QUANTUM-INSPIRED ADAPTIVE LEARNING RATE OPTIMIZATION (QIALRO)

TECHNICAL REPORT

Ricard-Santiago Raigada-García*
Department of IT, Multimedia and Telecommunications
Universitat Oberta de Catalunya
Rambla del Poblenou, 156, 08018, Barcelona, Spain.
rraigadag@uoc.edu

March 1, 2024

ABSTRACT

The study proposes an optimization algorithm for machine learning, called Quantum-Inspired Adaptative Learning Rate Optimization (QIALRO), inspired by principles of quantum mechanics. Although the algorithm operates in the classical domain, its conception is based on the ability of quantum systems to simultaneously explore multiple potential states, a property that is sought to be emulated in the optimization of machine learning models. It has been implemented based on softmax regression, cross entropy loss, RMSProp optimizer for learning rate adaptation. The learning rate adjustment mechanism is inspired by the quantum search for optimal solutions by simultaneously exploring multiple possibilities. It adjusts the learning rate by increasing it when the current iteration of the model shows an improvement in loss, leading to an optimal solution space. The technique also permits a decrease in the rate by mimicking the reversion to previous states observed in quantum computing.

Keywords Quantumlike learning · Optimization · Machine Learning · Adaptive Learning Rate · Softmax Regression · Cross-Entropy Loss · RMSProp · Algorithm Convergence · Classical Computing · Multiclass Classification Models

1 Introduction

This work arises from a fascination with quantum mechanics and quantum computing. Inspired by the principles of superposition and entanglement, a simulation of a quantum system is explored through the exploration of multiple simultaneous states Cerezo et al.. It is based on the principles of quantum mechanics to design an algorithm for classical computers. This is what is known as quantumlike learning Ding and Chong [2020]. QIALRO aims to improve the convergence and efficiency of optimization processes in classical computational environments Pure AI Editors.

Machine learning models, especially those dedicated to classification tasks, use gradient-based optimization techniques that aim to minimize error functions and improve predictive accuracy Jiang et al.. One of the most traditionally used methods is Gradient Descent (GD) and its variants IBM. One of the main keys to using this method is choosing the appropriate learning rate so that the algorithm does not converge to a local minimum and can adjust the curvature of the loss surface. Methods such as RMSProp mitigate some problems by allowing individual learning rates for various parameters. There is still a path forward for further improvement in the efficiency of the optimization strategy.

Drawing a parallel between quantum computing and classical computing, where the probability distribution over possible states, allows a quantum algorithm to evaluate many potential outcomes simultaneously. QIALRO proposes dynamic adjustment of the learning rate based on the iterative performance of the model. One of the goals of implementing this approach is to accelerate convergence by adapting the learning rate in response to the loss landscape. Additionally, it introduces a level of exploration and exploitation inspired by quantum computing. QIALRO allows you to replicate the

*https://thedatascientist.digital/

quantum behavior of collapsing a state, through increasing or decreasing the learning rate towards a more favorable state. So, potentially avoiding the stagnation of gradient descent-based methods.

This work presents the theoretical foundations of the QIALRO algorithm, detailing the mathematical formulations of softmax regression, cross entropy loss, the RMSProp optimizer and the new proposed mechanisms. Furthermore, an empirical performance evaluation is carried out with synthetic data sets, comparing their efficiency and convergence rates with traditional gradient descent methods. The main objective of this research is to demonstrate the potential of QIALRO as a tool for machine learning optimization, specifically, in classification problems, offering an algorithm based on quantum mechanics within classical computing.

2 Theoretical fundament

Los métodos de aprendizaje automático supervisado se enfocan en predecir la etqueta de clase para cada instancia de entrada basándose en el entrenamiento previo con un conjunto de datos etiquetado. La etiqueta se llama ouput o response, en estadística clásica corresponde a la variable dependiente Wittek [2014].

La regresión softmax es una generalización de la regresión logística para el caso multiclase. Dado un conjunto de entrada X, la regresión softmax modela la probabilidad de que cada entrada pertenezca a una clase k a través de la función softmax:

$$P(y = k|X) = \frac{e^{W_k \cdot X}}{\sum_{j=1}^{K} e^{W_j \cdot X}} \tag{1}$$

donde W_k son los pesos asociados con la clase k, y K es el número total de clases.

