Exploring Quantum Machine Learning Algorithms

Harshil Jain jain.harshil@iitgn.ac.in

Anirban Dasgupta anirbandg@iitgn.ac.in

Computer Science and Engineering
IIT Gandhinagar
Gujarat, India

Abstract

Machine learning, as a collection of powerful data analytical methods, is widely used in classification, face recognition, nature language processing, etc. However, the efficiency of machine learning algorithms is seriously challenged by big data. Fortunately, it is found that quantum setting for these algorithms can help overcome this problem. The pace of development in quantum computing mirrors the rapid advances made in machine learning and artificial intelligence. It is natural to ask whether quantum technologies could boost learning algorithms: this field of inquiry is called quantum-enhanced machine learning. In this report, I will describe the quantum recommendation system algorithm, followed by the classical version of it and finally I will describe quantum generative adversarial networks (QGAN) and the results of QGAN on the MNIST dataset.

Keywords: Quantum ML, Quantum Recommendation Systems, Quantum GAN

1. Introduction

Quantum machine learning is at the crossroads of two of the most exciting current areas of research: quantum computing and classical machine learning. It explores the interaction between quantum computing and machine Learning, investigating how results and techniques from one field can be used to solve the problems of the other. With an ever-growing amount of data, current machine learning systems are rapidly approaching the limits of classical computational models. In this sense, quantum computational power can offer advantage in such machine learning tasks. The field of quantum machine learning explores how to devise and implement quantum software that could enable machine learning that is faster than that of classical computers. Fuelled by increasing computing power and algorithmic advances, machine learning techniques have become powerful tools for finding patterns in data.

2. Literature Review

2.1 Quantum Recommendation System

A recommendation system uses information about past purchases or ratings of products by a group of users in order to provide personalized recommendations to individual users. More precisely, we assume there are m users, for example clients of an online platform like

Amazon or Netflix, each of whom have some inherent preference or utility about n products, for example books, movies etc. The user preferences are modeled by an m × n matrix P, where the element P_ij denotes how much the user i values product j. If the preference matrix P had been known in advance, it would have been easy to make good recommendations to the users by selecting elements of this matrix with high value. However, this matrix is not known apriori. Information about P arrives in an online manner each time a user buys a product, writes a review, or fills out a survey. A recommendation system tries to utilize the already known information about all users in order to suggest products to individual users that have high utility for them and can eventually lead to a purchase.

Kerenidis and Prakash (2016) present a quantum algorithm for recommendation systems that has running time O(poly(k)polylog(mn)). All known classical algorithms for recommendation systems that work through reconstructing an approximation of the preference matrix run in time polynomial in the matrix dimension. Our algorithm provides good recommendations by sampling efficiently from an approximation of the preference matrix, without reconstructing the entire matrix. For this, we design an efficient quantum procedure to project a given vector onto the row space of a given matrix. This is the first algorithm for recommendation systems that runs in time poly-logarithmic in the dimensions of the matrix and provides an example of a quantum machine learning algorithm for a real world application.

2.1.1 Matrix Sampling

In general, the input to the reconstruction algorithm is a subsample of some matrix A. There are quite a few different ways of subsampling a matrix, for example, sampling each element of the matrix with some probability or sampling rows and/or columns of the matrix according to some distribution. Each element of the matrix A that has size m× n is sampled with probability p and rescaled so as to obtain the random matrix $\hat{A}$ where each element is equal to $\hat{A}_{ij} = Aij/p$ with probability p and 0 otherwise. The reconstruction algorithm computes the projection of the input matrix $\hat{A}$ onto its k-top singular vectors; we denote the projection by $\hat{A}_k$. The analysis of the algorithm shows that the approximation error $||A - \hat{A}_k||$ is not much bigger than $||A - A_k||$. Projecting onto the top k singular vectors of the subsampled matrix $\hat{A}$ thus suffices to reconstruct a matrix approximating A.

