^\omega)$ where $\omega \leq 2.37$ is the exponent of matrix multiplication, and we have defined a shorthand notation $|S_{k,k+1}^j| := |S_k^j| + |S_{k+1}^j|$ [764]. In practice, the textbook classical algorithm is observed to scale as $O(|S_{k,k+1}^j|)$ due to sparsity in the complex [764]. Moreover, for problems that do not naturally have this sparse structure, well-studied classical heuristics can be applied to sparsify the complex [765].

Classical algorithms based on the power method [403] can achieve worst-case scaling of $O(|S_k^i|)$ for computing individual Betti numbers β_k^i (an improve-

ment over the worst-case scaling of the textbook algorithm above). The classical power method scales approximately as

$$\widetilde{O}\left(\frac{|S_k^i|(Nk\beta_k^i + (\beta_k^i)^2)\lambda_{\max}}{\Lambda}\log\left(\frac{1}{\epsilon}\right)\right)$$

to compute β_k^i to additive error ϵ, where $\lambda_{\max}$ is a bound on the largest eigenvalue of the operator encoding the topology. The power method has recently been extended to compute persistent Betti numbers, with a similar complexity [755]. Although the power method for persistent Betti numbers is more efficient than the worst-case performance of the textbook classical algorithm described above, it must be repeated for each pair of length scales to compute the persistence diagram, which is a disadvantage in practice.

Recently, randomized classical algorithms have been proposed for estimating $\beta_k^i/|S_k^i|$ to additive error [143, 53]. The algorithm of [53] scales as

$$\left(\frac{N}{\lambda_{\max}}\right)^{O\left(\frac{1}{\sqrt{\Lambda}}\log\left(\frac{1}{\epsilon}\right)\right)} \cdot \mathrm{poly}(n)$$

assuming that we can efficiently sample and check k-simplices. When $k = \Omega(N)$, the algorithm runs in polynomial time for clique complexes with constant gap Λ and error $\epsilon = \Omega(1/\mathrm{poly}(N))$ (or ϵ constant and $\Lambda = \Omega(1/\log^2(N))$).

Speedup

As discussed above, quantum algorithms naturally compute $\beta_k^*/|S_k^i|$ to additive error ϵ, with runtime $\mathrm{poly}(\epsilon^{-1})$. A number of complexity-theoretic results have been shown for this problem. Reference [474] showed that estimating the kernel dimension of general Hamiltonians is DQC1-hard.[12] In [214], it was shown that estimating normalized quasi-Betti numbers (which accounts for miscounting low-lying but nonzero singular values) of general cohomology groups is also DQC1-hard. The hardness of estimating normalized (persistent) Betti numbers of a clique complex, subject to a gap assumption of $\Lambda = \Omega(1/\mathrm{poly}(N))$—which is the problem solved by existing quantum algorithms—has not been established (see [214, Section 1.1]).

Reference [903] showed that determining if the Betti number of a (clique-dense) clique complex is nonzero is NP-hard in general. This was superseded by the results of [319, 620] which showed that this problem is QMA_1-hard.

We can consider the speedup of quantum algorithms for TDA in two regimes, constant additive error and relative error:

[12] DQC1 is a complexity class that is physically motivated by the "one clean qubit model" [635]. This model has a single pure state qubit which can be initialized, manipulated, and measured freely, as well as $N - 1$ maximally mixed qubits.

- For constant additive error, the most natural comparison is between the quantum algorithm and classical algorithms based on the power method. Quantum algorithms are able to achieve a quadratic speedup over the classical power method in the clique-dense regime [403, 755]. An exponential speedup is not possible for general graphs, due to the aforementioned NP- and QMA-hardness results [903, 319, 620].
- For the task of computing β_k^i to relative error, graph families have been identified for which the quantum algorithm provides superpolynomial [143] or quartic [143, 903] speedups over the classical power method. Recently introduced randomized classical algorithms [143, 53] may scale efficiently for this same task of estimating β_k^i to relative error. For example, when $k = \Omega(N)$ the algorithm of [53] runs in polynomial time for clique complexes with constant gap Λ and error $\epsilon = \Omega(1/\text{poly}(N))$ or ϵ constant and $\Lambda = \Omega(1/\log^2(N))$. These are more restrictive conditions than quantum algorithms (which can simultaneously have both $\Lambda, \epsilon = \Omega(1/\text{poly}(N))$). These features will not occur for all graphs.

