Improving Gradient Descent Optimization Algorithm using Inverse Model-Parameter Difference

Author Name
Affiliation

Abstract

Stochastic gradient descent (SGD) plays a central role in training neural networks. Several adaptive optimization algorithms based on SGD exist, such as SGD with Momentum, AdaGrad, Adam, etc., that enhance training by utilizing an accumulation of the past loss function gradients. Despite their prevalence, each of these algorithms works well on some datasets while performing poorly on others. The poor performance of these algorithms can be attributed to undesired learning-rates due to extreme values of accumulated gradients. Hence, there is a need for a new optimization approach. This paper presents a new class of adaptive SGD algorithm that computes the adaptive learning-rate for each model-parameter as an inverse function of the displacement of the model-parameter from the preceding iterations. Theoretical analysis was conducted to verify the convergence of the proposed algorithm. Additionally, experiments were performed on multiple benchmark datasets to demonstrate the advantages of the proposed algorithm. For instance, on the test data of the ImageNet dataset, the proposed algorithm achieved a 3.85% improvement in accuracy over SGD with momentum and a 1.23% improvement over the Adam algorithm, simultaneously with faster convergence.

1 Introduction

Machine learning has undergone tremendous progress over the years and finds application in an increasing number of domains, including computer vision, audio processing, and natural language processing [Dong *et al.*, 2016; Simonyan and Zisserman, 2015; Dong *et al.*, 2014]. Training a machine learning model essentially involves solving a minimization problem, where the model-parameters are iteratively adjusted to minimize a loss or a cost function by means of an optimization algorithm. The stochastic gradient descent (SGD) is a widely popular optimization algorithm used in machine learning. SGD iteratively updates the model-parameters, starting from some initial values, in the direction of the negative gradient of the loss function until the loss is minimized. The parameter learning-rate determines the steps with which the model-parameters are updated [Agarwal *et al.*, 2012; Nemirovski *et al.*, 2009]. SGD has been extensively researched over the years and many algorithms based on SGD have been developed to improve the training of neural networks. These can be categorized as (i) the algorithms which accelerate the gradient descent using an accumulation of past gradients, e.g., SGD with Momentum, Nesterov's accelerated gradient (NAG), etc., [Qian, 1999; Botev *et al.*, 2017]; (ii) the algorithms that compute adaptive learning-rate for each model-parameter using the squared sum (or exponentially decaying squared sum) of the past gradients, e.g., AdaGrad, RMSProp, Adadelta, etc. [Duchi *et al.*, 2011; Tieleman and Hinton, 2012; Zeiler, 2012]; and (iii) the algorithms which combine the above-mentioned momentum methods and the adaptive learning-rate methods, e.g., Adam, AMSGrad, AdamW, etc., [Kingma and Ba, 2015; Reddi *et al.*, 2018; Loshchilov and Hutter, 2019]. The Adam algorithm and its variants, because they are robust and less sensitive to hyperparameter tuning, are preferred optimizers for training large neural network models and language processing models, like ImageNet, BERT, etc. [Devlin *et al.*, 2019; Xu *et al.*, 2015]. On the other hand, the SGD with momentum algorithm shows comparatively improved performance over the Adam algorithm in many scenarios like computer vision and Speech Recognition [Gupta *et al.*, 2021; Keskar and Socher, 2017].

Despite their widespread use, algorithms such as SGD with momentum, and Adam often exhibit poor performance due to the uneven scaling of past gradients caused by unstable and extreme values of the adaptive learning-rate [Wilson *et al.*, 2017; Zhou *et al.*, 2020]. This has spurred the need for the development of new algorithms. This paper proposes a new class of adaptive SGD optimization that dynamically adapts the learning-rate for each model-parameter, inversely proportional to the difference in the model-parameter value from the past iterations. The proposed algorithm incorporates knowledge of the trajectories of the model-parameters to compute the corresponding learning-rate. Specifically, for a model-parameter that is updated with a larger displacement along a steeper dimension of the loss function, the algorithm lowers the learning-rate. On the other hand, for a model-parameter updated with a smaller displacement, i.e., on a flatter dimension of the loss function, the algorithm generates a larger learning-rate.

To evaluate the proposed algorithm further, a convergence analysis based on the regret bound approach was performed verifying that the proposed algorithm converges with a rate of $\mathcal{O}(\frac{1}{T})$ for both the convex and the non-convex loss functions. Also, experimental analysis was carried out by implementing the proposed algorithm for training standard datasets like the CIFAR-10, CIFAR-100, and ImageNet datasets, and the performance of the proposed algorithm is compared with the currently popular optimization algorithms. The major contribution of the paper is a new class of adaptive SGD optimization algorithm, which updates the learning-rate for each model-parameter dynamically, inversely proportional to the displacement of the model-parameters from the preceding iterations.

The remainder of the paper is organized as follows: In Section 2, the proposed algorithm is presented, followed by a theoretical convergence analysis in Section 3. Section 4 details the experiment results evaluating the performance of the proposed algorithm using benchmark datasets. Following this, a conclusion and a brief outlook on the future aspects of the work are given in Section 5.

2 Proposed algorithm

The proposed algorithm is introduced and discussed in detail in this section. As mentioned in the previous section, popular optimization algorithms such as SGD with momentum, and Adam use an accumulation of past gradients to update model-parameters, but often suffer from poor convergence due to undesired values of the adaptive learning-rate. To address these limitations, this paper proposes a new class of adaptive SGD algorithm that computes the learning-rate for each model-parameter as a function of its displacement. Specifically, the algorithm computes the learning-rate for each model-parameter inversely proportional to the squared difference in the values of that parameter from the immediate preceding iterations. The algorithm adjusts the individual learning-rate of model-parameters by (a) lowering the learning-rate for parameters along steeper regions of the loss function to avoid overshoots or divergent learning because of large gradient values, and (b) increasing the learning-rate for parameters along flatter regions of the loss function so that learning does not get stuck due to an accumulation of small gradient values.

The update of the model-parameters $\boldsymbol{\theta}_k$, with the adaptive learning-rate $\boldsymbol{\eta}_k$, is given by the following equation

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k g(\boldsymbol{\theta}_k, \xi) + \beta(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \qquad (1)$$

where $\boldsymbol{\eta}_k = diag\left[\eta_k^{(1)}, \eta_k^{(2)}, \ldots, \eta_k^{(d)}\right]$ is the diagonal learning-rate matrix, d is the number of model-parameters, $\eta_k^{(i)}$, with $i = 1, 2, \ldots, d$, represents the learning-rate corresponding to each model-parameter, $g(\boldsymbol{\theta}_k, \xi)$ represents the stochastic gradient of the loss function w.r.t the model-parameters $(\boldsymbol{\theta}_k)$ and a random variable ξ, and $\beta \in (0, 1]$ is the parameter which determines the contribution of the previous model-parameter difference.