2.1.2 The data structure

Each row of a matrix can be viewed as a vector in Rn and the 2n values are stored as a full binary tree of n leaves. The leaves hold the individual amplitudes of the vector and each internal node holds the sum of the squares of the amplitudes of the leaves rooted on this node. For each entry added to the tree, we need to update log(n) nodes in the tree as shown in Figure 1.

This can be done by having a copy of the data structure specified by for each row of A and has all of the desired properties.

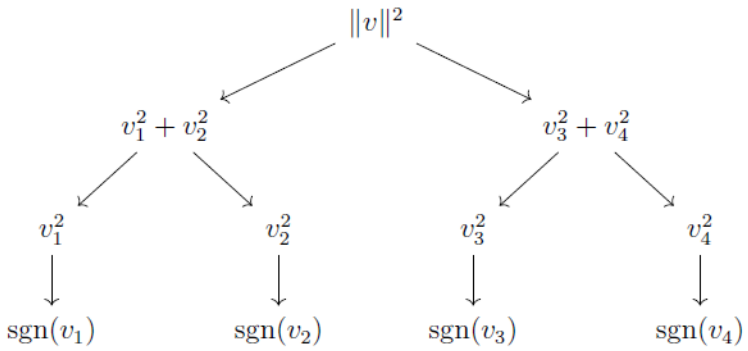

Figure 1: Data structure used

Theorem: Let A $\in R^{m\times n}$. Entries (i, j,A_{ij}) arrive in the system in an arbitrary order and w denotes the number of entries that have already arrived in the system. There exists a data structure to store the entries of A with the following properties:

i. The size of the data structure is O($w \cdot log^2(mn)$)
ii. The time to store a new entry (i, j,A_{ij}) is O($log^2(mn)$)

2.1.3 Quantum Singular Value Estimation

The second tool required for the projection algorithm is an efficient quantum algorithm for singular value estimation. In the singular value estimation problem we are given a matrix A such that the vector states corresponding to its row vectors can be prepared efficiently. P Given a state $|x\rangle = \sum_i \alpha_i |v_i\rangle$ for an arbitrary vector $x \in R^n$ the task is to estimate the singular values corresponding to each singular vector in coherent superposition. Running Time is O(polylog(mn)/ϵ where ϵ is the error margin in the singular values. The algorithm is outlined below:

1. Create $|x\rangle = \sum_i \alpha_i |v_i\rangle$.
2. Append a first register $|0^{\lceil \log m \rceil}\rangle$ and create the state $|Qx\rangle = \sum_i \alpha_i |Qv_i\rangle$
3. Perform phase estimation with precision parameter $2\epsilon > 0$ on the input $|Qx\rangle$ for the unitary $W = U \cdot V$ and obtain $\sum_i \alpha_i |Qv_i, \overline{\theta_i}\rangle$.
4. Compute $\overline{\sigma_i} = \cos(\overline{\theta_i}/2)\|A\|_F$ where $\overline{\theta_i}$ is the estimate from phase estimation, and uncompute the output of the phase estimation.
5. Apply the inverse of the transformation in step 2 to obtain $\sum_i \alpha_i |v_i\rangle |\overline{\sigma_i}\rangle$.

Figure 2: Quantum singular value estimation

2.1.4 Quantum projection with threshold

Let $A = \sum_i \sigma_i u_i v_i^t$. We recall that $A_{\geq\sigma} = \sum_{\sigma_i \geq \sigma} \sigma_i u_i v_i^t$ is the projection of the matrix A onto the space spanned by the singular vectors whose singular values are bigger than σ. Also, $A_{\geq\sigma,\kappa}$ is the projection of the matrix A onto the space spanned by the union of the singular vectors whose corresponding singular values is bigger than σ and some subset of singular vectors whose corresponding singular values are in the interval $[(1-\kappa)\sigma, \sigma)$.

A quantum algorithm is presented that given access to vector state x, a matrix A and parameters σ, κ outputs a quantum state which is the projection of x onto the subspace spanned by the union of the row singular vectors whose corresponding singular values are bigger than σ and some subset of row singular vectors whose corresponding singular values are in the interval $[(1-\kappa)\sigma, \sigma)$. Running Time is O(polylog$(mn)/\epsilon$ where ϵ is the error margin in the singular values.

