Quantum Natural Gradient Descent Algorithm[*]

Oza Harshkumar Ajaykumar (20929)
Quantum Technology, IISc Bengaluru

A quantum generalization of Natural Gradient Descent (NGD) is presented as part of a general-purpose optimization framework for variational quantum circuits. The optimization dynamics is interpreted as moving in the steepest descent direction concerning the Quantum Information Geometry, corresponding to the real part of the Quantum Geometric Tensor (QGT), also known as the Fubini-Study metric tensor. An efficient algorithm is presented for computing a block-diagonal approximation to the Fubini-Study metric tensor for parametrized quantum circuits..

Original paper: Quantum Natural Gradient by Stokes et al. (2020)

I. INTRODUCTION

Optimization, a fundamental concept familiar to us all, is crucial in almost every field of science and engineering. Enhancing the performance of a model involves minimizing the cost function. This pursuit of efficiency in achieving low-cost models drives continual exploration for novel optimization methods.

Quantum machine learning (QML) is a research area that explores the interplay of ideas from quantum computing and Machine Learning (ML). For example, we might be interested in exploring if quantum computers can train an ML model. On the other hand, we can also utilize techniques from ML to improve quantum error-correcting codes, estimate the properties of quantum systems, or develop new quantum algorithms.

Variational optimization of parameterized quantum circuits is an important component of many hybrid quantum-classical algorithms called Variational Quantum Algorithms (VQAs), the most promising applications of Noisy Intermediate-Scale Quantum (NISQ) computers. Applications include Variational Quantum Eigensolver (VQE), Quantum Approximate Optimization Algorithm (QAOA), and Quantum Neural Networks (QNNs) [5].

Researchers have used classical optimization techniques for VQAs and QML models for some time. These methods have been doing a good job, but the thing is, when it comes to quantum, we can do better. This report reviews an algorithm called Quantum Natural Gradient Descent (QNGD), which exploits a parameterized quantum space that allows for a more efficient optimization method than the well-known Stochastic Gradient Descent (SDG) algorithm (also called vanilla gradient descent).

[*] This report is scripted as a part of the course project for *DS 211: Numerical Optimization.* Visit: https://github.com/Dingley007/Quantum-Natural-Gradient-Descent-QNGD-algorithm

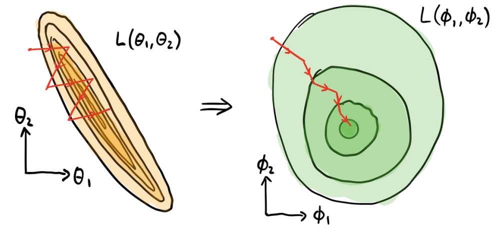


FIG. 1. Landcapes for different parameterizations of $\mathcal{L}(\theta)$ [4].

II. CLASSICAL GRADIENT DESCENT ALGORITHMS

Before understanding QNGD, let us briefly overview two classical gradient descent algorithms.

A. Stochastic Gradient Descent

The most common and simple form of gradient descent is the regular one. It can be described with the following formula

$$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{L}(\theta) \tag{1}$$

where θ_{n+1} is the new set of parameters, θ_n is the current set of parameters, η is the step size, and $\nabla \mathcal{L}(\theta)$ is the gradient with respect to our cost function.

The problem with the above approach is that each optimization step is strongly connected to *Euclidean geometry* on the parameter space. Since the choice of parameterization is not unique, different parameterizations can distort distances within the optimization landscape. For example, consider the the cost function $\mathcal{L}(\theta)$, parameterized using two different coordinate systems, (θ_0, θ_1), and (ϕ_0, ϕ_1) as shown in Fig. 1.

When we apply gradient descent in the (θ_0, θ_1) parameter space, we adjust each parameter by a uniform Euclidean distance η. However, this approach does not consider the differences in the rate of change of the cost function concerning each parameter. As a result, the optimization

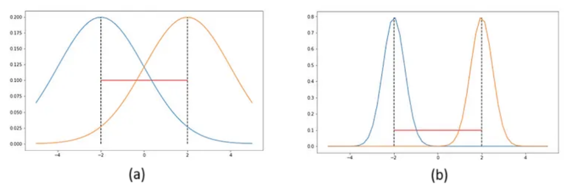


FIG. 2. Example showing why Euclidean metric doesn't work for distributions [1].

process may face challenges in locating or reaching the minimum, and in some cases, it might entirely overlook it.

Instead, after re-parameterization of the cost function, we might find a parameter space where variations in $\mathcal{L}(\theta)$ are similar across different parameters. This is precisely the case with (ϕ_0, ϕ_1) parameterization; the cost function is unchanged, but the new landscape is nicer to perform gradient descent. This leads to faster convergence and can help avoid local minima.

