Quantum Generalization of Natural Gradient Descent

Aradhita Sharma
EEE 606 Final Report
Electrical Engineering Department
Oncampus, MS, Arizona State University
ashar314@asu.edu

***Abstract*—The Gradient Descent optimization algorithm is used to find a function's minima. This algorithm is commonly used to train machine learning models and neural networks. The quantum generalization of this algorithm (Quantum Gradient Descent) provides the benefit of exponential speedup for problems with high-dimensional inputs. Quantum Geometric Tensor (QGT) is used to determine the optimization dynamics with respect to Quantum Information Geometry. This paper presents a comparison of gradient descent optimization performance in the classical and quantum domain. Furthermore, binary classification using neural network and quantum variational classifier are performed to compare the two approaches.**

Keywords— Gradient descent, optimization, machine learning, quantum generalization, Quantum gradient descent, exponential speedup, quantum geometric tensor (QGT), quantum information theory, binary classification, quantum variational classifier.

I. Introduction

Gradient descent is the most popular adaptive optimization algorithm used in machine learning and deep learning technology [2]. Gradient descent is the core foundation of deep neural network. It considers the first-order derivative to adapt the parameters of the system that minimize the error between predicted functions and empirical data. Various methods have been presented in the domain of gradient descent optimization such as RMSprop and Adam to enhance the neural network training performance. However, training massive datasets takes long training time per iteration, which is not time efficient. Therefore, research is being done to look out for ways to make the training time efficient [10].

Nowadays, quantum computation has become an emerging technology to integrate within machine learning to achieve quantum advantage [3]. It refers to the computational advantage that quantum computation can achieve for certain problems which are too complex to be solved by classical computation. Numerous quantum computing implementation approaches have taken optimization into account. Quantum circuits are being constructed for obtaining a quantum minimum searching algorithm and a quantum approximate optimization algorithm [4], [5]. The literature on variational quantum circuits has presented a number of optimization techniques for obtaining the best variational parameters, including derivative-free (zeroth-order) techniques like Nelder-Mead, finite-differencing [6], and SPSA [7]. Work has been done to evaluate the first-order derivative quantum algorithms.

In this work, an attempt is being made to develop a quantum algorithm for replicating the iterative optimization of gradient descent in quantum computing where the effect of geometry on the dynamics of variational algorithms has been examined [8]. The space of quantum states is a complex projective space that possesses Fubini-Study metric tensor. By computing the block-diagonal approximation to the quantum geometric tensor of the circuit, a quantum circuit for quantum gradient descent is constructed which outperforms vanilla gradient descent in terms of the number of iterations required to reach the convergence [1]. Furthermore, the implementation of this gradient descent algorithm is done to construct quantum binary classification system and is compared with classical binary classification system.

II. Theoretical Background

A. Gradient Descent Algorithm

Gradient is a measurement of how steep a function's slope is. For gradient descent algorithm to work on a function, the function should be differentiable having a derivative for each point and convex in nature having a local minimum point. For a function $f(x)$ to be convex, the second derivative should be always greater than zero.

$$\frac{d^2 f(x)}{dx^2} > 0$$

The primary goal of gradient descent is to find the convex function's minima value, where the slope of the function is zero. The following is the optimal gradient descent definition for a function's local minimum or maximum.

- For getting the local minimum of the function, move towards the negative gradient, that is, move away from the gradient of function at the current value.
- For getting the local maximum of the function, move towards the positive gradient, that is, move towards the gradient of function at the current value.

For optimization, function's minimum value is calculated, which is getting the local minimum of the function by moving towards the negative gradient as shown in the Fig 1. This is known as a method of gradient descent or steepest descent.

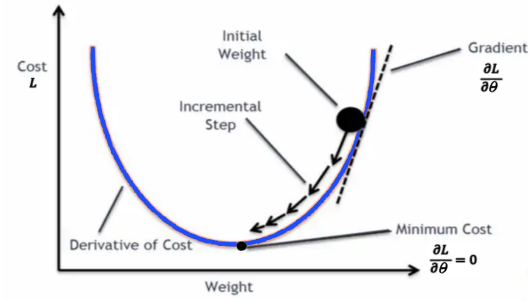


Fig. 1. Gradient descent algorithm.