Table 1 Proposed algorithm

Input:
initial model-parameters $\boldsymbol{\theta}_0$
initial learning-rate η_0
parameter β

Iterate:

1: **for** $k = 1, \ldots, T$ **do**
2: compute the gradient: $g(\boldsymbol{\theta}_k, \xi)$
3: compute the adaptive learning-rate:
$\boldsymbol{\eta}_k = diag\left[\eta_k^{(1)}, \eta_k^{(2)}, \ldots, \eta_k^{(d)}\right]$
where $\eta_k^{(i)} = \frac{\alpha}{\left(1+\|\Delta\theta_k^{(i)}\|^2\right)^{\frac{1}{2}}}$
4: update model-parameters:
$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k g(\boldsymbol{\theta}_k, \xi) + \beta(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$
5: **end for**
6: **return** solution

The adaptive learning-rate $\eta_k^{(i)}$ for the i^{th} model-parameter, is defined as a function of the squared difference of its values from the immediate preceding iterations, and is given by

$$\eta_k^{(i)} = \frac{\alpha}{\left(1+ \parallel \Delta\theta_k^{(i)} \parallel^2\right)^{\frac{1}{2}}} \qquad (2)$$

where $\Delta\theta_k^{(i)} = \theta_k^{(i)} - \theta_{k-1}^{(i)}$ gives the difference in the values of the i^{th} model-parameter between the k^{th} and the $(k-1)^{th}$ iterations. The parameter α is given by $\alpha = \eta_0\alpha_0$, where η_0 is the initial learning-rate and $\alpha_0 > 0$ is a constant. The framework of the proposed algorithm is described in the Table 1.

The algorithm improves convergence by controlling the learning-rate for each model-parameter according to an inverse relation with the change in model-parameter values. The parameter α determines the range of the learning-rate $\eta_k^{(i)}$ and its influence on the training behaviour of the algorithm. Figure 1 shows the range of values of the learning-rate $\eta_k^{(i)}$ plotted with respect to the values of $\Delta\theta_k^{(i)}$. It can be observed from the plot that a larger change in the values of the model-parameter results in a smaller value of the adaptive learning-rate. Similarly, a smaller change in model-parameter values leads to a larger value of the learning-rate. The maximum value of $\eta_k^{(i)}$, i.e., for $\Delta\theta_k^{(i)} = 0$ is given by $\eta_k^{(i)} = \alpha$. The impact of different values of α_0 is analyzed in the experimental section, and the convergence of the algorithm is mathematically established in the theoretical analysis section.

It is worth mentioning that the limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm, which belongs to the category of quasi-Newton methods, is known to use the history of past model-parameter and their gradients to estimate the Hessian of the objective function $H\left(\boldsymbol{\theta}_k\right) = \nabla^2 J\left(\boldsymbol{\theta}_k\right)$ [Liu and Nocedal, 1989; Kochenderfer and Wheeler, 2019]. In contrast, the proposed algorithm utilizes the difference in model-parameters to update the

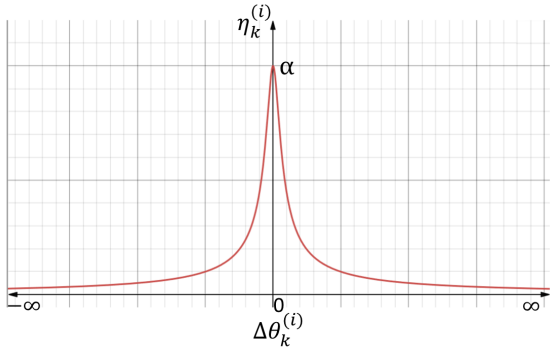

Figure 1: Range of values of the adaptive learning-rate $\eta_k^{(i)}$ with respect to $\Delta\theta_k^{(i)}$

learning-rate of the SGD algorithm. The difference between these two methods is in-fact also obvious from their compexities. The proposed algorithm has a computational complexity and the storage capacity of $\mathcal{O}(n)$, whereas the L-BFGS algorithm requires computational costs and storage capacity of the order $\mathcal{O}(nd)$, where d is the number of model parameters and n is the memory size of the algorithm [Bottou *et al.*, 2018; Nocedal and Wright, 2006].[1]

3 Convergence analysis

In this section, the convergence analysis of the proposed algorithm is presented. To facilitate a clear presentation of the analysis, the symbols and notations used in the paper are described first. Given a vector $\boldsymbol{p} \in \mathbb{R}^n$, $p_k^{(i)}$ denotes the i^{th} element of the vector $\boldsymbol{p}$ at instance k. $||\boldsymbol{p}||^2$ gives the squared norm of the vector. $\langle \boldsymbol{p}, \boldsymbol{q} \rangle$ represents the dot product of the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, and $\frac{\boldsymbol{p}}{\boldsymbol{q}}$ denotes the element-wise division of vectors $\boldsymbol{p}$ and $\boldsymbol{q}$.

Further in this section, the convergence of the proposed algorithm in terms of the regret bound for both convex and non-convex loss functions is analyzed. The regret bound is one of the basic criteria to evaluate the performance of any optimization algorithm. The regret bound, denoted by $R_J(T)$, measures the difference between the cumulative loss function values at any instance k and the optimum loss, and is defined as

$$R_J(T) = \sum_{k=0}^{T} \left[J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}^*) \right] \tag{3}$$

where $J(\boldsymbol{\theta}^*) = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the minimum of the loss function for the optimum model-parameters $\boldsymbol{\theta}^*$, $J(\boldsymbol{\theta}_k)$ is the loss function at instance k, and T is the number of iterations of the algorithm.

[1]The order of compuational complexitx refers to the number of model parameters and the number of basic operations (such as additions, multiplications, etc.) required by the algorithm, and storage refers to the amount of memory required to store the intermediate variables.

The convergence analysis for the proposed algorithm proves that the regret bound in case of both convex and non-convex loss functions is bounded by a constant, and the rate of convergence is bounded by the inverse of the number of iterations.

The following assumptions are made for the convergence analysis.

Assumption 1. *For a smooth function $J(\boldsymbol{\theta})$ which is twice continuously differentiable with L-Lipschitz continuous gradients, there exists for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$ [Boyd and Vandenberghe, 2004],*

$$\| \nabla J(\boldsymbol{\theta}_1) - \nabla J(\boldsymbol{\theta}_2) \| \leq L \| \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2 \| \tag{4}$$

and,

$$J(\boldsymbol{\theta}_2) \leq J(\boldsymbol{\theta}_1) + \langle J(\boldsymbol{\theta}_1), (\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1) \rangle + \frac{L}{2} \| \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \|^2 \tag{5}$$

This implies that the function is differentiable at all points in its domain and the derivative of the function is continuous.

Assumption 2. *The unbiased and independent stochastic gradient of the loss function $g(\boldsymbol{\theta}_k, \xi)$ at any instance k is given by [Ghadimi* et al.*, 2015]*

$$E\left[g\left(\boldsymbol{\theta}_k, \xi \right) \right] = \nabla J\left(\boldsymbol{\theta}_k \right) \tag{6}$$

where $\nabla J\left(\boldsymbol{\theta}_k \right)$ is the deterministic gradient of the loss function with respect to $\boldsymbol{\theta}_k$, and ξ is a random variable stochastically independent of the parameter $\boldsymbol{\theta_k}$. Further in the paper, the stochastic gradient is denoted as $g_k := g(\boldsymbol{\theta}_k, \xi)$.