2.1.5 Main Algorithm

The quantum recommendation system uses the two procedures mentioned to output a recommendation for a user in time polylogarithmic in the dimensions of the matrix and polynomial is k (low rank approximation of the preference matrix)

2.2 Quantum Inspired Classical Algorithm for Recommendation Systems

The algorithm discussed here has been proposed by (Tang, 2019). It is a classical analogue to Kerenidis and Prakash's quantum recommendation system, previously believed to be one of the strongest candidates for provably exponential speedups in quantum machine learning. Our main result is an algorithm that, given an m $\times$ n matrix in a data structure supporting certain l^2-norm sampling operations, outputs an l^2-norm sample from a rank-k approximation of that matrix in time O(poly(k) log(mn)), only polynomially slower than the quantum algorithm. As a consequence, Kerenidis and Prakash's algorithm does not in fact give an exponential speedup over classical algorithms. Further, under strong input assumptions, the classical recommendation system resulting from our algorithm produces recommendations exponentially faster than previous classical systems, which run in time linear in m and n.

2.2.1 Data Structure and Matrix Sampling

The data structure and matrix sampling methods remain the same as in the Kerenidis and Prakash Quantum Recommendation Algorithm.

2.2.2 Rejection Sampling

Given sampling access to a distribution P, rejection sampling allows for sampling from a "close" distribution Q, provided we can compute some information about their corresponding distributions.

Algorithm 1: Rejection Sampling

Pull a sample s from P;
Compute $r_s = \frac{Q(s)}{MP(s)}$ for some constant M;
Output s with probability r_s and restart otherwise;

Figure 3: Rejection Sampling

If $r_i \leq 1$ for all i, then the above procedure is well-defined and outputs a sample from Q in M iterations in expectation.

2.2.3 MODFKV Algorithm

MODFKV algorithm subsamples the input matrix, computes the subsample's large singular vectors and values and outputs them with the promise that they give a good description of the singular vectors of the full matrix. It is outlined below.

Algorithm 2: ModFKV

Input: Matrix $A \in \mathbb{R}^{m \times n}$ supporting operations in Proposition 3.2, threshold σ, error parameters ε, η
Output: A description of an output matrix D
Set $K = \|A\|_F^2/\sigma^2$ and $\bar{\varepsilon} = \eta\varepsilon^2$;
Set $q = \Theta\left(\frac{K^4}{\bar{\varepsilon}^2}\right)$;
Sample rows $i_1, \ldots, i_q$ from $\mathcal{D}_{\tilde{A}}$;
Let $\mathcal{F}$ denote the distribution given by choosing an $s \sim_u [q]$, and choosing a column from $\mathcal{D}_{A_{i_s}}$;
Sample columns $j_1, \ldots, j_q$ from $\mathcal{F}$;
Let W be the resulting $q \times q$ row-and-column-normalized submatrix $W_{rc} := \frac{A_{i_r j_c}}{q\sqrt{\mathcal{D}_{\tilde{A}}(i_r)\mathcal{F}(j_c)}}$;
Compute the left singular vectors of W $u^{(1)}, \ldots, u^{(k)}$ that correspond to singular values $\sigma^{(1)}, \ldots, \sigma^{(k)}$ larger than σ;
Output $i_1, \ldots, i_q$, $\hat{U} \in \mathbb{R}^{q \times k}$ the matrix whose ith column is $u^{(i)}$, and $\hat{\Sigma} \in \mathbb{R}^{k \times k}$ the diagonal matrix whose ith entry on the diagonal is $\sigma^{(i)}$. This is the description of the output matrix D;

Figure 4: MODFKV Algorithm

2.2.4 Main Algorithm

The quantum inspired classical recommendation system uses the two procedures mentioned to output a recommendation for a user. The algorithm is outlined below.