Can we avoid gradient descent in the parameter space altogether?

Suppose we consider the optimization problem as a probability distribution of possible output values given an input (i.e., maximum likelihood detection). In that case, a better strategy involves conducting gradient descent in the distribution space. This space is dimensionless and invariant for the parametrization. Consequently, each optimization step will consistently select the optimal step size for every parameter, independent of the specific parametric representation. In classical ML, the above process is known as *natural gradient descent*, explained in the next section.

B. Natural Gradient Descent (NGD)

In this method, the standard gradient descent is modified as

$$\theta_{n+1} = \theta_n - \eta F^{-1} \nabla \mathcal{L}(\theta) \tag{2}$$

where F is the *Fisher Information matrix.* It acts as a metric tensor, transforming the Euclidean parameter space's gradient descent to the distribution space's gradient descent. Let us understand the algorithm in more detail.

First, we need to define a new metric. In regular GD, we use a simple Euclidean metric to measure distances between parameters, but in the case of output distributions, this will not work. Let's understand this through an example.

Fig. 2(a) and 2(b) show Gaussians parameterized by only their '*means*' with variances fixed to 2.0 and 0.5, respectively. In both the images, the distances according to the Euclidean metric (red line) are the same, i.e., 4.0. However, clearly, in the distribution space (taking the shape of the Gaussians into account), the distance is different in (a) and (b).

So, when measuring the difference between two distributions, a popular choice would be the *Kullback Leibler (KL) Divergence.* The KL-Divergence measures the overlap and closeness between two output distributions. It allows us to work in the distribution rather than the parameter space.

Learning about the *Fisher Information Matrix* (F) is the last step to understanding the natural gradient descent. We know we need to use the KL-Divergence to optimize the distribution space; F provides precisely that. It defines the local curvature in the distribution space for which KL-Divergence is the metric.

Why has not this been used if it is so great?

In classical computing, calculating F and its inverse becomes computationally expensive as the number of parameters grows. And in typical neural networks, there are a lot of parameters. So, other state-of-the-art methods like momentum and adam (variants SDG) exist. But, if we translate this over into the quantum world, the complexity of this problem no longer scales as fast. This leads us to the quantum version of the natural gradient descent algorithm.

III. QUANTUM NATURAL GRADIENT DESCENT

This section explains the QNGD algorithm and shows some results using an example.

A. The Algorithm

QNGD is the same algorithm as the NGD, but we must adapt our metric appropriately as we have moved into the quantum world. We utilize geometric properties of parameterized quantum states, which is useful when optimizing a variational algorithm. Instead of the Fisher metric, we use *Fubini-Study metric* (or the quantum Fisher metric) represented by a matrix g. All we do with this metric is define distance within quantum geometric space, which is natural for quantum algorithms.

The Fubini-Study matrix is defined as,

$$g_{ij} = Re(\langle \partial_i \phi | \partial_j \phi \rangle) - \langle \partial_i \phi | \phi \rangle \langle \phi | \partial_j \phi \rangle \tag{3}$$

Where $|\phi(\theta)\rangle$ is our initial parameterized circuit (also called ansatz), and $\frac{\partial |\phi(\theta)\rangle}{\partial \theta_i}$ is the particle derivative of $|\phi(\theta)\rangle$ with respect to θ.

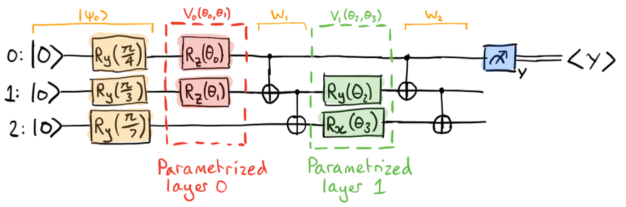
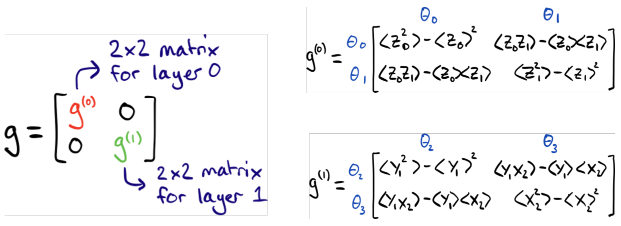


FIG. 3. An example of a parameterized circuit having two layers each with two parameters [4].



