December 28, 2020

Digital Object Identifier NaN

An Efficient Optimization Method: Natural Gradient Descent

TOGAN TLIMAKHOV YUSUF
Department of Electrical and Electronics Engineering, Ankara University, 06830 Ankara, Turkey
Corresponding author: Togan Tlimakhov Yusuf (e-mail: t0gan@icloud.com)

ABSTRACT Gradient Descent is a technique used to optimize a given cost function for a learning model. Although this technique is broadly used, it has a lot of limitations. This paper aims to explicate the natural gradient descent, an optimization algorithm that is useful for finding the local minimum of a differentiable function. We start by defining several concepts that are necessary to understand the gradient descent, then we explain the limitations of the standard gradient descent, which is generally used to find the values of a function's parameters that minimize a cost function as much as possible. In the next section, we introduce the natural gradient descent, where we explain the theoretical background and give examples to demonstrate its efficiency in optimizing parameters for a required path and on a given data set. Finally, an introduction to the quantum analog of natural gradient descent, the quantum natural gradient is made, which is useful for optimizing the parameters used in variational quantum circuits, where we first construct a variational circuit, then we try to optimize its parameters using both the Qiskit and Pennylane libraries. In the end, we compare both natural and standard gradient algorithms on the variational circuit.

INDEX TERMS Optimization, Gradient Descent, Natural Gradient Descent, Quantum Natural Gradient Descent, Variational Quantum Circuit.

I. INTRODUCTION

OPTIMIZATION techniques are one of the most significant steps during the development of a learning algorithm. Generally, when given a set of data we try to construct a useful relation in between the data or more precisely, we try to find a pattern that will allow us to construct a mathematical model that will help us not only fit a function into the given data but more importantly predicting future outcomes that we don't know yet. During this process, we try to develop a function that fits out data the best, this function; sometimes referred to as a hypothesis can be any linear, polynomial or something else, the only property that matters is to fit the given data. The hypothesis that we develop will have a certain number of constant values that will be multiplied with the dependent variables, these constants are referred to as parameters of the hypotheses. After deciding the structure of the mathematical model, we try to adjust the parameters to a certain value, indeed the values that the parameters hold will decide how good the hypothesis is, alongside the structure of the function. Although it might seem that the higher the degree of the function is, the more accurate it will be, certain data-sets won't require complex higher degrees polynomials, and using them will only make the hypothesis unnecessarily complex and expensive.

After having developed a function with certain parameters, the next question is what values do we assign to these parameters and how do we determine how good the predictions are after choosing certain values? The best algorithm for this purpose is the cost function, also referred to as the loss function, the cost function takes the average difference of the results of the hypothesis with inputs from the independent variables and the actual output of the dependent variable. After having a cost function that will determine the efficiency of the hypothesis we need to develop an algorithm that will choose these parameters in a way that the predictions are accurate and the cost function is minimized i.e. the difference between the provided data and the hypothesis is very low. This type of algorithm is the main topic of this paper. In the rest of the paper, we will focus on developing an efficient algorithm that will assign the best parameters to the hypothesis hence the best predictions. We will show that there is a different type of algorithms that are better for different data sets and a comparison between them are then conducted to decide which one fits the data better.

II. GRADIENT DESCENT AND ITS LIMITATIONS

Gradient descent is a useful algorithm for adjusting a set of given parameters to minimize the cost function. In order to understand the gradient descent, we first introduce the mathematical presentation of the cost function as follow:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n. \tag{1}$$

Where m is the number of training sets in the data set, h is the hypothesis that we use, and y is the actual values of the given data set. One algorithm that will minimize the cost function is gradient descent.

Gradient descent is an optimization algorithm that changes the values of the parameters until convergence, to get the theta value with a minimum cost function. Gradient descent uses the following equations simultaneously to update the theta value until convergence to a minimized cost function.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{2}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}).x^{(i)} \tag{3}$$

In both equations (3) and (4), we update the parameters to get the theta value with the minimum cost function. The learning rate or step size, a, is a variable that determines the step size of gradient descent convergence.