Assumption 3. *At each iteration k, the expected value of the adaptive learning-rate $E\left[\eta_k^{(i)}\right]$, and the expected value of the squared learning-rate $E\left[\left(\eta_k^{(i)}\right)^2\right]$ are bounded by constants c_1 and c_2 respectively. i.e.*

$$E\left[\eta_k^{(i)}\right] = E\left[\frac{\alpha}{\left(1 + \| \Delta\theta_k^{(i)} \|^2\right)^{\frac{1}{2}}} \right] \leq c_1$$

and

$$E\left[\left(\eta_k^{(i)}\right)^2\right] = E\left[\frac{(\alpha)^2}{\left(1 + \| \Delta\theta_k^{(i)} \|^2\right)} \right] \leq c_2$$

The assumption on the bounds of $\eta_k^{(i)}$ is important for proving the convergence and stability of the proposed algorithm.

Next, the results of the convergence analysis of the proposed algorithm for both convex and non-convex loss functions are presented.

Convergence for a convex function

Theorem 1 (Proof in appendix A.2)**.** *For a convex and twice differentiable loss function $J\left(\boldsymbol{\theta}\right) : \mathbb{R}^d \to \mathbb{R}$, whose gradient is L-Lipschitz (satisfying assumptions $1 - 3$), the proposed algorithm satisfies the regret bound given by*

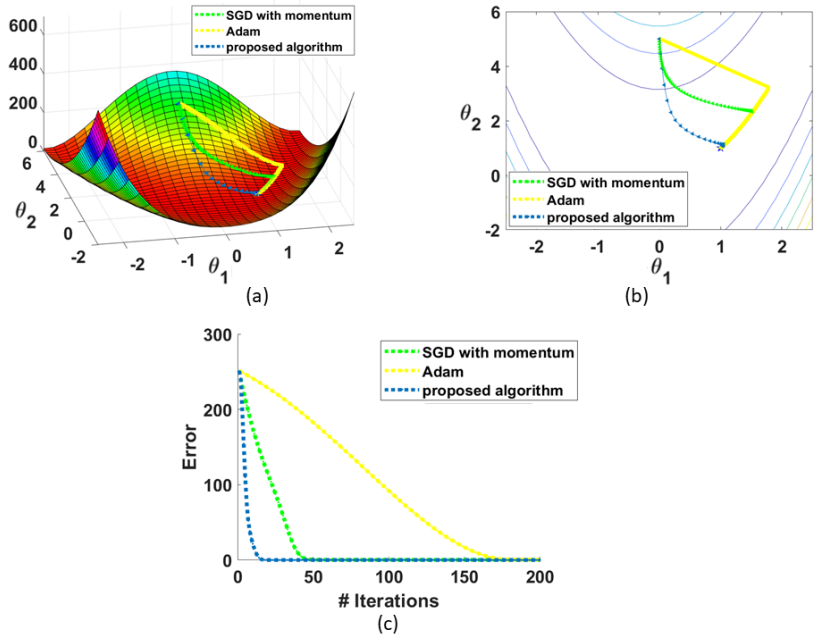

Figure 2: Comparison of the proposed algorithm with SGD with momentum, and Adam algorithms on Rosenbrock function: (a) 3D Surface plot of convergence. (b) 2D contour plot of the convergence. (c) Convergence of the function w.r.t. number of iterations.

$$R_J(T) = J(\bar{\boldsymbol{\theta}}_T) - J(\boldsymbol{\theta}^*) \leq \frac{1}{T}\left[\frac{2(1-\beta)^2 + L\beta \boldsymbol{C}_1}{2\boldsymbol{C}}\right]\left(\| \boldsymbol{\theta}_0 - \boldsymbol{\theta}^* \|_2^2\right) \quad (7)$$

where $J\left(\bar{\boldsymbol{\theta}}_T\right) = \sum_{k=0}^{T}\left(J\left(\boldsymbol{\theta}_k\right)\right)$, $\boldsymbol{C}$ and $\boldsymbol{C}_1$ are positive constants defined during the proof, and T is the number of iterations of the adaptive algorithm. An $\mathcal{O}\left(1/T\right)$ regret bound guarantees that the algorithm converges to the optimal weight vector, and the rate of convergence is proportional to $1/T$. This implies that as the number of iterations increases, the algorithm approaches an optimum solution.

Convergence for a non-convex function

Theorem 2 (Proof in appendix A.2)**.** *Given a smooth non-convex loss function $J\left(\boldsymbol{\theta}\right) : \mathbb{R}^d \rightarrow \mathbb{R}$ (satisfying assumptions $1-3$), the gradient of the loss function for the proposed algorithm satisfies*

$$\min_{k=0\ldots T-1} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq \frac{1}{\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}}\frac{1}{T}E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \quad (8)$$

where $\bar{\boldsymbol{C}} \geq \bar{\boldsymbol{D}} \geq 0$ are constants defined during the proof, and T is the maximum number of iterations. The equation gives a bound on the minimum expected squared norm of the gradient of the loss function over T iterations. The theorem implies that as the number of iterations increases, the gradient of the loss function approaches the minimum, and the algorithm converges to the optimal solution.

4 Experiments

In this section, an extensive experimental analysis was carried out to evaluate the efficacy and performance of the proposed algorithm. The performance of the proposed algorithm was compared with a number of state-of-the-art optimizers, such as SGD with momentum, Adam, AdaBelief, LookAhead, and AdamW.

The first part of the experiments involved implementing the proposed algorithm on a standard two-parameter loss function, i.e., the Rosenbrock function, to visualize and compare its convergence with SGD with momentum, and Adam algorithm. In the next part, further experiments were carried out by implementing the above-mentioned optimizers for training the following benchmark datasets, i.e., (i) CIFAR 10 and CIFAR 100 datasets using the ResNet18 architecture, and (ii) ImageNet dataset using the EfficientNet architecture.