The primary goal of gradient descent is to reduce the cost convex function through iterative parameter updates, where the cost function depends on these parameters. This error function (E) is defined as a measure of difference between true value and the predicted value given as

$$E = Y(true) - Y(predicted)$$

The mean square of this error function is calculated to obtain the cost function (L) shown in Fig 1 as

$$MSE = L(\theta) = mean(E^2(\theta))$$

where, the loss function depends on the variable parameters θ. Gradient descent algorithm is used to calculate next value of θ iteratively using the current value of θ and gradient of the value's cost function until the cost function is minimized. The iterative equation is written as

$$\theta(t+1) = \theta(t) + \eta * \frac{dL(\theta)}{d\theta}$$

where η is the learning rate or step size taken to reach the minimum of the cost function.
A lot of variations of gradient descent algorithm have been proposed to make this algorithm more efficient. In this project, vanilla gradient descent and ADAM optimization algorithms are implemented for the conducting the classical computations of gradient descent.

B. Vanilla Gradient Descent Optimization

Vanilla gradient descent is also known as batch gradient descent. This is the most basic variation of the gradient descent method. This algorithm computes the gradient of the cost function for the entire data. Here, the whole input is given to calculate the error first, and then the gradient is calculated and weights are updated.

$$\theta(t+1) = \theta(t) - \eta * \nabla L(\theta)$$

where, ∇ is the gradient of the cost function L. Since the whole data is given, this method results in stable gradient and stable convergence as a big advantage. However, because, for every iteration, the whole data is processed again and again, this optimization algorithm is slower [13].

C. ADAM Optimization

ADAM optimizer's name is derived from Adaptive Moment Estimation. This algorithm is s stochastic gradient descent method, where stochastic means random. It randomly picks one data point from the whole dataset at each iteration to reduce the computations. This is much more computationally efficient than vanilla gradient descent.

This technique calculates the adaptive learning rate for every parameter using momentum. Momentum is a method which accelerates the gradient descent in the relevant direction and decays the step size. When the gradients of two dimensions point in the same directions, the momentum term increases, and when the gradient changes the directions, the momentum term is reduced. As a result, oscillation is diminished and convergence becomes faster. [13] ADAM optimization keeps an exponentially decaying average of past gradient's mean (first moment m_t) and variance (second moment v_t), and makes the learning rate adaptive in the training phase using β_1 and β_2 values as the decay factors. The adaptive equation is given as

$$\theta(t+1) = \theta(t) - \frac{\eta}{\sqrt{v_t} + \epsilon} * m_t)$$

$$m_t = \beta_1 * m(t-1) + (1-\beta_1) * g_t$$

$$v_t = \beta_2 * v(t-1) + (1-\beta_2) * g_t^2$$

where, β_1, β_2 are the exponential decay rate with respect to first and second moments of the gradient, g_t is the previous gradient, m_t is the mean of the gradient as first moment, and v_t is the variance of the gradient as second moment. [13]

D. Natural Gradient Descent Optimization

Optimization in gradient descent is dependent to the Euclidean geometry. Gradient descent calculates the gradient based on Euclidean metric of the parameter space. Therefore, gradient descents have problems with badly scaled data, since the gradient will vary a lot in this situation.
This situation is explained in figure 2, two gaussians are considered which are solely parameterized by its mean, with varying variance in both images. The distance between the gaussians are same according to Euclidean metric (represented as red line), however, when analyzing the distributions of the gaussians, the distance is different for the first and second image. Therefore, Euclidean metric of the parameter space is not considering the distributions realised by the parameters. this information can be taken into account by finding the KL divergence of the two distributions. The first image has lower KL divergence, because of the greater overlap between the two gaussians as compared to the second image. This example is taken from [14].

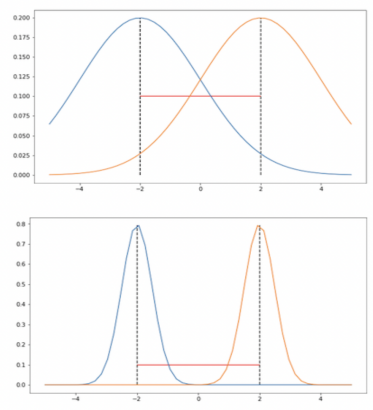


Fig. 2. Demonstration of two distributions for distribution space [14].

