Quantum-inspired classical algorithms for principal component analysis and supervised clustering

Ewin Tang

November 2, 2018

Abstract

We describe classical analogues to quantum algorithms for principal component analysis and nearest-centroid clustering. Given sampling assumptions, our classical algorithms run in time polylogarithmic in input, matching the runtime of the quantum algorithms with only polynomial slowdown. These algorithms are evidence that their corresponding problems do not yield exponential quantum speedups. To build our classical algorithms, we use the same techniques as applied in our previous work dequantizing a quantum recommendation systems algorithm. Thus, we provide further evidence for the strength of classical ℓ^2-norm sampling assumptions when replacing quantum state preparation assumptions, in the machine learning domain.

1 Introduction

In our previous work on dequantizing quantum machine learning algorithms, we give the guideline [7]:

> When QML algorithms are compared to classical ML algorithms in the context of finding speedups, any state preparation assumptions in the QML model should be matched with ℓ^2-norm sampling assumptions in the classical ML model.

In this note, we follow this guideline to dequantize two quantum machine learning algorithms, quantum principal component analysis [6] and quantum supervised clustering [5]. That is, we give classical algorithms that, with sampling assumptions analogous to quantum state preparation assumptions, match the bounds and runtime of the corresponding quantum algorithms, with only polynomial slowdown. We do so using our classical toolkit originally applied to the recommendation systems problem, with the hopes of giving prototypical examples to reference when dequantizing other quantum machine learning algorithms.

The toolkit is as follows [7]:

(a) To estimate values: standard sampling for estimators (Proposition 6.2);

(b) To sample from matrix-vector products: rejection sampling (Proposition 6.4);

(c) To reduce dimensionality of a dataset: matrix-subsampling (as done in FKV in Theorem 6.9).

Our belief is that QML algorithms that are *not* BQP-complete are highly susceptible to dequantization using the approaches we outline here, with only polynomial slowdown. We, therefore, urge caution when claiming exponential speedups over classical algorithms. Nevertheless, BQP-complete problems, such as the sparse version of matrix inversion [3] and its applications, are still unlikely to be dequantized in full.

A thorough argument for the guideline can be found in Section 2 of [7]; we describe here why ℓ^2-norm sampling assumptions are reasonable. First, although sampling access is stronger than simple query access to input vectors, all sublinear time algorithms (quantum and classical) require some strong assumption on the input model, whether via state preparation, sampling, sparsity, or streaming. In practice, many settings can implement sampling assumptions. If our vectors are dynamically updated, such as when they represent data that we learn over time, sampling can be implemented with a dynamic data structure with low overhead, as is done for recommendation systems [4]. This amortizes the typical cost to read in a vector to produce a sample. This data structure can also be implemented with one pass through the data, allowing for sampling in streaming models. Further, when input vectors are close to uniform, such samples can be performed quickly using rejection sampling[1]. Such assumptions are standard and have applications throughout data analysis [2]. Finally, classical sampling assumptions are certainly easier to satisfy in practice than quantum state preparation, at least for the moment.

Throughout, we use the following notation. $[n] := \{1, \ldots, n\}$. For a matrix A, A_i and $A^{(i)}$ will refer to the ith row and column, respectively. $\|A\|_F$, $\|A\|_2$, and $\|A\|_{tr}$ will refer to Frobenius, spectral, and trace norm, respectively. Norm of a vector v, denoted $\|v\|$, will always refer to ℓ^2-norm.

For a nonzero vector $x \in \mathbb{R}^n$, a sample from x refers to sampling an index $i \in [n]$ with probability $x_i^2/\|x\|^2$ In all situations, sampling access will be present in addition to query access, and accordingly, we will conflate samples i with the corresponding entries x_i.

Further, for a matrix $A \in \mathbb{R}^{m \times n}$, we define *sampling and query access* to A as having sampling and query access to A_i and $\tilde{A} = \{\|A_i\|\}_{i \in [m]}$ the vector of row norms, and knowledge of $\|A\|_F^2$. Alternatively, we could replace sampling to rows and row norms with column and column norms. This is precisely the type of sampling access supported by the recommendation system data structure [4]; similar assumptions occur frequently in the QML literature.

2 Principal Component Analysis

While Lloyd, Mohseni, and Rebentrost describe a more general strategy for Hamlitonian simulation of density matrices in their paper on QPCA, they propose one immediate application: producing quantum states corresponding to the top principal components of a low-rank

[1]This assumption on input vectors also implies fast quantum state preparation.

dataset [6]. We give a classical algorithm for this task by applying FKV as described in [7], giving oracles to the singular vectors and values. Then, by treating this sampling and query access analogously to the quantum states, we can perform basic PCA routines.

Suppose we are given a matrix $A \in \mathbb{R}^{n \times d}$ whose rows correspond to data in a dataset. We will find the principal eigenvectors and eigenvalues of $A^T A$; when A is a mean zero dataset, this corresponds to the top principal components.

To do so, assume we have sampling and query access to A. These are the analogous classical assumptions for QPCA. Without loss of generality, $\|A\|_F = 1$, since QPCA assumes the equivalent statement that $\|A^T A\|_{tr} = 1$.