NISQ implementations

In [13], a NISQ-amenable compilation of the quantum algorithm described above was proposed, trading deep quantum circuits for many repetitions of shallower circuits, which comes at the cost of worsening the asymptotic scaling of the algorithm (see the table in [755] for a quantitative comparison). A proof-of-principle experiment was performed demonstrating this method [13]. In [214], it was shown that the TDA problem can be mapped to a fermionic Hamiltonian, and it was proposed to use the variational quantum eigensolver to find the ground states of this Hamiltonian (the degeneracy of which gives β_k^i). It is unclear what ansatz circuits one should use to make this approach advantageous compared to classical algorithms, as naive (e.g., random) trial states would have exponentially small overlap with the target states.

Outlook

The complexity-theoretic results that provide evidence for the classical hardness and quantum tractability of estimating normalized (persistent) Betti numbers suggest that quantum algorithms for TDA may be an interesting area to search for new quantum speedups.

Nevertheless, it is important to emphasize that current quantum algorithms do not provide more than quadratic speedups for the practical problem solved in current TDA applications, and complexity-theoretic results suggest that exponential speedups will not be possible for general graphs for this task.

As such, an important open problem is to identify applications for the task naturally solved by quantum computers (providing relative error estimates for clique-dense graphs with large Betti numbers). If new applications can be identified for datasets that are both clique-dense and have large high-dimensional (persistent) Betti numbers (that are practically interesting to compute to relative error), then quantum algorithms may be of practical relevance.

9.5 Quantum neural networks and quantum kernel methods

Overview

In this section, we discuss two collections of proposals to use a quantum computer as a platform to execute machine learning models, often known as *quantum neural networks* and *quantum kernel methods*. Some early ideas in this space were motivated by the constraints of near-term, "NISQ" [843] devices. Despite this, not all subsequent proposals are necessarily implementable on NISQ devices. Moreover, the proposals need not be restricted to running on NISQ devices, but could also be run on devices with explicit quantum error correction. For simplicity, we present concrete examples based on supervised machine learning tasks. However, outside of these examples, we keep our discussion more general and note that the techniques are also applicable to other settings, such as unsupervised learning and generative modeling.

Given access to some data, our goal is to obtain a function or distribution that emulates certain properties of the data, which we will call a *hypothesis*. This is obtained by first defining a *hypothesis set* or *model family*, and using a learning algorithm to output a hypothesis from this set. For example, in supervised learning, we have data $x_i \in X$ that have respective labels $y_i \in Y$. The goal is then to find a hypothesis function $h : X \to Y$ that approximates the "true" unknown underlying labeling function, such that it correctly labels previously unseen data with high probability. Note that we have left the exact descriptions of the sets X and Y ambiguous. They could, for instance, correspond to sets of numbers or vectors. More generally, this description encompasses the possibility of operating on quantum data such that each x_i or y_i corresponds to a quantum state.

Quantum neural networks and quantum kernel methods use a quantum computer to assist in constructing the model family, in place of a classical model such as a neural network. Specifically, here we prepare some quantum state(s) encoding the data and measure some observable(s) to ultimately construct model predictions. We first elaborate on both quantum neural networks and quantum kernel methods.

Quantum neural networks

Actual end-to-end problem(s) solved: Given data x, we consider a model constructed from a parameterized quantum circuit:

$$h_{\boldsymbol{\theta}}(x) = f(\mathrm{tr}[\rho(x,\boldsymbol{\theta})O]), \tag{9.13}$$

where $\rho(x,\boldsymbol{\theta})$ is a quantum state (output of some parameterized quantum circuit) that encodes both the data x as well as a set of adjustable parameters $\boldsymbol{\theta}$, O is some chosen measurement observable, and f is some function that can be enacted as classical postprocessing on the measurement result (we remark that O itself can also be trainable [475], but we do not explicitly indicate this in the notation for simplicity of exposition). As a basic example, if x corresponds to a classical vector, $\rho(x,\boldsymbol{\theta})$ could correspond to initializing in the $|0\rangle\langle 0|$ state and applying some data-encoding gates $U(x)$ followed by parameterized gates $V(\boldsymbol{\theta})$. Alternatively, the data itself could be a quantum state, and a more general operation in the form of a parameterized channel $\mathcal{V}(\boldsymbol{\theta})$ could be applied. There is also no a priori reason why data encoding and trainable gates need to be applied each once in separate steps rather than in a mixed or repeated fashion.

The model is optimized via a learning algorithm which aims to find the optimal parameters $\boldsymbol{\theta}^*$ by minimizing a loss function. For instance, in supervised learning, given some labeled training dataset $T = \{(x_i, y_i)\}$, a suitable choice of loss should compare how close each $h_{\boldsymbol{\theta}}(x_i)$ is to the true label y_i for all data in T. The quality of the model can then be assessed on a set of previously unseen data outside of T. It is important to pause here and reflect that an optimized loss does not guarantee good performance on unseen data. This is referred to in the literature as the gap between *empirical* and *total* risk, or simply the generalization gap/error. Conversely, a small generalization error alone is not sufficient to guarantee good performance (one should then also ask for a good loss on training data).