$g = \begin{bmatrix} g^{(0)} & 0 \\ 0 & g^{(1)} \end{bmatrix}$
2x2 matrix for layer 0
2x2 matrix for layer 1
$g^{(0)} = \begin{array}{c|cc} & \theta_0 & \theta_1 \\ \hline \theta_0 & \langle z_0^2\rangle - \langle z_0\rangle^2 & \langle z_0 z_1\rangle - \langle z_0\rangle\langle z_1\rangle \\ \theta_1 & \langle z_0 z_1\rangle - \langle z_0\rangle\langle z_1\rangle & \langle z_1^2\rangle - \langle z_1\rangle^2 \end{array}$
$g^{(1)} = \begin{array}{c|cc} & \theta_2 & \theta_3 \\ \hline \theta_2 & \langle y_1^2\rangle - \langle y_1\rangle^2 & \langle y_1 x_2\rangle - \langle y_1\rangle\langle x_2\rangle \\ \theta_3 & \langle y_1 x_2\rangle - \langle y_1\rangle\langle x_2\rangle & \langle x_2^2\rangle - \langle x_2\rangle^2 \end{array}$

FIG. 4. The Fubini-Study tensor calculation for block-diagonal approximation [4].

So, the QNGD equation then becomes,

$$\theta_{n+1} = \theta_n - \eta g^{+} \nabla \mathcal{L}(\theta) \tag{4}$$

Where g^{+} refers to the pseudo-inverse.

While the full Fubini-Study metric tensor cannot be evaluated on quantum hardware, we can compute a block-diagonal approximation. Also, it turns out that the block-diagonal approximation of the metric tensor can be advantageous over SDG. Let's study an example to understand the algorithm.

B. An Example

Consider the variational quantum circuit shown in Fig. 3. Each layer corresponds to a diagonal submatrix of g, with dimensions determined by the number of parameters in the layer. Since there are two layers with two parameters each, the block-diagonal approximation consists of two $2 \times x$ matrices.

The bock-diagonal matrices $g^{(0)}$ and $g^{(1)}$ are shown in Fig. 4. They are calculated by creating subcircuits of the original circuit.

Fig. 5 compares the cost function (the expectation value of Pauli-Y observable) value vs the iteration number for vanilla gradient descent and QNGD. We clearly observe that QNGD converges faster. Visit the GitHub repository here [2] for the detailed code.

Fig. 6 compares vanilla, Adam, block-diagonal, and diagonal QNGD algorithms. We can clearly observe that QNGD variants perform better than vanilla GD and Adam. The results are taken from [5].

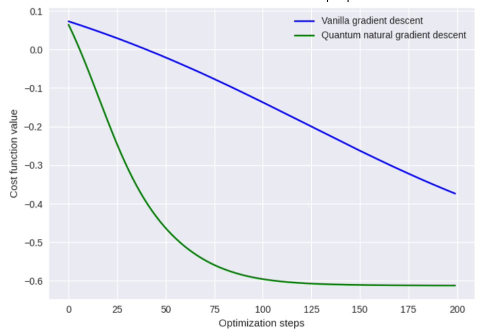


FIG. 5. The cost function value for 9 qubits and $L = 3, 4, 5$ layers vs iteration number for four different algorithm [4].

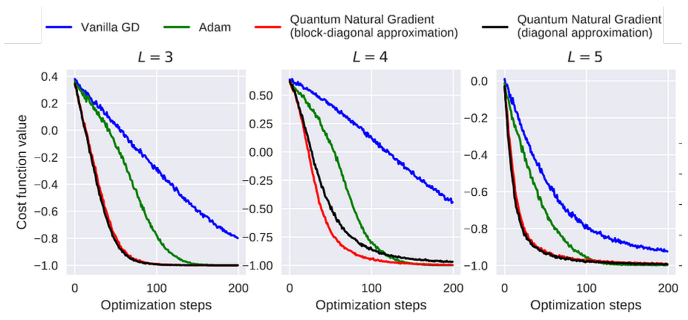


FIG. 6. An example of a parameterized circuit having two layers each with two parameters [5].

IV. CONCLUSION

The QNGD algorithm is a quantum generalization of NGD with a different metric tensor called the Fubini-Study metric tensor. This matrix's block-diagonal and diagonal forms are easy to calculate on quantum hardware, showing its advantage over NDG, SDG, and Adam. So, QNGD is a promising candidate for VQAs.

REFERENCES

[1] Lana Bozanic. Medium: Quantum natural gradient from the ground up, 2020.

[2] Oza Harshkumar. Github: Quantum natural gradient descent (qngd) algorithm, 2023.

[3] Josh Izaac. Pennylane: Quantum natural gradient, 2021.

[4] Josh Izaac and Nathan Killoran. Medium: Optimizing quantum computations with the quantum natural gradient, 2019.

[5] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum Natural Gradient. *Quantum*, 4:269, May 2020.