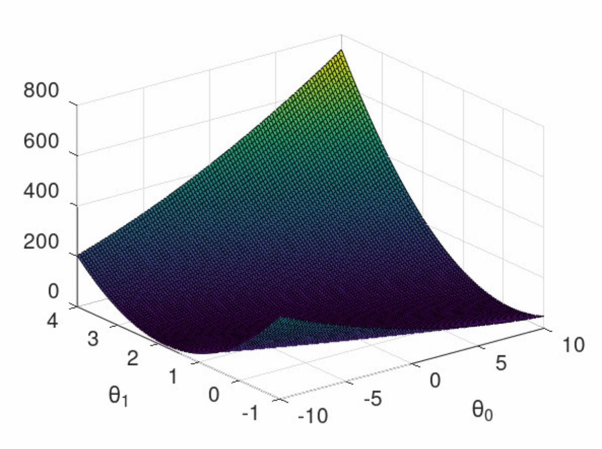


FIGURE 1. An example for the parameter space for an isotropic cost function with single local minimum that is easy for gradient descent to solve.

Although the gradient descent algorithm makes good predictions for cost functions that have a single minimum and whose gradients are isotropic in magnitude with respect to any direction away from this minimum. In practice, the cost functions that are being optimized are generally multi-model, and the gradient magnitudes are non-isotropic about any minimum. In such cases, the parameter estimates are only guaranteed to locally minimize the cost function, and convergence to any local minimum can be slow.

In the following examples, we will demonstrate the limitations of gradient descent.

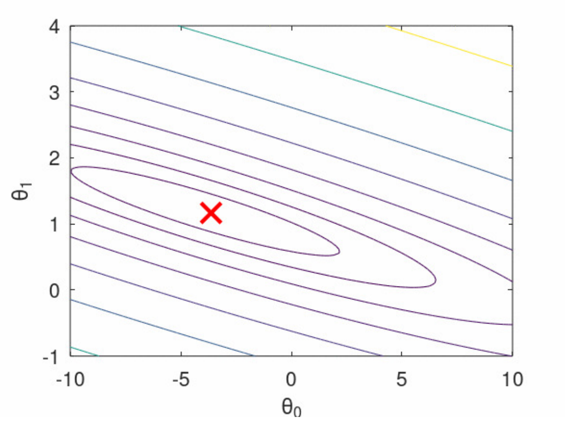


FIGURE 2. Visualisation of the parameter space for an example with two parameters, the point of "X" represents the optimum solution.

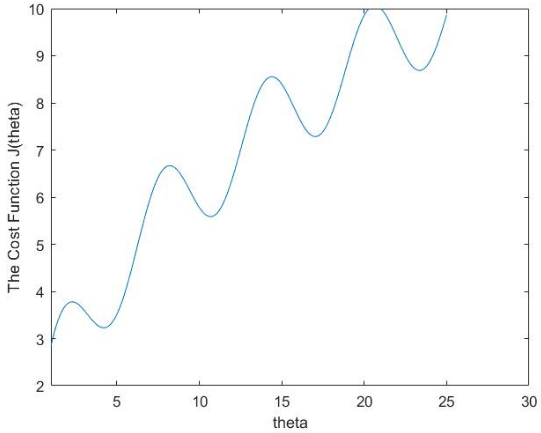


FIGURE 3. Demonstration of local minimums in a function. The values of the function where theta equals around 24, 17, and 12 are all local minimums that the gradient descent algorithm may fail to overcome.

A. THE PROBLEM WITH LOCAL MINIMUM

Mathematically, the local minimums of a function, are the smallest value of the function, within a given range (the local), or in the case of the entire domain (the global or absolute extrema), it is called the global minimum. Figure 2 shows that the gradient descent algorithm may get stuck in one of the local minimum, which is a big problem to overcome.