We remark that the setting we presented has substantial overlap with the setting of variational quantum algorithms (VQAs)—indeed, a quantum neural network can be thought of as a VQA that incorporates data—thus, the same challenges and considerations that apply to VQAs also apply here. There will additionally be extra considerations due to the role of the data.

Dominant resource cost/complexity: The encoding of data x and parameters $\boldsymbol{\theta}$ in Eq. (9.13) should be sufficiently expressive that it (i) leads to good performance on data and (ii) is (at minimum) not efficiently simulable classically [243], if one is to seek quantum advantage. These requirements set some criteria for minimum circuit complexity.

The learning algorithm to find optimal parameters is usually performed by classical heuristics, such as gradient descent, and can have significant time overhead, requiring evaluation of Eq. (9.13) at many parameter values (see Chapter 20 on VQAs for more details).

The size of the training dataset required can also have direct implications for runtime, with a larger amount of training data typically taking a longer time to process. Reference [235] proves that good generalization can be achieved with the size of the training data $|T|$ growing in tandem with the number of adjustable parameters M. Specifically, it is shown that the generalization error with high probability scales as $O(\sqrt{M \log(M)/|T|})$. Thus, only a mild amount of data is required for good generalization. We stress again that this alone does not say anything about the ability for quantum neural networks to obtain low training error.

Scope for advantage: Quantum neural networks could achieve advantage in a number of ways, for example, by improving on runtime or by using less training data. In supervised learning settings, generalization performance is a separate consideration and an additional domain for possible advantage. Machine learning with quantum neural networks has yielded some promising performance empirically and encouraging theoretical guarantees exist for certain stages of the full pipeline in restricted settings [901, 234, 235, 700, 1070] (loss minimization can remain a challenge [663, 243], see Chapter 20 on VQAs again for more details). Nevertheless, there are currently no practical use cases with full end-to-end performance guarantees in the same way that we have for other quantum algorithms. However, due to the heuristic nature of classical machine learning, one may debate whether such a guarantee is possible, or even if seeking theoretical quantum advantage in the traditional algorithmic sense is the most appropriate goal [915].

Quantum kernel methods

Actual end-to-end problem(s) solved: Quantum kernel methods are a quantum instance of a class of techniques known as kernel methods, of which support vector machines are a prominent example. We first briefly review the general framework. Given a dataset $T = \{x_i\} \subset X$, the model can be written as

$$h_{\boldsymbol{\alpha}}(x) = \sum_{i\colon x_i \in T} \alpha_i k(x, x_i)\,, \tag{9.14}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots)$ is a vector of parameters to be optimized, and $k(x, x')\colon X \times X \to \mathbb{R}$ is a measure of similarity known as the kernel function. This model has several theoretical motivations:

- The matrix with entries $K_{ij} = k(x_i, x_j)$ is usually defined to be symmetric positive semidefinite for any choice of $\{x_1, \ldots, x_m\} \subseteq X$ and $k(x_i, x_j)$. By Mercer's theorem, it is thus an inner product of feature vectors $\phi(x_i), \phi(x_j)$ which embed the data x_i and x_j in a (potentially high-dimensional) Hilbert space. Linear statistical methods can be used to learn a linear function in this high-dimensional space, only using the information of the inner products $k(x_i, x_j)$ and never having to explicitly evaluate $\phi(x_i)$ and $\phi(x_j)$, which can be much harder to compute.
- The Representer Theorem [905] states that the optimal model over the dataset T (optimal for T, though not necessarily for expanded datasets) can be expressed as a linear combination of kernel values evaluated over T—that is, the optimal model exactly takes the form in Eq. (9.14). This is known as the kernel trick.
- Further, if the loss function is convex, then the dual optimization program to find the optimal parameters $\boldsymbol{\alpha}^*$ is also convex [913].

A key question that remains is then how to choose a kernel function. Quantum kernel methods embed data in quantum states, and thus evaluate $k(x_i, x_j)$ on a quantum computer. Similar to quantum neural networks or any other quantum model, the quantum kernel should be hard to simulate classically [243]. As an example, we present two common choices of quantum kernel (see [425] for a more general discussion).