4.1 Performance assessment on Rosenbrock function

For the experiments in this section, the above-mentioned algorithms were implemented on a two-dimensional optimization function and their convergence were compared. The Rosenbrock function is often used as a benchmark for evaluating different optimization algorithms [Emiola and Adem, 2021]. The Rosenbrock function for a two-parameter setup is given by

$$f\left(\theta_1, \theta_2\right) = \kappa\left(\theta_1^2 - \theta_2\right)^2 + \left(\theta_1 - 1\right)^2 \quad (9)$$

where θ_1 and θ_2 are real-valued parameters, and the constant κ determines the steepness of the valley of the Rosenbrock function. Figure 2(a) shows the 3D surface plot of the Rosenbrock function with $\kappa = 100$. The function has a minimum

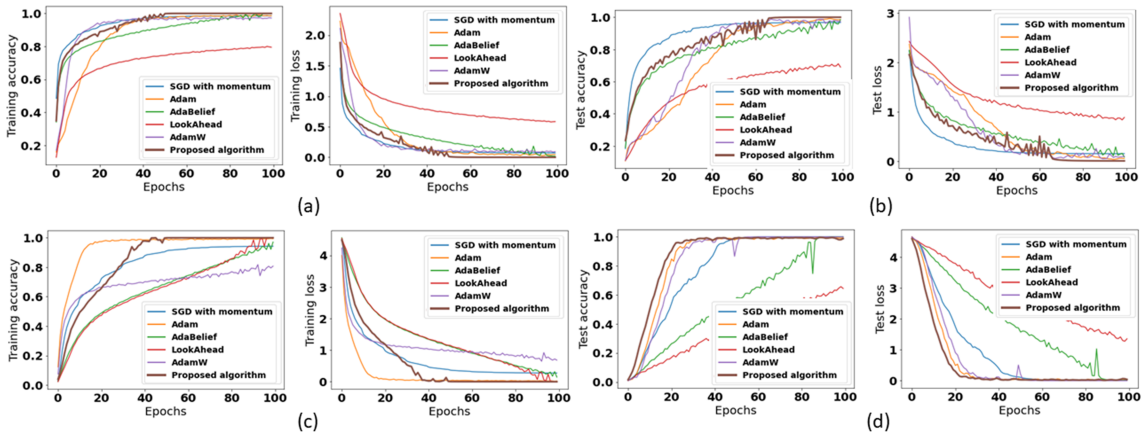

Figure 3: Performance comparison of different optimization algorithms: (a) Training accuracy and training loss, and (b) Test accuracy and test loss on the CIFAR-10 dataset. (c) Training accuracy and training loss, and (d) Test accuracy and test loss on the CIFAR-100 dataset.

Optimizer	Dataset							
	CIFAR 10				CIFAR 100			
	Training		Test		Training		Test	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
SGD with momentum	0.9854	0.0730	0.7431	0.9625	0.9424	0.2746	0.8578	0.8105
Adam	0.9887	0.0356	0.9877	0.0863	0.9938	0.0190	0.9843	0.0513
AdaBelief	0.9958	0.0255	0.7486	0.9996	0.8916	0.3316	0.3797	6.0604
LookAhead	0.8022	0.5671	0.6554	0.9911	0.9992	0.0096	0.3592	3.0982
AdamW	0.9754	0.0831	0.9750	0.0892	0.7876	0.7899	0.9798	0.0598
Proposed algorithm	**1.0**	**0.0012**	**0.9897**	**0.0856**	**0.9996**	**0.0053**	**0.9943**	**0.0213**

Table 2: Performance comparison of different optimization methods applied on the CIFAR-10 and CIFAR-100 datasets.

at $(1, 1)$, along a long, parabolic-shaped flat valley, making it difficult for some optimization algorithms to converge quickly.

The experiment was set up as follows: The model-parameters θ_1 and θ_2 were initialized to $(0, 5)$ and the initial learning-rate was set to $\eta_0 = 0.001$. The hyperparameters were set to $\beta = 0.9$ for the momentum optimizer, and $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the Adam optimizer respectively.

Figure 2 shows the convergence comparison of the proposed algorithm with the SGD with momentum and the Adam optimizer. Figures 2(a) and 2(b) show the 3D surface plot and the contour plot of the convergence of the algorithms respectively, while 2(c) shows the convergence against the number of iterations. It can be observed from the plot 2(c) that the proposed algorithm converged to the minimum in about 25 iterations. In comparison, the SGD with momentum algorithm converged in about 40 iterations, while the Adam algorithm showed the slowest convergence for this experiment and required about 160 iterations to converge to the minimum. These results indicate that the proposed algorithm outperforms the SGD with momentum and Adam algorithms in terms of the number of iterations required for convergence.

4.2 Performance Assessment on Neural Networks

CIFAR-10 and CIFAR-100

In this section, the performance of the proposed optimization algorithm is compared with different state-of-the-art optimizers for image classification tasks on the standard CIFAR-10 and CIFAR-100 datasets using the ResNet-18 architecture [He *et al.*, 2017]. The datasets comprises of 32×32 color images with 10 and 100 classes, respectively. The dataset is split into 50,000 images for training and 10,000 images for testing [Krizhevsky *et al.*, 2017].

The SGD with momentum optimizer was implemented with a step-decay scheduler, starting with an initial learning rate of 0.001 and decreased by a factor of 10 after every 50 epochs. For the Adam and AdaBelief optimizers, the initial learning rate was set to 0.001, with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.999$ [Zhuang *et al.*, 2020]. The LookAhead optimizer was implemented with a learning rate of 0.001 and the hyperparameters $\kappa = 5$ and $\alpha = 0.5$ [Zhang *et al.*, 2019]. The AdamW optimizer was applied with a learning rate of 0.001, hyperparameters $\beta_1 = 0.9, \beta_2 = 0.999$, and a weight decay of 2.5×10^{-4} [Loshchilov and Hutter, 2019]. For the proposed algorithm, the initial learning rate was set at 0.001, and the parameter $\alpha_0 = 1$ was chosen based on extensive simulations with different parameter values (refer to the fol-

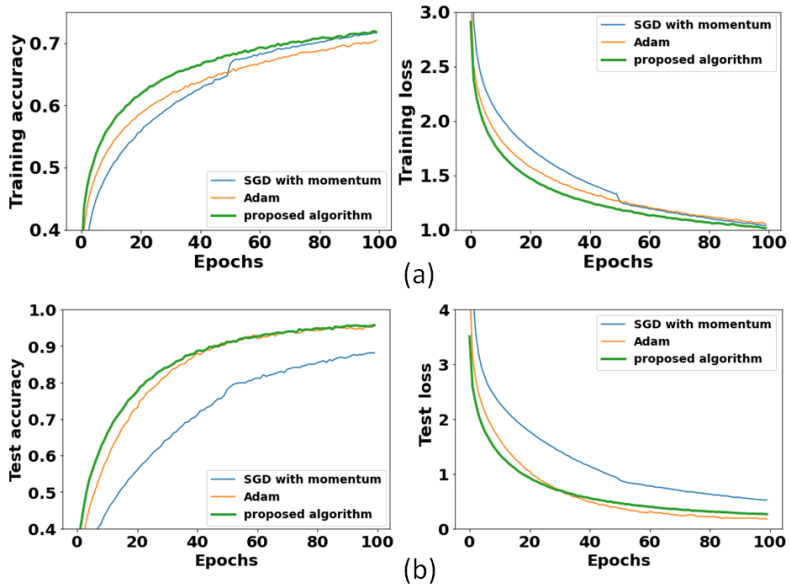

Figure 4: Performance comparison of different optimization algorithms on the ImageNet dataset: (a) Training accuracy and Training loss. (b) Test accuracy and Test loss.