La función de pérdida de entropía cruzada se utiliza para medir el rendimiento del modelo de clasificación, cuyo objetivo es minimizar la diferencia entre las distribuciones de probabilidad predichas y reales:

$$L(W) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(P(y = k|X_i)) \tag{2}$$

donde N es el número de muestras, y_{ik} es un indicador (0 o 1) si la clase k es la correcta para la i-ésima observación, y $P(y = k|X_i)$ es la probabilidad predicha de que la i-ésima observación pertenezca a la clase k.

3 Proposed method

A synthetic dataset has been generated with scikit-learn.

```
X, y = make_classification(
    n_samples=1000,
    n_features=20,
    n_informative=15,
    n_classes=3,
    n_clusters_per_class=1,
    random_state=42
    )
```

One-Hot Encoding has been applied. This process converts the categorical labels y into a vector format, where each label is represented as a binary vector of length equal to the number of classes. The value 1 indicates that it belongs to the class, and 0 otherwise. Additionally, a bias column has been added. This makes it possible for the linear regression model to include intercept without needing to treat it as a special case. It is important because it allows you to capture the variance in the data that is not explained by the characteristics. $\hat{X} = [1, X]$, where $\hat{X}$ is the extended feature matrix with the bias column.

Subsequently, the softmax function defined as follows has been implemented:

$$P(y = k|X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{3}$$

Where $z_k = W_k \cdot X$ is the score for class k, and $P(y = k|X)$ is the probability that the sample X belongs to class k. It is a generalization of the logistic function applied to multiple classes. The logits of the classes are converted into probabilities by exponentiating and normalizing each logit.

The softmax cross-entropy loss function has been implemented as follows:

$$L(W) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(P(y = k|X_i)) \tag{4}$$

Where N is the number of samples, K is the number of classes, y_{ik} is 1 if the sample i belongs to class k and 0 otherwise, and $P(y = k|X_i)$ is the probability that sample i belongs to class k calculated with the softmax function. The cross entropy function measures the performance of the model based on the interpretation of the previous function. The objective is to minimize the loss function in a way that maximizes the probability of success. The reference metric is precision; therefore, it is the metric that is intended to be maximized.

The softmax cross entropy loss gradient defined by:

$$\nabla_{W_k} L = \frac{1}{N}\sum_{i=1}^{N}(P(y = k|X_i) - y_{ik})X_i \tag{5}$$

Where $\nabla_{W_k} L$ is the gradient of the loss function with respect to the weights W_k of the class k. This gradient indicates how the weights should be adjusted to reduce the loss.

The RMSProp optimizer adaptively adjusts the learning rate for each parameter, smoothing out updates and allowing the use of a higher learning rate without divergence. It allows finer adjustments in flat areas of the cost function and larger steps in steep areas. The weight's update is done as:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{6}$$

Where W_t are the weights at time t, η is the learning rate, $E[g^2]_t$ is the moving average of the square of the gradients, g_t is the gradient at time t, and ϵ is a small number to avoid division by zero.

The QIALRO algorithm proposes an iterative optimization where the learning rate is dynamically adjusted based on the progress of the loss. The "measurement and correction" process is simulated in quantum algorithms. In the event that the loss decreases, the learning rate is increased in order to accelerate the exploration process in the solution space. In the case that the loss increases, then the learning rate is reduced so that the changes in the weights are reversed; simulated quantum reversibility. To achieve this, the comparison of the loss before and after updating the weights and adjusting the learning rate is carried out:

Algorithm 1 Quantum-Inspired Adaptive Learning Rate Optimization (QIALRO)