Therefore, the probability distribution, called as distribution space should be considered for steepest descent direction. Therefore, KL divergence is used to measure the difference between two models, the previous one and the updated one. KL divergence is the distance between two probabilistic distributions θ, θ' , named as distribution space, and steepest descent direction should be in that space.

$$D(\theta||\theta') = \sum(\theta(i) * log\frac{\theta(i)}{\theta'(i)}$$

1) Fisher Information Matrix: Since KL divergence is roughly analogous to a distance measure between distributions, this means Fisher information serves as a local distance metric between distributions, transforming the steepest descent in Euclidean parameter space to the steepest descent in the distribution space. Fisher information matrix is the second order derivative of KL divergence D.

$$F_\theta = \nabla^{2\prime}_\theta D(\theta'||\theta)$$

The inverse of this matrix, F inverse is multiplied with the gradient of loss function to make this natural gradient, and the whole process as natural gradient descent algorithm. The adaptive equation is given as

$$\theta(t+1) = \theta(t) - \eta * F^{-1} * \nabla L(\theta)$$

where F is the Fisher information matrix [15]. However, the calculation of the Fisher matrix and its inverse becomes computationally hard in classical domain. This fisher matrix can be calculated much faster in quantum domain.

E. Quantum Natural Gradient Descent Optimization

The analogous to natural gradient in quantum domain is quantum natural gradient descent. Just as the fisher information matrix, the term in quantum domain is fubini study metric tensor. When this metric tensor is restricted to the quantum states defining the parametric family, the real part of tensor gives the Quantum Geometric Tensor (QGT). This calculates the divergence between two distributions by computing the expectation values of the hermitian of a quantum state.

1) Quantum Circuit Representation of Quantum Geometric Tensor: The approximation of QGT as diagonal form is represented as :

$$\mathbf{QGT} = \begin{array}{c} \\ \theta_1 \\ \theta_2 \\ \\ \theta_L \end{array} \begin{array}{c} \begin{array}{cccc} \theta_1 & \theta_2 & \cdots & \theta_L \end{array} \\ \begin{pmatrix} G^1 & 0 & \cdots & 0 \\ 0 & G^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G^L \end{pmatrix} \end{array}$$

where θ_l parameterizes the l^{th} layer of the quantum circuit. QGT's block diagonal approximation is similar to the Fubini-Study metric tensor's block diagonal approximation.

$$g^l_{i^l_j} = Re(G^l_{i^l_j}) = G^l_{i^l_j}$$

Therefore, the Fubini-Study metric tensor can be calculated using the Hermitian observable's expectation values, and these expectation values are the values of the subcircuits of the full quantum circuit which is measurable. [9]

These obtained $g^l_{i^l_j}$ values corresponds to the values of the Fubini-Study metric tensor which is the quantum analogous of Fisher Information matrix. Therefore, the adaptive equation for quantum natural gradient descent becomes

$$\theta(t+1) = \theta(t) - \eta * g^+ * \nabla L(\theta)$$

where g^+ is the pseudo inverse of g, Fubini-Study metric tensor. Quantum Natural gradient (QNG) optimizer calculated the Fubini-Study metric tensor $g(\theta)$ at each iteration and updates the parameter θ values using the above equation.

III. Implementation

In this project, gradient descent algorithms are implemented and analysed for binary classification systems in classical and quantum domain. Firstly, for an adaptive system, three algorithms namely, vanilla gradient descent, ADAM, and quantum natural gradient descent are implemented for optimization of the system parameters, and their cost convergence curve are compared. Secondly, two binary classification systems are constructed using classical neural network and quantum variational classifier to compare classical and quantum optimizations. The implementation is described in the upcoming subsections.

A. Quantum Variational Circuits

In quantum computations, we can implement adaptive algorithms in variational quantum circuits, which are also called adaptable quantum circuits, in which the inputs are encoded and there are variational parameters as well, which are represented as quantum information.

As seen in Fig 3, the input value is x and the variational parameters θ are encoded forming the unitary transformation as $U(x, \theta)$. James, et al [1] have created a random variational circuit, for H2 molecule for finding the energy value with some

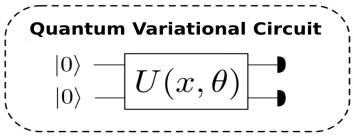


Fig. 3. Variational Quantum Circuit Representation.

initial parameters, which formed the variational circuit shown in the Fig 4. In this project, same H2 molecule variational circuit is designed to compared the loss convergence for gradient descent and quantum natural gradient descent.