Optimizer	ImageNet			
	Training		Test	
	Accuracy	Loss	Accuracy	Loss
SGD with momentum	0.8156	0.4205	0.7675	0.3698
Adam	0.8279	0.3642	0.7881	0.2233
proposed algorithm	**0.8533**	**0.3183**	**0.7983**	**0.2121**

Table 3: Performance comparison of different optimization algorithms on the ImageNet dataset

lowing subsection). The dataset were trained on the Google Colaboratory environment using a 25 GB RAM GPU for 100 epochs with a batch size of 128.

Figures 3(a) and (b) show the results on the training and test datasets for the CIFAR-10 dataset, while Figures 3(c) and (d) show the corresponding results for the CIFAR-100 dataset. While all algorithms achieved high accuracy levels, with values approaching 99%, the proposed algorithm outperforms the other algorithms in both datasets. The performance evaluation based on the accuracy and loss for training and test data for each optimizer is compiled in Table 2.

ImageNet

For the next experiments, the ImageNet dataset was used and the training was carried out on the EfficientNetB3 architecture [Deng *et al.*, 2009; Tan and Le, 2019]. For this experiment, the simulation comparisons were carried out with SGD with momentum and the Adam algorithm. The hyperparameters for the SGD with momentum (with step-decay), Adam optimizer and the proposed algorithm were the same as in the previous experiments. Figure 4 shows the learning curves for each optimization method for both the training and test datasets. It can be observed that both the Adam and the proposed algorithm perform better than the SGD with momentum, but the proposed algorithm shows slightly better performance than the Adam algorithm. Table 3 summarizes the training and test accuracy and loss for the above mentioned model in case of each optimizer.

Experiments with different hyperparameters

In this subsection, the performance of the proposed algorithm was evaluated for different hyperparameters applied on the CIFAR-10 dataset. The proposed algorithm was implemented for training the dataset for different values of the parameter α_0, i.e., 0.01, 0.1, 1, 5, and 10. It can be observed that for smaller values of the parameter α_0, the algorithms shows slower convergence and results in lower accuracy. However, convergence of the algorithm is fast and aggressive for larger values of α_0. For example, $\alpha_0 = 10$ shows large oscillations at the middle stage of training. Training with parameter $\alpha_0 = 1$ shows the best overall performance. Figure 5 shows the comparisons of the training curves for the different hyperparameters and Table 4 lists the training accuracy and training loss for the same.

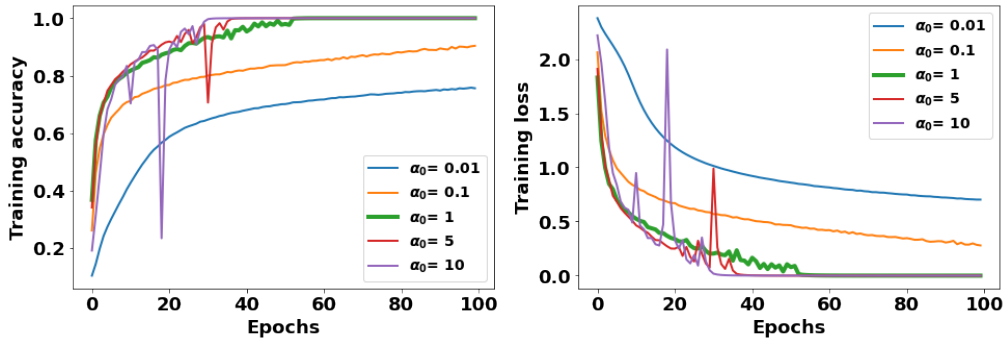

Figure 5: Performance comparison for different values of parameter α_0

Parameter: α_0	Loss	Accuracy
$\alpha_0 = 0.01$	0.71115	0.75323
$\alpha_0 = 0.1$	0.21901	0.92940
$\alpha_0 = 1$	**0.00012**	**1.0**
$\alpha_0 = 5$	0.0010	1.0
$\alpha_0 = 10$	4.35486	1.0

Table 4: Training Loss and accuracy for different values of parameter α_0

5 Conclusion

In this paper, a new approach to gradient descent optimization is presented, which updates the learning-rate of the algorithm for each model-parameter, inversely proportional to the change in model-parameter values from the preceding iteration. The algorithm adapts the model-parameters on gentle slope with larger learning-rate relative to model-parameters on steep slope, thereby improving the overall convergence. The efficacy of the proposed algorithm was proven based on the training of standard datasets like the CIFAR-10/100 and the ImageNet datasets, and it was noted that the proposed algorithm surpasses the currently popular algorithms. Future work involves deeper analysis of the proposed algorithm on complex non-convex loss functions and implementing it on other benchmark datasets and models.

References

[Agarwal *et al.*, 2012] Alekh Agarwal, Peter L. Bartlett, Pradeep Ravikumar, and Martin J. Wainwright. Information-Theoretic Lower Bounds on the Oracle Complexity of Stochastic Convex Optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, May 2012. Conference Name: IEEE Transactions on Information Theory.

[Botev *et al.*, 2017] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1899–1903, May 2017. ISSN: 2161-4407.

[Bottou *et al.*, 2018] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[Boyd and Vandenberghe, 2004] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, 2019.

[Dong *et al.*, 2014] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a Deep Convolutional Network for Image Super-Resolution. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 184–199, Cham, 2014. Springer International Publishing.

[Dong *et al.*, 2016] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, February 2016. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[Emiola and Adem, 2021] Iyanuoluwa Emiola and Robson Adem. Comparison of Minimization Methods for Rosenbrock Functions. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 837–842, June 2021. ISSN: 2473-3504.

[Ghadimi *et al.*, 2015] Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson. Global convergence of the Heavy-ball method for convex optimization. In *2015 European Control Conference (ECC)*, pages 310–315, Linz, Austria, July 2015. IEEE.

[Gupta *et al.*, 2021] Aman Gupta, Rohan Ramanath, Jun Shi, and S Sathiya Keerthi. Adam vs. SGD: Closing the generalization gap on image classification. *OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021.

[He *et al.*, 2017] Ying He, Bin Liang, Yu Zou, Jin He, and Jun Yang. Depth Errors Analysis and Correction for Time-of-Flight (ToF) Cameras. *Sensors*, 17(1):92, January 2017.

[Keskar and Socher, 2017] Nitish Keskar and Richard Socher. Improving Generalization Performance by Switching from Adam to SGD. *CoRR*, December 2017.

[Kingma and Ba, 2015] D. P. Kingma and L. J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015. Publisher: Ithaca, NYarXiv.org.

[Kochenderfer and Wheeler, 2019] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. The MIT Press, 2019.

[Krizhevsky *et al.*, 2017] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[Liu and Nocedal, 1989] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[Loshchilov and Hutter, 2019] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

[Nemirovski *et al.*, 2009] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust Stochastic Approximation Approach to Stochastic Programming. *SIAM Journal on Optimization*, January 2009. Publisher: Society for Industrial and Applied Mathematics.

[Nocedal and Wright, 2006] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, New York, 2nd ed. 2006 edition edition, July 2006.