B. THE PROBLEM OF STEP SIZE AND MONOTONICITY

Another issue with gradient descent is the step size a, which is proportional to the gradient size. We adjust the parameters, x, using the gradient of the cost function. One must be aware that the scaling of the x-axis is admittedly arbitrary. Ideally, an optimization algorithm should be invariant under the rescaling of the x-axis. The same holds for the y-axis. The plain gradient df(x) dx is of course not re-scaling/transformation invariant. Furthermore, functions may have very different characteristics in different regions: Consider the function in Figure 4; while the right region is near flat (small gradient), it is very steep (large gradient) in other regions [10]. This may cause an issue while updating the parameters in large steps; steps that are larger than the fall region in this example.

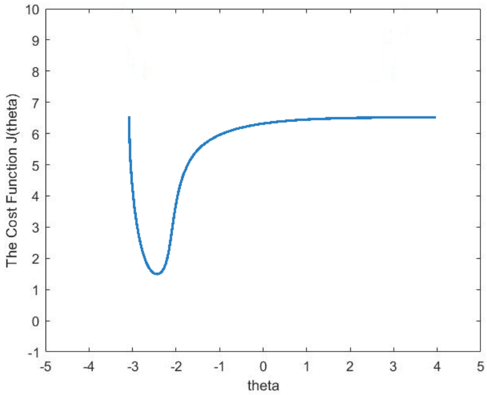


FIGURE 4. The function demonstrate that a small gradient dose not imply a small step-size or a large gradient a large step-size.

III. NATURAL GRADIENT DESCENT

In order to understand the natural gradient descent, we must reconsider the fundamental notion of distance in the physical meaning. Mathematically, the Euclidean distance between two points v and v +δv in the N-dimensional space spanned by v = [v$_1 v_2 ... v_n]^T$ is defined as follow [7]:

$$d_E(v, v+\delta v) = \sqrt{\sum_{i=1}^{N} \delta {v_i}^2} = \sqrt{\delta v^T \delta v} = ||\delta v||_2 \qquad (4)$$

Where ||v|| denotes the Euclidean norm of v. Although this notation hold for most cases, it is an approximation of a more complex notion of distance. A more accurate geometry would be employing the mathematics of curved space, a field known as differential1 or Riemannian geometry [9]. It is Riemannian geometry upon which natural gradient adaptation is based [3]. In Riemannian geometry, the distance between two objects is not measured according to the Euclidean norm as in (4). Rather, for two vectors w and w + δw where the elements of δw are of small value, we define the new metric as follow:

$$d_w(w, w+\delta w) = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} \delta w_i \delta w_j g_{ij}(w)} \qquad (5)$$

$$= \sqrt{\delta w^T G(w) \delta w} \qquad (6)$$

where G(w), is defined as the Riemannian metric tensor, that is (N x N) positive definite matrix whose (i, j)th entry is g$_{ij}$(w). The Riemannian metric tensor characterizes the intrinsic curvature of a particular manifold in N-dimensional space. If in the equation of the Euclidean coordinate system, G(v) = I is the identity matrix, in this case (5) reduces to (4). As stated in [4], the natural gradient adaptation is asymptotically Fisher-efficient. Employing the natural gradient algorithm for the maximum-likelihood cost function in this situation with u(k) = 1/k, the asymptotic covariance matrix of the resulting parameter errors approaches the Cramer-Rao lower bound for unbiased parameter estimates. Now we will introduce the Fischer information matrix, which is the Riemannian metric tensor in the space of the parameters within a statistical model. The Fisher information matrix is a key element in the deduction of natural gradient descent. Several new concepts will be defined throughout the processes.

The learning rate update is the key for the learning algorithm so that it doesn't blindly move from one point to the other and miss reaching the global minimum. As we have seen, the Euclidean geometry has some limitations, therefore instead of fixing the euclidean distance each parameter moves i.e. the distance in the parameter space, it is possible to fix the distance in the distribution space of the required output. In other words, instead of changing all the parameters within an epsilon distance, we focus on the output distribution of the model to be within an epsilon distance from the distribution from the previous step. The key step at this point is measuring the distance between two distributions. For this, we will be introducing the Kullback-Leibler divergence.