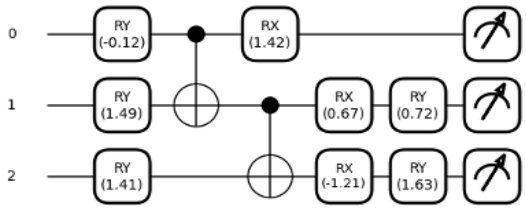


Fig. 4. Variational Quantum Circuit for H2 molecule

B. Gradient Descent Optimization Implementations

For implementing the quantum simulations, **Pennylane** library of python is used, which is an open-source software framework for quantum machine learnig [16]. For comparing the cost convergence of classical and quantum gradient descents, the H2 molecule variational circuit is chosen as the system with adaptive parameters which needs to be adapted. Circuit using 2 qubits and 3 qubits are designed for comparison of performance with different number of qubits. For 500 iterations, with 0.01 learning/adaptive rate, VGD, ADAM and QNG optimizers are compared. initial parameters are given input to variational circuit, updated parameters are calculated using different gradient descent optimizers and cost history is saved to further analysis, for 500 iterations. This cost history convergence is plotted to see the convergence rates using different optimizers.

C. Classical Binary Classification

For this project, dataset is collected from the University of California Irvine (UCI) machine learning repository [11]. This IRIS dataset is about classification of IRIS plants, which consists of 2 classes of IRIS plants, with 4 input features about the measurement of petal length and width, sepal length and width.

Classical Binary classification can be implemented using a neural network model, where the 4 input features are given to input layers, which are then computed with the weights associated with hidden layers, such that the output of output layer, that is the predicted output is as accurate as the desired output, as shown in Fig 5. These weights are updated after every iteration using gradient descent optimization.

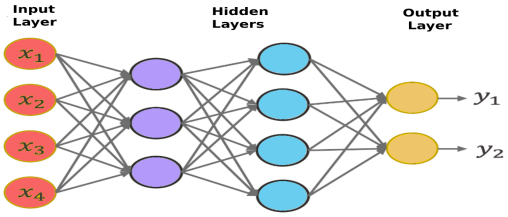


Fig. 5. Classical Binary classification model.

D. Quantum Binary Classification

Quantum classification model is also known as quantum variational classifier. A quantum circuit is designed which behaves similarly to a traditional machine learning algorithm. As seen in the Fig 6, the input data features (4 feature values) are encoded as quantum states represented as U(x). The second part consists of variational quantum circuit, which includes the adaptive parameters which are updated after every iteration, and the obtained measurement results are optmized using the gradient descent algorithms by evaluating the cost functions in classical domain, to be classified as the desired output class.

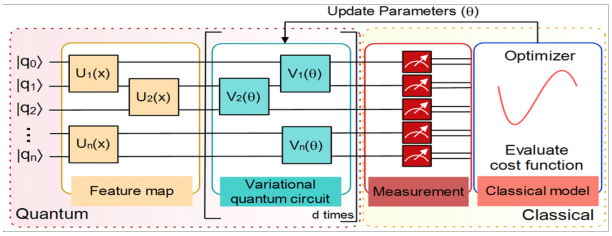
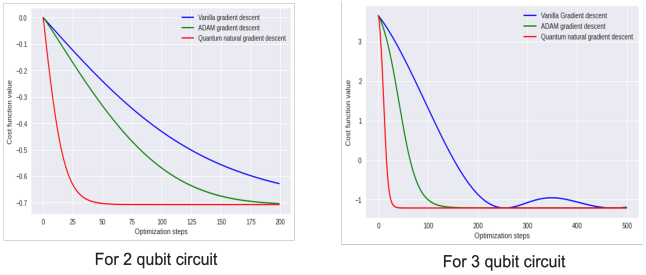


Fig. 6. Quantum Binary classification model [12].

IV. Results

A. Gradient Descent Comparisons

For the H2 molecule variational circuit, with random initial parameters, the gradient descent optimization techniques is compared for varying number of qubits, which is the circuit depth. The cost convergence curve using various optimizers is shown in Fig 7.



For 2 qubit circuit

For 3 qubit circuit

Fig. 7. Cost Convergence curves for a given quantum variational circuit with different optimizers.