Require: Dataset X, labels y, initial learning rate η, number of iterations, RMSProp parameters.
Ensure: Optimized weights W, loss history.
Preprocessing:
Encode y in one-hot format, split into training and test sets, add bias column.
Optimization:
Initialize W randomly.
for $t = 1$ **to** number of iterations **do**
    Compute gradient g_t using softmax gradient.
    Update W using RMSProp.
    Adjust learning rate based on loss evolution.
end for

```
def oisc_simulated(
            self,
            initial_learning_rate,
            iterations,
            beta_increase=1.05,
            beta_decrease=0.5,
```

```
            tolerance=1e-4):
        num_features = self.X_train_bias.shape[1]
        num_classes = self.y_train.shape[1]
        W = np.random.randn(num_features, num_classes) * 0.01
        learning_rate = initial_learning_rate
        losses = []
        convergence_steps = 0

        for i in range(iterations):
            grad = self.softmax_gradient(W, self.X_train_bias, self.y_train)
            W_prev = W.copy()
            loss_prev = self.softmax_loss(W, self.X_train_bias, self.y_train)

            W, _ = self.rmsprop_optimizer(W, learning_rate, 1)

            loss = self.softmax_loss(W, self.X_train_bias, self.y_train)
            losses.append(loss)

            # Verify the convergence criterion based on the change of the
            # loss
            if np.abs(loss_prev - loss) < tolerance:
                convergence_steps = i + 1
                print(f"Converged at iteration {convergence_steps}")
                break  # Exit the loop if the convergence criterion is met

            if loss < loss_prev:
                learning_rate *= beta_increase
            else:
                W = W_prev
                learning_rate *= beta_decrease

        self.W_optimized = W
        self.losses = losses
        return convergence_steps
```

QIALRO offers a more dynamic approach to learning rate tuning than the SGD optimizer. QIALRO adapts the rate based on the actual performance of the model during training and not through manual adjustment. So it theoretically has a faster and more efficient convergence, especially in cases where the loss function is complex.

4 Experiments and Results

The methodology included the generation of a synthetic data set with standard data preprocessing techniques, as well as the implementation of the optimization algorithms. The computational complexity has been measured in terms of execution time in seconds, and the convergence efficiency was evaluated according to the number of iterations required to reach the convergence criterion.

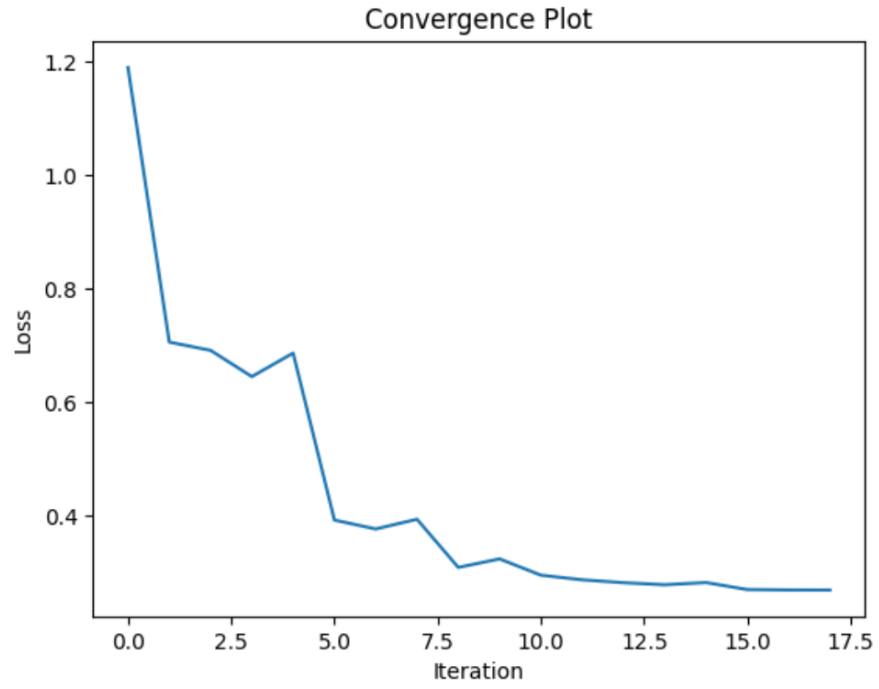

Figure 1: Graph: 1

The QIALRO convergence graph shows the decreasing loss function in each iteration, indicating that the algorithm converges and improves in the optimization task. After 10 iterations, the loss stabilizes, suggesting that the algorithm converges. An accuracy of 89.5 % is achieved after 18 iterations.