[Qian, 1999] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 1999.

[Reddi *et al.*, 2018] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, February 2018.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Tan and Le, 2019] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019. ISSN: 2640-3498.

[Tieleman and Hinton, 2012] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[Wilson *et al.*, 2017] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4151–4161, Red Hook, NY, USA, December 2017. Curran Associates Inc.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[Zeiler, 2012] Matthew D Zeiler. Adadelta: an adaptive learning rate method. In *Advances in Neural Information Processing Systems*, pages 685–693. Curran Associates, Inc., 2012.

[Zhang *et al.*, 2019] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead Optimizer: k steps forward, 1 step back. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Zhou *et al.*, 2020] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, and Steven Hoi. Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning. 2020.

[Zhuang *et al.*, 2020] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. *NeurIPS 2020*, December 2020.

A Appendix

A.1 Lemmas

In this section, we present a convergence proof for the optimization algorithm. The following lemmas are defined and proven before proceeding to the main theorem.

Lemma 1. *For a smooth function $J : \mathbb{R}^d \to \mathbb{R}$ which is convex and whose gradient is L-Lipschitz, there exists $\forall \boldsymbol{\theta_k} \in \mathbb{R}^d$, such that*

$$\| \nabla J(\boldsymbol{\theta}_k) \|^2 \leq 2L \left(J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}^*) \right) \tag{10}$$

Proof. This statement is a well-known result in convex optimization known as the descent lemma that states that the value of a smooth function decreases along the direction of the gradient.

For a smooth function,

$$J(\boldsymbol{\theta}_2) \leq J(\boldsymbol{\theta}_1) + \langle \nabla J(\boldsymbol{\theta}_1), (\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1) \rangle + \frac{L}{2} \| \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \|^2 \tag{11}$$

Assume $\boldsymbol{\theta}_1 = \boldsymbol{\theta}, \boldsymbol{\theta}_2 = \boldsymbol{\theta}^*, \boldsymbol{\theta}^* = \boldsymbol{\theta} - \frac{1}{L} \nabla J(\boldsymbol{\theta})$, then

$$J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{\theta}) - \frac{1}{L} \| \nabla J(\boldsymbol{\theta}) \|^2 + \frac{1}{2L} \| \nabla J(\boldsymbol{\theta}) \|^2 \tag{12}$$

$$\frac{1}{2L} \| \nabla J(\boldsymbol{\theta}) \|^2 \leq \left(J(\boldsymbol{\theta}) - J(\boldsymbol{\theta}^*) \right) \tag{13}$$

$$\| \nabla J(\boldsymbol{\theta}) \|^2 \leq 2L \left(J(\boldsymbol{\theta}) - J(\boldsymbol{\theta}^*) \right) \tag{14}$$

□

Lemma 2. *Consider the update equation (Eq. (1)), given by $\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} - \boldsymbol{\eta_k} g_k + \beta (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$.*
Let $\boldsymbol{p}_k = \frac{\beta}{(1-\beta)} (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$, then

$$\boldsymbol{\theta}_{k+1} + \boldsymbol{p}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{p}_k - \frac{\boldsymbol{\eta}_k}{(1-\beta)} g_k \tag{15}$$

Proof.

$$\boldsymbol{p}_k = \frac{\beta}{(1-\beta)} (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \tag{16}$$

$$\boldsymbol{p}_{k+1} = \frac{\beta}{(1-\beta)} (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) \tag{17}$$

$$\boldsymbol{p}_{k+1} + \boldsymbol{\theta}_{k+1} = \frac{\beta}{(1-\beta)} (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) + \boldsymbol{\theta}_{k+1} \tag{18}$$

$$\boldsymbol{p}_{k+1} + \boldsymbol{\theta}_{k+1} = \frac{1}{(1-\beta)} (\boldsymbol{\theta}_{k+1}) - \frac{\beta}{(1-\beta)} \boldsymbol{\theta}_k \tag{19}$$

$$= \frac{1}{(1-\beta)} (\boldsymbol{\theta}_k - \boldsymbol{\eta}_k g_k + \beta (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})) - \frac{\beta}{(1-\beta)} \boldsymbol{\theta}_k \tag{20}$$

$$= \boldsymbol{\theta}_k + \frac{\beta}{(1-\beta)} (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) - \frac{\eta_k}{(1-\beta)} g_k \tag{21}$$

$$\boldsymbol{p}_{k+1} + \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{p}_k - \frac{\boldsymbol{\eta}_k}{(1-\beta)} g_k \tag{22}$$

□

Lemma 3. *Consider a function $J : \mathbb{R}^d \to \mathbb{R}$ whose gradient is L-Lipschitz, and let $\boldsymbol{z}_k = \boldsymbol{\theta}_k + \boldsymbol{p}_k$, then for any $k \geq 0$*

$$E \left[J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k) \right] \leq \frac{1}{2L} E \left[\| \nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \|^2 \right] - \left(\frac{\boldsymbol{C}_1}{(1-\beta)} - \frac{L \boldsymbol{C}_2}{(1-\beta)^2} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \right) \tag{23}$$

where L is the lipschitz constant, β is the decay parameter, and $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are constants.

Proof. Let $\boldsymbol{z}_k = \boldsymbol{\theta}_k + \boldsymbol{p}_k$, and let $\delta_k = g_k - \nabla J(\boldsymbol{\theta}_k)$. Here, $E[g_k] = E[\nabla J(\boldsymbol{\theta}_k)]$, where g_k is an unbiased estimate of $\nabla J(\boldsymbol{\theta}_k)$. Also, this implies $E[\delta_k] = 0$
Next, using the smoothness property:

$$J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k) \leq \langle \nabla J(\boldsymbol{z}_k), (\boldsymbol{z}_{k+1} - \boldsymbol{z}_k) \rangle + \frac{L}{2} \| \boldsymbol{z}_{k+1} - \boldsymbol{z}_k \|^2 \tag{24}$$

Since $\boldsymbol{z}_k = \boldsymbol{\theta}_k + \boldsymbol{p}_k$, and $\boldsymbol{z}_{k+1} = \boldsymbol{\theta}_{k+1} + \boldsymbol{p}_{k+1}$, using Eq. (Eq. 15), the following can be inferred

$$\boldsymbol{z}_{k+1} - \boldsymbol{z}_k = \frac{-\boldsymbol{\eta}_k}{(1-\beta)} g_k \tag{25}$$

Putting this into Eq. (24) gives

$$J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k) \leq \frac{-\boldsymbol{\eta}_k}{(1-\beta)} \left(\langle \nabla J(\boldsymbol{z}_k), g_k \rangle \right) + \frac{L}{2} \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \| g_k \|^2 \tag{26}$$