Mathematically, the Kullback–Leibler divergence is a measure of how one probability distribution is different from a second reference probability distribution. In some sense it can be considered a distance in the locality it is defined. Since we are concerned about how the output distribution changes when we make small steps in the parameter space, the Kullback–Leibler divergence is a good method. The essential conclusion is that unlike the normal gradient descent, we are not moving in Euclidean parameter space, but rather in the distribution space with the Kullback-Leibler divergence as the metric.

$$x_{t+1} = x_t - \eta_t \cdot F(\theta_t)^{-1} \cdot \frac{df(x)}{dx} \qquad (7)$$

After having defined the Kullback-Leibler divergence, the Fisher Information Matrix is deduced from it and it changes the gradient descent, as (3) will turn into (7), Where F(θ_t) is the Fisher Information Matrix, transforming the gradient descent algorithm into the natural gradient descent moving in the distribution space. The final equation that relates the Kullback-Leibler divergence and the Fisher Information Matrix is deduced in [3] as follow:

$$KL(p(x;\theta_t)||p(x;\theta_t+\delta\theta)) \approx \frac{1}{2} \cdot \delta\theta^T \cdot F \cdot \delta\theta \qquad (8)$$

Where KL(p(x;θ_t)$||p(x;\theta_t+\delta\theta)$) is the Kullback-Leibler divergence of the two distributions in which the p(x;θ_t) and p(x;θ_t + $\delta\theta$) are the likelihood functions. The likelihood function is defined as the measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters. Furthermore, the Fisher Information

Matrix can be viewed as the curvature of the log-likelihood function and it is also the covariance of the score function, that is the log-likelihood which measures the efficiency of the predictions.

Although the natural gradient descent algorithm is extremely efficient in predicting with the help of the Fisher Information Matrix in transferring the parameters into the distribution space, as can be seen in (8), the equation is seemingly hard to use and the algorithm for finding the Fisher Information Matrix is expensive to calculate on a regular set of data as of linear regression in the previous examples. Therefore, to demonstrate the efficiency of natural gradient descent, we will go through two procedures. First, we will use the example given in [4] to show how natural gradient descent is more efficient when changing the space parameters. Then, we will use a simplified version of the Fisher Information Matrix to calculate it on a given data set, where we will compare the outcome of the cost function with comparison to the standard gradient descent.

IV. DEMONSTRATING THE EFFICIENCY OF NATURAL GRADIENT

A. THE RIEMANNIAN METRIC

To demonstrate the difference between the two algorithms, we will try first to combine the equations of Euclidean and Riemannian distances, when the nature of the manifold can be described in terms of a transformation of Euclidean orthogonal space with coordinate vector v to w, then one can determine the form of G(w) through the following relationship:

$$d_E^2(v, v+\delta v) = d_W^2(w, w+\delta w) \tag{9}$$

(9) is a mathematical relation between (4) and (5) [9], where δv is the small and w + δw is the transformed value of v + δv. We will illustrate this relation with an example [4].

We define v = [x $y]^T$ such that [x,y] define a point in two-dimensional Euclidean space. We now represent the same point using polar coordinates, where

$$x = r\cos(\theta), \quad y = r\sin(\theta) \tag{10}$$

This way we represent the same point in polar space. Where w = [r $\theta]^T$ represent the exact same point in polar space. The distance between v and v + δv in Euclidean space was

$$d_E(v, v+\delta v) = \sqrt{\delta x^2 + \delta y^2} \tag{11}$$

From (11), a deduction of The Riemannian metric can be made using the relation in (9) as follow:

$$v + \delta v = \begin{pmatrix} (r+\delta r)\cos(\theta+\delta\theta) \\ (r+\delta r)\sin(\theta+\delta\theta) \end{pmatrix} \tag{12}$$

$$= \begin{pmatrix} r\cos\theta + \delta r\cos\theta - \delta\theta r\sin\theta \\ r\sin\theta + \delta r\sin\theta + \delta\theta r\cos\theta \end{pmatrix} \tag{13}$$

In (13), we have neglected the two terms with δ for i>1. If we subtract v from both sides, we are left with

$$\delta v = \begin{pmatrix} \delta r\cos\theta - \delta\theta r\sin\theta \\ \delta r\sin\theta + \delta\theta r\cos\theta \end{pmatrix} \tag{14}$$