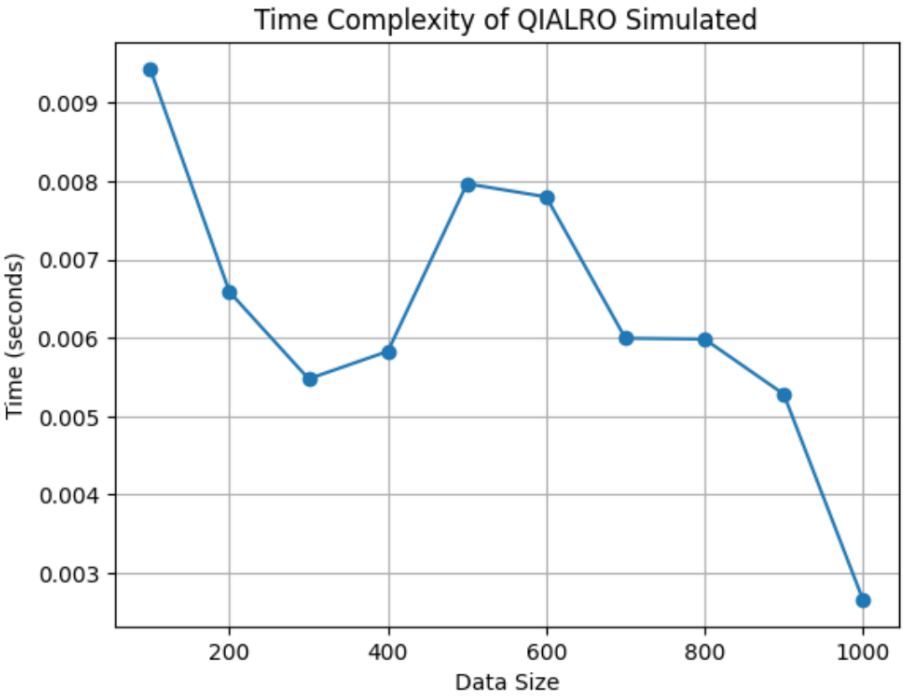

Figure 2: Graph: 2

The complexity graph shows how the execution time changes based on different data sizes. The execution time decreases as the size of the data set increases.

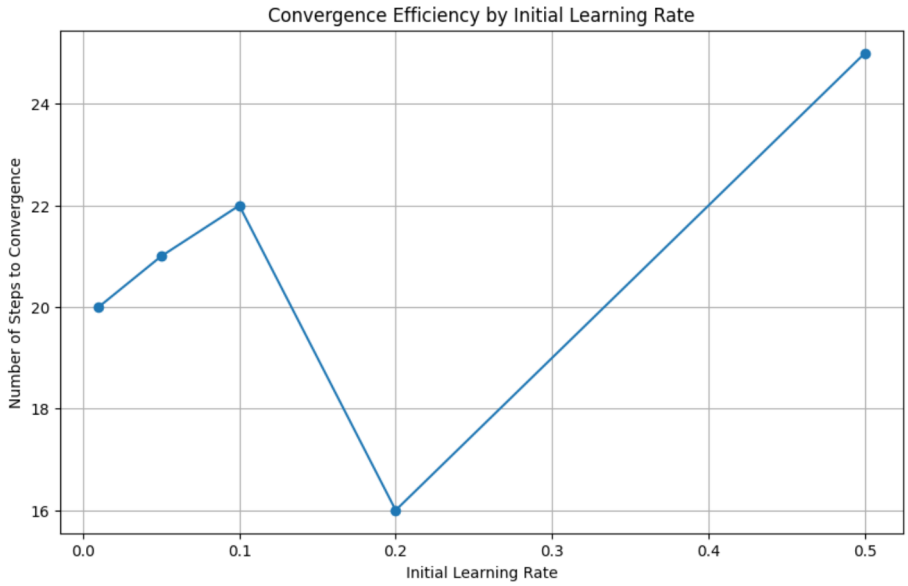

Figure 3: Graph: 3

This graph shows the number of steps required to reach convergence in relation to different initial learning rates. It is highlighted that in principle there is an optimal learning rate so that the number of steps to converge is minimized. Particularly, it is interesting to initialize the optimizer with the learning rate of 0.2.