Since $\delta_k = g_k - \nabla J(\boldsymbol{\theta}_k)$, the above equation becomes

$$J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k) \leq \frac{-\boldsymbol{\eta}_k}{(1-\beta)} \left(\langle \nabla J(\boldsymbol{z}_k), (\delta_k + \nabla J(\boldsymbol{\theta}_k)) \rangle \right) + \frac{L}{2} \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \| \delta_k + \nabla J(\boldsymbol{\theta}_k) \|^2 \tag{27}$$

$$J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k) \leq \frac{-\boldsymbol{\eta}_k}{(1-\beta)} \left(\langle \nabla J(\boldsymbol{z}_k), \nabla J(\boldsymbol{\theta}_k) \rangle + \langle \delta_k, \nabla J(\boldsymbol{\theta}_k) \rangle \right) + \frac{L}{2} \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \| \delta_k + \nabla J(\boldsymbol{\theta}_k) \|^2 \tag{28}$$

Adding and subtracting $\nabla J(\boldsymbol{\theta}_k)$, and rearraging the equation gives

$$\leq \frac{-\boldsymbol{\eta}_k}{(1-\beta)} \left(\langle \nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) + \nabla J(\boldsymbol{\theta}_k), \nabla J(\boldsymbol{\theta}_k) \rangle \right) - \frac{\boldsymbol{\eta}_k}{(1-\beta)} \langle \delta_k, \nabla J(\boldsymbol{\theta}_k) \rangle + \frac{L}{2} \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \| \delta_k + \nabla J(\boldsymbol{\theta}_k) \|^2 \tag{29}$$

$$\leq \frac{-\boldsymbol{\eta}_k}{(1-\beta)}\left(\langle \nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k), \nabla J(\boldsymbol{\theta}_k)\rangle\right) + \frac{-\boldsymbol{\eta}_k}{(1-\beta)}\langle \nabla J(\boldsymbol{\theta}_k), \nabla J(\boldsymbol{\theta}_k)\rangle + \frac{-\boldsymbol{\eta}_k}{(1-\beta)}\langle \delta_k, \nabla J(\boldsymbol{\theta}_k)\rangle + \frac{L}{2}\frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \parallel \delta_k + \nabla J(\boldsymbol{\theta}_k) \parallel^2 \quad (30)$$

Taking expectation on both sides of the above equation and using $E[g_k] = E[\nabla J(\boldsymbol{\theta}_k)]$ and $E[\delta_k] = 0$ gives

$$E\left[J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k)\right] \leq E\left[\frac{-\boldsymbol{\eta}_k}{(1-\beta)}\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right)^T \nabla J(\boldsymbol{\theta}_k)\right] - E\left[\frac{\boldsymbol{\eta}_k}{(1-\beta)} \parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] + E\left[\frac{L\boldsymbol{\eta}_k^2}{2(1-\beta)^2} \parallel g_k \parallel^2\right] \quad (31)$$

$$E\left[J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k)\right] \leq E\left[\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right)^T \left(\frac{-\boldsymbol{\eta}_k}{(1-\beta)}\left(\nabla J(\boldsymbol{\theta}_k)\right)\right)\right] - E\left[\frac{\boldsymbol{\eta}_k}{(1-\beta)} \parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] + E\left[\frac{L\boldsymbol{\eta}_k^2}{2(1-\beta)^2} \parallel g_k \parallel^2\right] \quad (32)$$

Considering the property $ab \leq \frac{1}{2}(a^2+b^2)$ and rearranging the terms of the above equation as $\frac{1}{\sqrt{L}}\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right) = a$ and $\left(\frac{-\sqrt{L}\boldsymbol{\eta}_k}{(1-\beta)}\left(\nabla J(\boldsymbol{\theta}_k)\right)\right) = b$, gives

$$\leq E\left[\frac{1}{2L}\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right)^2\right] + E\left[\frac{L\boldsymbol{\eta}_k^2}{2(1-\beta)^2} \parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] - E\left[\frac{\boldsymbol{\eta}_k}{(1-\beta)} \parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] + E\left[\frac{L\boldsymbol{\eta}_k^2}{2(1-\beta)^2} \parallel g_k \parallel^2\right] \quad (33)$$

Recall from Assumption 3 that at any k, the expected value of the adaptive learning-rate and the expected value of the squared learning-rate for the i^{th} model-parameter are bounded by constants c_1 and c_2 respectively. Extending this for all the model-parameters, i.e. $i = 1 \dots d$ gives

$$E\left[\boldsymbol{\eta_k}\right] = E\left[diag\left(\eta_k^{(1)} \dots \eta_k^{(d)}\right)\right] \leq diag\left(c_1 \dots c_1\right) = \boldsymbol{C_1} \quad (34)$$

Similarly,

$$E\left[(\boldsymbol{\eta_k})^2\right] = E\left[diag\left(\left(\eta_k^{(1)}\right)^2 \dots \left(\eta_k^{(d)}\right)^2\right)\right] \leq diag(c_2 \dots c_2) = \boldsymbol{C_2} \quad (35)$$

Thus, substituting the above constants and rearranging the above Eq. (33) gives

$$\leq \frac{1}{2L}E\left[\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right)^2\right] - \left(\frac{\boldsymbol{C}_1}{(1-\beta)} - \frac{L\boldsymbol{C}_2}{(1-\beta)^2}\right)E\left[\parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] \quad (36)$$

Finally, the equation becomes

$$E\left[J(\boldsymbol{z}_{k+1}) - J(\boldsymbol{z}_k)\right] \leq \frac{1}{2L}E\left[\left(\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k)\right)^2\right] + \left(\frac{L\boldsymbol{C}_2}{(1-\beta)^2} - \frac{\boldsymbol{C}_1}{(1-\beta)}\right)E\left[\parallel \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] \quad (37)$$

□

This lemma will be used in the proof of the Theorem 2 in the next section.

Lemma 4. *for a function $J : \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz gradients, there exists*

$$E[\parallel \nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2] \leq \left(\frac{L^2\beta^2}{(1-\beta)^2}\sum_{i=0}^{k-1}\beta^{2i}\boldsymbol{C}_1 \parallel \nabla J(\boldsymbol{\theta}_{k-1-i}) \parallel^2\right) \quad (38)$$

where $\Gamma_{k-1} := \sum_{i=0}^{k-1}\beta^i = \frac{1-\beta^k}{1-\beta}$, L is the lipschitz constant, $\beta \in (0, 1]$ is the parameter defined previously, and $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are constants.