Input: Matrix $A \in \mathbb{R}^{m \times n}$ user $i \in |m|$, threshold σ, $\varepsilon > 0$, $\eta \in (0, 1]$

Output: Sample $s \in [n]$
Run ModFKV with parameters $(\sigma, \varepsilon, \eta)$ to get a description of $D = A\hat{V}\hat{V}^T = AS^T\hat{U}\hat{\Sigma}^{-2}\hat{U}^T S$.
Estimate $A_i S^T$ entrywise with parameter $\frac{\varepsilon}{\sqrt{K}}$ to estimate $\langle A_i, S_t^T \rangle$ for all $t \in [q]$. Let est be the resulting $1 \times q$ vector of estimates;
Compute est$\hat{U}\hat{\Sigma}^{-2}\hat{U}^T$ with matrix-vector multiplication;
Sample s from (est$\hat{U}\hat{\Sigma}^{-2}\hat{U}^T)S$
Output s;

Figure 5: Quantum inspired classical recommendation system

2.2.5 Error Analysis

Figure 6 shows the plot for MAE vs training set ratio for the dataset on which the Tang Recommendation algorithm was tested. We can see that the MAE reduces as the training set ratio increases.

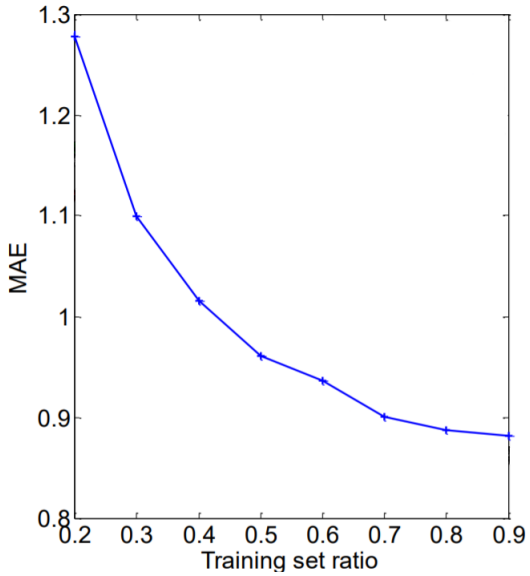

Figure 6: MAE values vs training set ratio

2.2.6 Code

The code which I implemented for the Ewin Tang Recommendation Algorithm can be found here.

2.3 Quantum Generative Adversarial Networks

QGAN algorithm described here was proposed by (Zoufal, 2019). The classical generative adversarial networks employ two neural networks - a generator and a discriminator to learn random distributions that are implicitly given by training data sample. Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution.

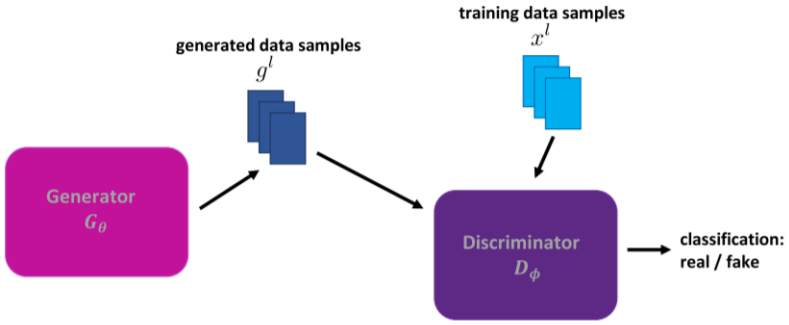

Figure 7: Generative Adversarial Network. First, the generator creates data samples which shall be indistinguishable from the training data. Second, the discriminator tries to differentiate between the generated samples and the training samples. The generator and discriminator are trained alternately

2.3.1 Quantum Distribution Learning

The QGAN implementation uses a quantum generator and a classical discriminator to capture the probability distribution of the classical training examples. In this setting, a parametrized quantum channel, i.e., the quantum generator, is trained to transform a given n-qubit input state $|\psi_{in}\rangle$ to an n-qubit output state

$$G_\theta|\psi_{in}\rangle = |g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle$$

where p_θ^j describe the resulting occurrence probabilities of the basis states $|j\rangle$.