And therefore, using (14), (11) becomes

$$d_E^2(v, v+\delta v) = \delta r^2 + r^2\delta\theta^2 \tag{15}$$

Furthermore, we write (15) as in the matrix form

$$d_E^2(v, v+\delta v) = \delta w^T G(w)\delta w \tag{16}$$

From these equations, G(w), the Riemannian metric for w is found to be

$$G(w) = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix} \tag{17}$$

It is important to note that although this example indicates that G(w) depends on the value of w. In general, it can be constant-valued in certain cases. The most general example is when w is obtained from v in standard Euclidean space via a linear transformation [7]. Furthermore, despite the form of the result in (17), G(w) is not diagonal in general, and its structure may vary depending on the type of transformation.

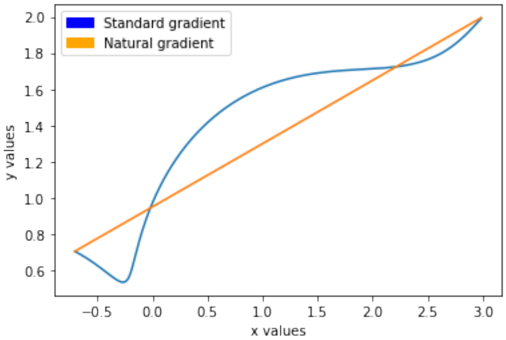


FIGURE 5. Comparison between the standard and natural gradient algorithm, where the Riemannian geometry here is considered to be the polar form. As seen, the natural gradient is more efficient in reaching the aimed destination.

After having solved the equation for the Riemannian metric G(w), which is the Fisher information matrix in the space of the parameters within a statistical model as stated before, we will now demonstrate this example on a specific point and compare both the natural and standard gradient descent algorithms. In this code, we set the specified point as x, y = [3.0, 2.0]. The results obtained as in Figure 5, the interpretation of this outcome is as we stated previously, the standard gradient descent aims for the direction of steepest descent in the parameters space, where the natural gradient descent aims to find the best direction on the underlying dimension.

B. UPDATING THE COST FUNCTION FOR A GIVEN DATA SET USING NATURAL GRADIENT DESCENT

Since we have confirmed that the natural gradient algorithm is efficient in obtaining the necessary results. It is important to measure its performance of a given data set. But as we have mentioned earlier, it is generally expensive to use the algorithm especially on regression models, therefore we are going to construct a simplified version of the Fisher Information matrix, to use it on the data set, then we will calculate the cost function for both the natural and standard gradients and observe which algorithm is more efficient.

As expressed in [3], natural gradient descent is an approximate second-order optimization method and has been applied successfully in many domains [6]. By using the positive semi-definite Gauss-Newton matrix to approximate the Hessian, which is a square matrix of second-order partial derivatives of a scalar-valued function, that describes the local curvature of a function of many variables. Natural gradient descent often works better than exact second-order methods, as stated before natural gradient descent is expensive in high dimensional parameter spaces, even with approximations [11]. But in this deduction, the algorithm will be modified to be less complex. From (7), given the gradient of x, g = $\partial f(x)/\partial x$, the natural gradient algorithm computes the update as follow:

$$\Delta z = aF^{-1}g \tag{18}$$

where a is the learning rate and F, the Fisher information matrix is defined as

$$F = E_{p(t|z)}[\nabla \ln p(t|z) \nabla \ln p(t|z)^T] \tag{19}$$

The log of the likelihood function $\ln p(t|z)$ typically corresponds to commonly used error functions such as the cross-entropy loss. This correspondence is not necessary when we interpret natural gradient descent as an approximate second-order method, as has long been done this way [11]. Therefore the empirical Fisher F' with Tikhonov damping, which is is a method of regularization of ill-posed problems (An example of Tikhonov regularization is ridge regression, which is particularly useful to mitigate the issue of multicollinearity in linear regression) is used to obtain

$$F' = g.g^T + \beta I \tag{20}$$

The most important property of (20) is that the F' is cheaper to compute compared with the full Fisher since "g" is already available. The damping factor β regularises the step sizes, which is important when F' only poorly approximates the Hessian or when the Hessian changes too much across the step. Using the Sherman-Morrison formula, which computes the inverse of the sum of an invertible matrix F and the outer product, uv^T of vectors u and v. the natural gradient descent update can be simplified into the following [12]:

$$\Delta z = a\,(\frac{I}{\beta} - \frac{g\,g^T}{\beta^2 + \beta g^T g})\,g = \frac{a}{\beta + ||g||^2} g \tag{21}$$