Theorem 2.1. *Given $A \in \mathbb{R}^{n \times d}$ such that $\|A\|_F = 1$ with $O(1)$ time sampling and query access, we can query and sample from the approximate singular vectors of A in $\text{poly}(\frac{1}{\varepsilon})$ time.*

We will leave the analysis to a future version of this note. The singular values are thresholded at $\sqrt{\varepsilon}$ (or equivalently, the eigenvalues at ε), and thus the number of singular vectors is bounded by ε. When the top singular values are "well-separated" (that is, $\sigma_i - \sigma_{i+1} = \Omega(\kappa)$ for $\kappa = \text{poly}(\varepsilon)$ and σ_i above the threshold), these approximate principal components are close to the true principal components, but otherwise, the subspace they define is close to the subspace defined by the true top principal components.

Algorithm 1: Principal Component Analysis

Input: Matrix $A \in \mathbb{R}^{n \times d}$ supporting the sampling operations, user $i \in [m]$, thresholds ε, δ

Run ModFKV (Algorithm 3 [7]) with $(\sigma, \varepsilon, \kappa)$ parameters as $(\sqrt{\varepsilon}, \varepsilon^2, \kappa)$ to get a description of approximate singular vectors $\hat{V} \in \mathbb{R}^{n \times k}$;

Simulate $\hat{V}$ from its description as described in Proposition 6.14 [7];

From there, we can perform two types of operations:

Dimensionality Reduction. Using Proposition 6.2, we can estimate $\hat{V}x$ for any input vector x, *only given query access to x*. The error can be as large as ε, but can be reduced to $\varepsilon\|A\|_2$ under certain niceness assumptions, as done for recommendation systems [7].

Projection Sampling. Using Lemma 6.8 combined with the above, we can produce a sample from the $\hat{V}^T\hat{V}x$, approximately a projector onto the image of $\hat{V}^T$, the top principal components. This is precisely what we do in [7].

3 Supervised Clustering

Lloyd, Mohseni, and Rebentrost's paper on supervised machine learning describes a quantum algorithm for clustering, with an algorithm estimating distance to the centroid of a cluster. We give a corresponding classical algorithm using straightforward sampling, which can be

extended to more general cases via rejection sampling. A classical algorithm similar to this was also considered by Aaronson [1], in the cases where input vectors are relatively uniform.

We can combine this algorithm with principal component analysis to provide a complete supervised learning system that takes time polylogarithmic in input. By the dimensionality reduction of PCA, we need only consider our vectors to be in a space of low dimension. If we can label a small amount of vectors to accurately determine their corresponding clusters, we can use the algorithm described below to continue assigning new data to clusters (provided that clustering based on nearest centroid is a good heuristic to follow).

Theorem 3.1. *Suppose we have $O(1)$-time sampling and query access to $V \in \mathbb{R}^{d\times n}$ (with respect to columns, not rows). Given some $u \in \mathbb{R}^d$ with sampling access and knowledge of $\|u\|$, we can estimate the distance to the centroid of the cluster $\|u - \frac{1}{n}V\vec{1}\|^2$ in query and time complexity independent of m and n.*

We will leave the complete analysis to a future version of this note. Let $\bar{u}$ and $\bar{V}$ be the column-normalized versions of u and V, respectively. Then let

$$M := \begin{bmatrix} \bar{u} & \frac{1}{\sqrt{n}}\bar{V} \end{bmatrix} \qquad w := \begin{bmatrix} \|u\| & -\frac{1}{\sqrt{n}}\tilde{V} \end{bmatrix}$$

These just correspond to a normalized matrix and its corresponding norms. Notice that $Mw = u - \frac{1}{n}V\vec{1}$, as desired.

Since w is not an input vector, we need to construct sampling access and find its norm. Given knowledge of $\|\tilde{V}\|^2 = \|V\|_F^2$, these are straightforward: the squared norm is $\|\tilde{V}\|^2 + \|u\|^2$, and one can sample from w by picking $\|u\|$ with the requisite probability, and sampling from $\tilde{V}$ otherwise.

We want to estimate $w^T M^T M w$. Suppose we have sampling access to w and the columns of M. Then consider sampling indices i, j from w and k from the ith column of M, and let our estimator be $X = M_{kj}\|M_i\|^2/w_i w_j M_{ki}$.

$$\mathrm{E}[X] = \sum_{i,j,k} \frac{w_i^2 w_j^2 M_{ki}^2}{\|w\|^4\|M_i\|^2} X_{i,j,k} = \sum_{i,j,k} \frac{w_i M_{ik}^T M_{kj} w_j}{\|w\|^4} = \frac{w^T M^T M w}{\|w\|^4}$$

$$\mathrm{Var}[X] \le \sum_{i,j,k} \frac{w_i^2 w_j^2 M_{ki}^2}{\|w\|^4\|M_i\|^2} X_{i,j,k}^2 = \sum_{i,j,k} \frac{M_{kj}^2\|M_i\|^2}{\|w\|^4} = \frac{\|M\|_F^4}{\|w\|^4} = \frac{16}{\|w\|^4}$$

Thus, by taking the median of means, we can produce an estimator for $w^T M^T M w$ with $O(\|w\|^4 \log(1/\delta)/\varepsilon^2)$ queries. This matches the quantum algorithm, which also has polynomial dependence on $\|w\|^2$.

Acknowledgements

Thanks to Ronald de Wolf for giving the initial idea to look at QPCA. Thanks to Daniel Liang and Patrick Rall for their help fleshing out these ideas and reviewing a draft of this document. Thanks to Scott Aaronson for helpful discussions.

References

[1] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

[2] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[3] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[4] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Innovations in Theoretical Computer Science*, 2017.

[5] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[6] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014.

[7] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *arXiv preprint arXiv:1807.04271*, 2018.