- The fidelity quantum kernel

$$k_F(x, x') = \mathrm{tr}[\rho(x)\rho(x')]\,, \tag{9.15}$$

  which can be evaluated either with a SWAP test or, given classical data with unitary embeddings, it can be evaluated with the overlap circuit $|\langle 0|U(x')^\dagger U(x)|0\rangle|^2$.
- The fidelity kernel can run into issues for high-dimensional systems (increasing qubit count), as the inner product in Eq. (9.15) can be very small for $x \neq x'$. This motivated the proposal of a family of projected quantum kernels [542], of which one example is the Gaussian projected quantum kernel

$$k_P(x, x') = \exp\left(-\gamma \sum_{\ell=1}^{n} \left\|\rho_\ell(x) - \rho_\ell(x')\right\|_2^2\right), \tag{9.16}$$

  where $\rho_\ell(x)$ is the reduced density matrix of the n-qubit state $\rho(x)$ on qubit ℓ, and γ is a hyperparameter.

Dominant resource cost/complexity: During the optimization of the dual program to find the optimal parameters α^*, $O(|T|^2)$ expectation values corresponding to the kernel values in Eq. (9.14) need to be accurately evaluated, as well as when computing $h_{\alpha^*}(x)$ for a new data point x with the optimized model. This can lead to a significant overhead in applications with large datasets. Alternatively, the primal optimization problem has reduced complexity in the dataset size, but greatly exacerbated dependence on the error [415]. The gate complexity is wholly dependent on the choice of data encoding leading to the kernel function. As the kernel function should be classically nonsimulable, this sets some minimum requirements in terms of circuit complexity. However, in the absence of standardized techniques for data encoding, it is hard to make more precise statements.

Scope for advantage: In [704], the authors demonstrate that using a particular constructed dataset and data embedding, concrete quantum advantage can be obtained for a constructed machine learning problem based on the discrete logarithm problem. The original work was based on the fidelity kernel, but a similar advantage can also be more simply obtained for the projected quantum kernel [542] and adapted beyond kernel methods to the reinforcement learning setting [573]. Beyond this, concrete advantage (up to similar computational assumptions) can be shown more generally for any learning problem where the underlying (unknown, to be learned) labeling function constitutes a BQP-hard family [473]. While great strides have been made in understanding the complexity of quantum kernel methods [88, 542], at present there do not yet exist examples of explicit end-to-end theoretical guarantees of advantage for classical data relevant for a real-world problem. As with quantum neural networks, it may be debated whether or not this is a reasonable question for theoretical research efforts.

Caveats

One consideration we have not discussed so far is how to encode classical data into a quantum circuit, which is a significant aspect of constructing the quantum model. There are many possibilities, such as amplitude encoding or encoding data into rotation angles of single-qubit rotations (see, e.g., [711, 513, 547, 664]). While certain strategies are popular, there is no universal strategy. In general, it is unclear what is the best choice for a given problem at hand, and thus selecting the data-encoding strategy can itself be a heuristic process. The same question extends to the choice of quantum neural network or quantum kernel. While certain choices may perform well in specific problem instances,

there is at present a lack of strong evidence why such approaches may be advantageous over their classical counterparts in general.

While optimization of parameterized quantum circuits is predominantly a concern for quantum neural networks, the search for good quantum kernels has also motivated proposals of trainable kernels [547, 414, 436] where a parameterized quantum circuit is used to construct the quantum kernel (note that this is distinct from the "classical" optimization of α in Eq. (9.14)). In the case that the parameter optimization process is performed using heuristics, it is subject to the same challenges and considerations that arise with VQAs (see Chapter 20 for more details).

Finite statistics is an important consideration for both settings. Where there is optimization of parameterized quantum circuits, one must take care to avoid the barren plateau phenomenon [663] (again, see Chapter 20 for more details and further references). Analogous effects can also occur in the kernel setting [655], which can arise even purely due to the data-encoding circuit [542, 988].

Outlook

The use of classical machine learning models is generally heuristic, guided by empirical evidence or sometimes physical intuition. Despite this, classical machine learning has found remarkable success in solving many practical problems of interest. The quantum techniques outlined in this section also broadly follow this approach (although theoretical progress has also been substantial in certain areas), and there is no a priori reason why they cannot also be useful. Nevertheless, it can be challenging to make concrete predictions for quantum advantage, particularly for learning problems with classical data (see [704, 473] as some exceptions). For practical problems this is exacerbated by our limited analytic understanding for end-to-end applications, even in the fully classical setting. Indeed, it may ultimately be challenging to have the same complete end-to-end theoretical analysis that other quantum algorithms enjoy, aside from a few select examples [915]. Within the realm of quantum data, there appears to be ripe potential for concrete provable advantage [543, 265, 236], however, this is beyond the scope of this section.

Further reading

We refer the reader to [913, 547] for pedagogical expositions of quantum kernel methods, to [118, 241] for comprehensive reviews of quantum neural networks, and to [242] for a review of quantum machine learning models at large, including an exposition of machine learning with quantum data.