By turning (18) into (21), we have developed an important change, which is avoiding any matrix inversion that makes the algorithm expensive. Therefore, the natural gradient descent adapts the step size according to the curvature estimate c = 1 / β + $||g||^2$. If β is small, natural gradient descent normalizes the gradient by its squared L2 norm. Natural gradient descent automatically smooths the scale of updates by downscaling the gradients as their norm grows, which also contributes to the smoothed norms of updates [5].

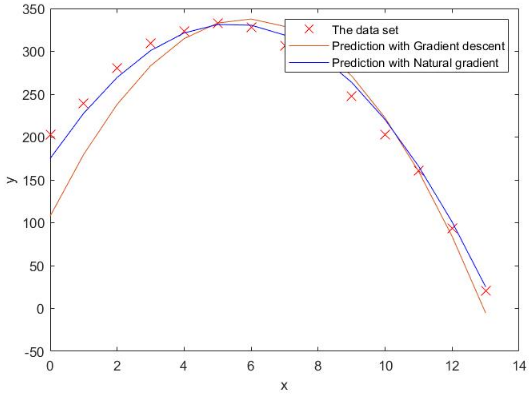


FIGURE 6. Demonstration of a function obtained using both the standard and natural gradient algorithms. A data set was given and the hypothesis was constructed accordingly.

In Figure 6, it is clear that the natural gradient algorithms allow a better function that fits the given data, more precisely the cost function for both algorithms was calculated after the final updates and the results were as follow;

$$Cost\,function\ J(\theta)\ after\ GD\ =\ 632.013118$$
$$Cost\,function\ J(\theta)\ after\ NGD\ =\ 70.101444$$

where the natural gradient algorithms provided a lower cost value, which means the difference between the actual data and the developed hypothesis is lower, hence the algorithm is more efficient.

V. THE QUANTUM NATURAL GRADIENT

Despite the fact that we tried different optimization algorithms and have compared them, all previous algorithms were classical in the sense that they will run on a classical computer. One of the best motivations to work with these algorithms is to try to run the of a quantum computer. Unlike classical circuits, quantum circuits use a different type of gates that their outcome is probabilistic. Formally, a quantum circuit is a model for quantum computation in which the computation is an algorithm of a sequence of quantum gates, which are reversible transformations on a quantum mechanical analog of an N-bit register. This analogous structure is referred to as an N-qubit register, where qubit refers to a quantum bit. Furthermore quantum variational circuits are quantum circuits that are quantum algorithms that depend on free parameters [2] which we will try to calculate the cost function of using natural gradient descent.

The quantum analog of natural gradient descent 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 with respect to the quantum information geometry, corresponding to the real part of the Fubini-Study metric tensor which is the quantum equivalent of the Fisher information matrix. An efficient algorithm is presented for computing a block-diagonal approximation to the Fubini-Study metric tensor for parametrized quantum circuits, which may be of independent interest, The quantum equivalent equation for natural gradient descent is given as following [2]:

$$\theta_{t+1} = \theta_t - \eta g^{+}(\theta_t)\nabla L(\theta_t) \tag{22}$$

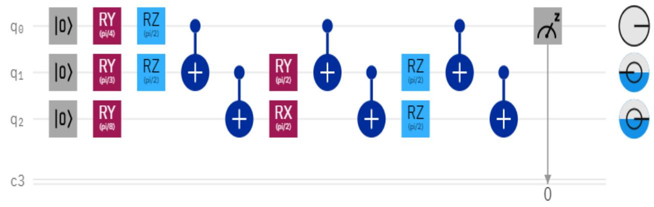
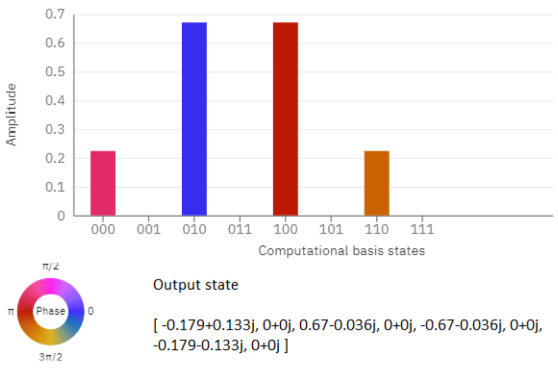


FIGURE 7. A simple variational quantum circuit that consists of 6 parameters, with 3 distinct parametrized layers of 2 parameters.

where g^{+} is the pseudo inverse of the Fubini-Study metric tensor. In variational quantum algorithms, a low-depth parametrized quantum circuit ansatz is chosen, and a problem specific observation is made. Then, an optimization loop is used to find the set of quantum parameters that minimize a particular measurement expectation value of the quantum device. Examples of such algorithms include the variational quantum eigensolver, the quantum approximate optimization algorithm, and quantum neural networks [14].



Output state

```
[ -0.179+0.133j, 0+0j, 0.67-0.036j, 0+0j, -0.67-0.036j, 0+0j,
-0.179-0.133j, 0+0j ]
```

FIGURE 8. A statevector representation of the outcome of the variational quantum circuit.

Using the qiskit [15] and pennylane [13] libraries, we are going to create a simple variational quantum circuit that in Figure 7, then try to optimize the parameters using both standard and natural gradient algorithms, Finally, we will observe both algorithms' efficiencies.

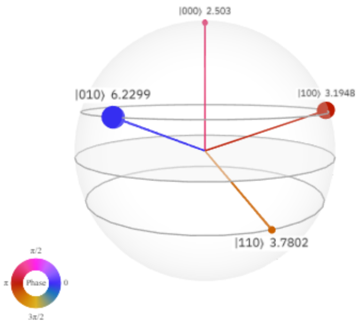


FIGURE 9. A Bloch sphere representation of the possible outcome of the variational quantum circuit in Figure 7.

After having constructed a variational quantum circuit with 6 parameters, in which 3 distinct parametrized layers of 2 parameters are used, we will try to optimize these parameters. using the optimizer in [13] that provides an implementation of the quantum natural gradient optimizer, we will try to compare the optimization convergence of the quantum natural gradient optimizer and the natural gradient for the variational circuit in Figure 7.

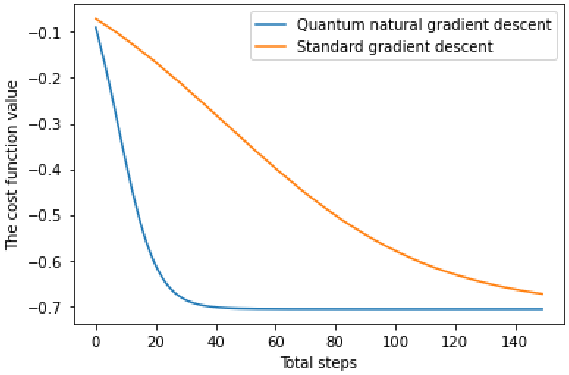


FIGURE 10. The cost function value versus the total steps for both quantum natural gradient descent and standard gradient descent algorithms.

VI. CONCLUSION

In conclusion, we have started by introducing natural gradient descent, where we defined several concepts that helped us understand the aim of the algorithm, then we explained the limitations of the standard gradient descent by pointing out two major weaknesses in the algorithm. Next, we introduced the natural gradient algorithm, where we learned that in the natural gradient, we are no longer considering the movement of the parameters in the parameter space. Instead, we arrest the movement of the output probability distribution at each step. We have seen that the Log-Likelihood, KL divergence, and the Fisher Information Matrix are key concepts to achieve this goal.