In the first curve on the left side, for a variational circuit with 2 qubits only (2 nodes in each layer), it is seen that vanilla gradient descent optimization takes more number of iterations, the optimization steps to converge to a loss value. On comparison, ADAM optimizes faster as compared to the Vanilla gradient descent, because it considers the variable learning rate/step size. Whereas if the quantum natural gradient cost curve converges much faster with very few iterations compared to classical optimization algorithms. Quantum algorithm performs better since the information of probabilistic distributions is included. Comparing with the right graph of Fig 7, where the variational circuits are designed using 3 qubits, a fast convergence is seen as the number of qubits is increased. Therefore, "Quantum Natural Gradient optimizer appears more significant with increasing number of qubits and retains its advantage with increasing circuit depth" [1].

B. Binary classification results

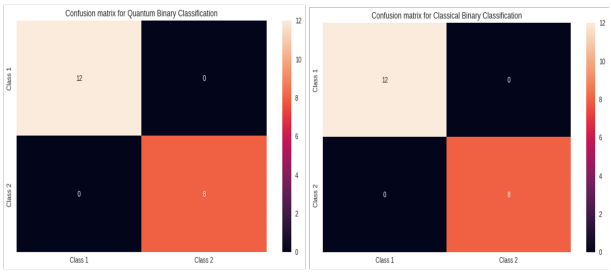


Fig. 8. Confusion Matrix for Binary Classification using classical and quantum computation.

For the classification analysis, binary classification has been performed using classical machine learning and quantum machine learning. It is seen that training using quantum variational classifier is much faster, as the loss function is reduced in a smaller number of epochs as compared to the epochs for loss function convergence using classical neural network model. However, for the testing samples, 100% accuracy is obtained using both the classification models with the confusion matrix shown in the Fig 8. Same results are obtained for both these classifications where 12 tests are classified correctly as class 1, and 8 tests for class 2 correctly.

V. Acknowledgement

I would like to express my gratitude to Professor Andreas Spanias for providing their guidance and suggestions throughout the course of the project. I would like to thank the center for Machine Learning and Intelligent Systems of University of California Irvine (UCI) for providing the IRIS Data Set in their Machine learning Repository to be publicly accessible.

VI. Conclusion

From this study, it is seen that Adam Optimization is more efficient than vanilla gradient descent. It has been said that quantum computing promises exponential speedup for some complex problems, and it is seen in this study that quantum natural gradient outperforms both the classical approaches, vanilla gradient descent and Adam optimization. As we increase the number of qubits, and number of layers of variational circuit, quantum natural gradient optimizer appears more significant, and retains its advantage with increasing circuit depth. It is seen that same classification results are obtained for both quantum and classical classification, where quantum classification model is optimized in less number of epochs (iterations) as compared to classical classification model.

References

[1] Stokes, James, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. "Quantum natural gradient." Quantum 4 (2020): 269.

[2] X. Wang, L. Yan and Q. Zhang, "Research on the Application of Gradient Descent Algorithm in Machine Learning," 2021 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2021, pp. 11-15, doi: 10.1109/ICCNEA53019.2021.00014.

[3] Ristè, D., da Silva, M.P., Ryan, C.A. et al. Demonstration of quantum advantage in machine learning. npj Quantum Inf 3, 16 (2017). https://doi.org/10.1038/s41534-017-0017-3

[4] Durr, Christoph, and Peter Hoyer. "A quantum algorithm for finding the minimum." arXiv preprint quant-ph/9607014 (1996).

[5] Farhi, Edward, and Aram W. Harrow. "Quantum supremacy through the quantum approximate optimization algorithm." arXiv preprint arXiv:1602.07674 (2016).

[6] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantumclassical algorithms. arXiv preprint arXiv:1701.01450, 2017.

[7] James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control, 37(3):332–341, 1992. DOI: 10.1109/9.119632.

[8] Aram Harrow and John Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. arXiv preprint arXiv:1901.05374, 2019.

[9] Naoki Yamamoto. "On the natural gradient for variational quantum eigensolver." arXiv:1909.05074, 2019

[10] Zou, F., Shen, L., Jie, Z., Zhang, W., Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. In Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (pp. 11127-11135).

[11] IRIS Dataset, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/iris

[12] Sen, Pinaki Bhatia, Amandeep. (2021). Variational Quantum Classifiers Through the Lens of the Hessian.

[13] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

[14] Pascanu, Razvan, and Yoshua Bengio. "Revisiting natural gradient for deep networks." arXiv preprint arXiv:1301.3584 (2013).

[15] Ly, Alexander, et al. "A tutorial on Fisher information." Journal of Mathematical Psychology 80 (2017): 40-55.

[16] pennylane.ai, Pennylane, https://pennylane.ai/