2.3.2 Quantum Generator Model

The quantum generator is implemented by a variational form, i.e., a parametrized quantum circuit. We consider variational forms consisting of alternating layers of parametrized single-qubit rotations, here Pauli-Y-rotations (R_Y) and blocks of two-qubit gates, here controlled-Z-gates (CZ) called entanglement blocks U_{ent}. The circuit consists of a first layer of RY gates, and then k alternating repetitions of U_{ent} and further layers of R_Y gates. The rotation acting on the ith qubit in the jth layer is parametrized by $\theta^{i,j}$. Moreover, the parameter k

is called the depth of the variational circuit. If such a variational circuit acts on n qubits it uses in total $(k+1)n$ parametrized single-qubit gates and kn two qubit gates. Similarly to increasing the number of layers in deep neural networks, increasing the depth k enables the circuit to represent more complex structures and increases the number of parameters.

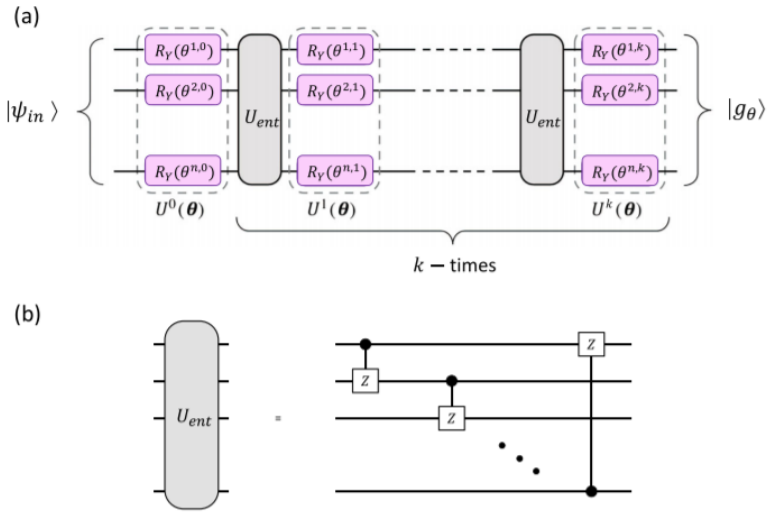

Figure 8: a) Quantum generator model b) Description of U_{ent}

The variational form depicted in (a) with depth k acts on n qubits. It is composed of $k+1$ layers of single-qubit Pauli-Y-rotations and k entangling blocks U_{ent}. (b) shows that each entangling block applies CZ gates from qubit i to qubit $(i+1)$ mod n, $i \in \{0, 1, ...n-1\}$ to create entanglement between the different qubits.

2.3.3 Classical Discrimator Model

The discriminator is a classical neural network consisting of a 50-node input layer, a 20-node hidden layer and a single-node output layer. First, the input and the hidden layer apply linear transformations followed by Leaky ReLU functions. Then, the output layer implements another linear transformation and applies a sigmoid function.

2.3.4 Training QGAN on MNIST

The QGAN model was trained on MNIST dataset. The qGAN is trained using AMSGRAD optimizer with the initial learning rate being 10-4. The results produced by the generator after various steps of training are as follows:

2.3.5 Code

The code which I implemented for the quantum generative adversarial network (QGAN) in the Tensorflow Quantum Framework can be found here.

Figure 9: Results at various steps of the image generated by quantum generator model at various steps

3. Conclusion

In this project, mainly three different algorithms were focussed namely the quantum recommendtion system by Kerenidis and Prakash, the quantum inspired classical recommendaion system by Ewin Tang and the quantum generative adversarial network (QGAN) by Zoufal et al.

4. Acknowledgement

I would like to thank Prof. Anirban Dasgupta for constant help and guidance throughout the course of the project

References

Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems, 2016. URL `https://arxiv.org/abs/1603.08675`.

Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019*, 2019. doi: 10.1145/3313276.3316310. URL `http://dx.doi.org/10.1145/3313276.3316310`.

Lucchi A. Woerner S. Zoufal, C. Quantum generative adversarial networks for learning and loading random distributions, 2019. URL `https://doi.org/10.1038/s41534-019-0223-2`.