Although the natural gradient descent is remarkably efficient in predicting parameters, we have concluded that the application of it was more difficult than the theoretical deduction, and the Fisher Matrix turned out to be expensive to calculate, especially as the number of parameters grows. Yet, it turned out that this difficulty facing classical computing was not an obstacle for a quantum computer that operates fundamentally different than its classical counterpart, where we have constructed a variational quantum circuit and compared both natural and standard algorithms. Nonetheless, there are still some cases where the natural gradient is used to estimate relatively small parameters, or the expected distribution is relatively standard like a Gaussian distribution, and in areas like Reinforcement Learning, besides the apparent efficiency, it exhibited on the variational circuit.

ACKNOWLEDGMENT

This work is not based on complete research and the paper is not officially published.

REFERENCES

[1] F. F. Lubis, Y. Rosmansyah and S. H. Supangkat, "Gradient descent and normal equations on cost function minimization for online predictive using linear regression with multiple variables," 2014 International Conference on ICT For Smart Society (ICISS), Bandung, 2014, pp. 202-205, doi: 10.1109/ICTSS.2014.7013173.

[2] James Stokes, Josh Izaac, Nathan Killoran, Giuseppe Carleo. “Quantum Natural Gradient.” arXiv:1909.02108, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, 2019.

[3] S. Amari, "Natural Gradient Works Efficiently in Learning," in Neural Computation, vol. 10, no. 2, pp. 251-276, 15 Feb. 1998, doi: 10.1162/089976698300017746.

[4] S. Amari and S. C. Douglas, "Why natural gradient?," Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP ’98 (Cat. No.98CH36181), Seattle, WA, USA, 1998, pp. 1213-1216 vol.2, doi: 10.1109/ICASSP.1998.675489.

[5] Y. Wu, J. Donahue, D. Balduzzi, K. Simonyan, T. Lillicrap: “LOGAN: Latent Optimisation for Generative Adversarial Networks”, DeepMind, London, UK 2019; arXiv:1912.00953.

[6] Pascanu, Razvan Bengio, Y.. (2013). Revisiting Natural Gradient for Deep Networks.

[7] S.C. Douglas and S. Amari, "Natural gradient adaptation," Proc. IEEE, vol. 86, to appear. no. 6, June 1998.

[8] J. Martens, "New insights and perspectives on the natural gradient method", DeepMind, martens 2020 new, arXiv:1412.1193, 2020.

[9] Riemann, Bernhard. (2016). On the Hypotheses Which Lie at the Bases of Geometry. 10.1007/978-3-319-26042-6.

[10] M. Toussaint, "Lecture Notes: Some notes on gradient descent", Machine Learning and Robotics lab, FU Berlin Arnimallee 7, 14195 Berlin, Germany. [Online]. Available: https://www.user.tu-berlin.de/mtoussai//notes.

[11] J. Martens: “New insights and perspectives on the natural gradient method”, 2014; arXiv:1412.1193..

[12] Bartlett, Maurice S. (1951). "An Inverse Matrix Adjustment Arising in Discriminant Analysis", Annals of Mathematical Statistics. 22 (1): 107–111., doi:10.1214/aoms/1177729698, MR 0040068, Zbl 0042.38203.

[13] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, N. Killoran. "PennyLane: Automatic differentiation of hybrid quantum-classical computations", 2018.

[14] N. Yamamoto. “On the natural gradient for variational quantum eigensolver.”, Keio University, Hiyoshi 3-14-1, Kohoku, Yokohama, 223-8522, Japan, arXiv:1909.05074, 2019.

[15] Qiskit, "Qiskit: An Open-source Framework for Quantum Computing", 2019, 10.5281/zenodo.2562110.

TOGAN TLIMAKHOV YUSUF is B. Sci. student at Ankara University, Department of Electrical and Electronics Engineering. He is a member of the APS and IEEE Quantum, where his first paper was presented in the 2021 IEEE Quantum week. His research interests include quantum computation and information, machine learning, and computational physics.