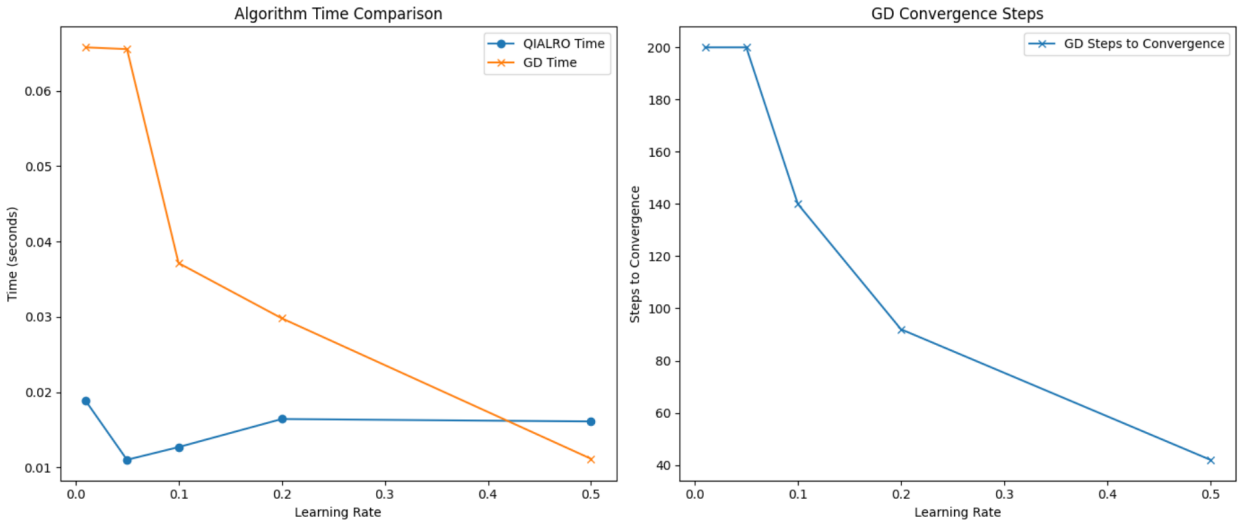

Figure 4: Graph: 4

Chart 4 compares QIALRO with traditional gradient descent in terms of execution time and number of steps to converge. It is highlighted that QIALRO is faster and more efficient in terms of steps to achieve convergence, especially at higher learning rates. It is suggested that dynamic adjustment of the learning rate is effective in navigating the optimization function.

The improvement in performance, speed, and efficiency is crucial in machine learning applications, where training time and computational resources entail a computational and monetary cost, in addition to being limited.

5 Conclusions and Future Work

This research shows the potential of QIALRO as a novel tool for optimization in classification tasks in machine learning. Despite this, there are still difficulties, such as the choice of the initial learning rate, which is still a challenge, since if it is chosen incorrectly, it leads to suboptimal efficiency, despite its ability to dynamically adjust. Despite this, QIALRO presents a promising approach to dynamically adjust the learning rate based on the actual performance of the model during training.

In future work, the challenge will be to overcome the choice of the initial learning rate, potentially aiming at a meta-learning strategy that can inform said choice automatically.

References

M. Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Łukasz Cincio, and Patrick J. Coles. Challenges and opportunities in quantum machine learning. 2(9):567–576. doi:10.1038/s43588-022-00311-3. URL https://doi.org/10.1038/s43588-022-00311-3.

Yongshan Ding and Frederic T. Chong. *Quantum Computer Systems*. Springer International Publishing, 2020. ISBN 978-3-031-00637-1. doi:10.1007/978-3-031-01765-0.

Pure AI Editors. Researchers explore quantum-inspired optimization – pure ai. URL https://pureai.com/articles/2021/12/01/quantum-inspired-optimization.aspx.

Mingxin Jiang, Keyi Shan, Chuansong He, and Can Li. Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. 14(1). doi:10.1038/s41467-023-41647-2. URL https://doi.org/10.1038/s41467-023-41647-2.

IBM. What is gradient descent? | ibm. URL https://www.ibm.com/topics/gradient-descent.

Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier, 8 2014. ISBN 9780128009536. doi:10.1016/C2013-0-19170-2.