Proof. From the property of lipschitz gradient

$$\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel \leq L \parallel \boldsymbol{z}_k - \boldsymbol{\theta}_k \parallel \quad (39)$$

Squaring both sides gives

$$\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2 \leq L^2 \parallel \boldsymbol{z}_k - \boldsymbol{\theta}_k \parallel^2 \quad (40)$$

Since $\boldsymbol{z}_k = \boldsymbol{\theta}_k + \boldsymbol{p}_k$

$$\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2 \leq L^2 \parallel \boldsymbol{p}_k \parallel^2 \quad (41)$$

Given $\boldsymbol{p}_k = \frac{\beta}{(1-\beta)}\left(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}\right)$

$$\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2 \leq \frac{L^2\beta^2}{(1-\beta)^2} \parallel \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} \parallel^2 \quad (42)$$

Now, representing $\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$ with the variable v_k, i.e., $v_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$

$$\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2 \leq \frac{L^2\beta^2}{(1-\beta)^2} \parallel v_k \parallel^2 \quad (43)$$

Since $v_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$ and $\boldsymbol{p}_k = \frac{\beta}{(1-\beta)}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$, the following relation between v_k and p_k can be derived

$$v_k = \frac{(1-\beta)}{\beta}\boldsymbol{p}_k \text{ and } v_{k+1} = \beta v_k - \boldsymbol{\eta}_k g_k \tag{44}$$

Assuming $\theta_{(-1)} = \theta_0$ gives $v_0 = 0$. Then applying induction for $k > 1$ gives

$$v_k = -\sum_{i=0}^{k-1} \beta^i \boldsymbol{\eta}_{k-1-i} g_{k-1-i} \tag{45}$$

Also

$$\parallel v_k \parallel^2 = \sum_{i=0}^{k-1} \beta^{2i} \boldsymbol{\eta}_{k-1-i}^2 g_{k-1-i}^2 \tag{46}$$

Substitute the above into Eq. (43) and take expectation on both sides

$$E\left[\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] \leq \frac{L^2\beta^2}{(1-\beta)^2} E\left[\left(\sum_{i=0}^{k-1} \beta^{2i} \boldsymbol{\eta}_{k-1-i}^2 g_{k-1-i}^2\right)\right] \tag{47}$$

Substituting $E[g_k] = E[\nabla J(\boldsymbol{\theta}_k)]$ and $E\left[(\boldsymbol{\eta}_k)^2\right] \leq \boldsymbol{C}_2$

$$E\left[\parallel (\nabla J(\boldsymbol{z}_k) - \nabla J(\boldsymbol{\theta}_k) \parallel^2\right] \leq \frac{L^2\beta^2}{(1-\beta)^2} \sum_{i=0}^{k-1} \beta^{2i} \boldsymbol{C}_2 E\left[\parallel \nabla J(\boldsymbol{\theta}_{k-1-i}) \parallel^2\right] \tag{48}$$

This lemma will be used in the proof of the Theorem 2 in the next section. □

A.2 Proof of Theorems

Theorem 1:
The theorem derives the regret bound of the proposed algorithm in context of a convex loss function. It is proven that the regret of the convex and differentiable loss function (with L-Lipschitz gradients) is upper bound by the square of the Euclidean distance between the initial model-parameters $\boldsymbol{\theta}_0$ and the optimal model-parameters $\boldsymbol{\theta}^*$.

$$R_J(T) = J(\bar{\boldsymbol{\theta}}_T) - J(\boldsymbol{\theta}^*) \leq \frac{1}{T}\left[\frac{2(1-\beta)^2 + L\beta\boldsymbol{C}_1}{2\boldsymbol{C}}\right]\left(\parallel \boldsymbol{\theta}_0 - \boldsymbol{\theta}^* \parallel_2^2\right) \tag{49}$$

where $J\left(\bar{\boldsymbol{\theta}}_T\right) = \frac{1}{T}\sum_{k=0}^{T}\left(J\left(\boldsymbol{\theta}_i\right)\right)$; $\boldsymbol{C}$ and $\boldsymbol{C}_1$ are positive constants defined during the proof and T is the number of iterations of the adaptive algorithm.

Proof. Consider the update equation given by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k g_k + \beta\left(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}\right) \tag{50}$$

where $\boldsymbol{\theta}_k \in \mathbb{R}^d$ represents the model-parameters at the k^{th} iteration of the optimization algorithm, $\boldsymbol{\eta}_k$ is the adaptive learning rate, g_k is the gradient of the objective function at k, and β is the decay parameter.
From lemma 2

$$\boldsymbol{\theta}_{k+1} + \boldsymbol{p}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{p}_k - \frac{\boldsymbol{\eta}_k}{(1-\beta)} g_k \tag{51}$$

Next, Subtracting $\boldsymbol{\theta}^*$ and squaring both sides gives,

$$\parallel \boldsymbol{\theta}_{k+1} + \boldsymbol{p}_{k+1} - \boldsymbol{\theta}^* \parallel^2 = \parallel \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^* \parallel^2 - \frac{2\boldsymbol{\eta}_k}{(1-\beta)}\left\langle g_k, \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^*\right\rangle + \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \parallel g_k \parallel^2 \tag{52}$$

rearranging the terms

$$\parallel \boldsymbol{\theta}_{k+1} + \boldsymbol{p}_{k+1} - \boldsymbol{\theta}^* \parallel^2 - \parallel \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^* \parallel^2 = -\frac{2\boldsymbol{\eta}_k}{(1-\beta)}\left\langle g_k, \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^*\right\rangle + \frac{\boldsymbol{\eta}_k^2}{(1-\beta)^2} \parallel g_k \parallel^2 \tag{53}$$

Considering the first term of the right hand side of the equation and taking into account the upper bounds on $E\left[\boldsymbol{\eta_k}\right]$ and $E\left[\left(\boldsymbol{\eta_k}\right)^2\right]$, gives

$$\begin{aligned}\frac{-2}{(1-\beta)} E[\boldsymbol{\eta}_k \left\langle g_k, \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^*\right\rangle] &= \frac{-2}{(1-\beta)} E[\boldsymbol{\eta}_k] E[\left\langle g_k, \boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^*\right\rangle] \\ &\leq \frac{-2}{(1-\beta)} \boldsymbol{C}_1 \left\langle E[g_k], (\boldsymbol{\theta}_k + \boldsymbol{p}_k - \boldsymbol{\theta}^*)\right\rangle \end{aligned} \tag{54}$$

Substituting $\boldsymbol{p}_k = \frac{\beta}{(1-\beta)}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$ and rearranging the terms

$$\leq \frac{-2}{(1-\beta)} \boldsymbol{C}_1 \left\langle E[g_k], \left((\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) + \frac{\beta}{(1-\beta)}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})\right)\right\rangle \tag{55}$$

$$\leq \frac{-2\boldsymbol{C}_1}{(1-\beta)}\left(\left\langle \nabla J(\boldsymbol{\theta}_k), (\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)\right\rangle\right) + \frac{-2\boldsymbol{C}_1\beta}{(1-\beta)^2}\left(\left\langle \nabla J(\boldsymbol{\theta}_k), (\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})\right\rangle\right) \tag{56}$$

Considering the convexity of the function (Assumption 1), gives

$$\leq \frac{-2\boldsymbol{C}_1}{(1-\beta)}\left(J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}^*)\right) + \frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}_{k-1})\right) \tag{57}$$

Now substituting Eq. 57 into Eq. 53 and summing over $k = 0...T$, gives

$$\frac{2\boldsymbol{C}_1}{(1-\beta)}\left(\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)$$
$$\leq \frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(\sum_{k=0}^{T}J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})\right)+\frac{1}{(1-\beta)^2}\sum_{k=0}^{T}\boldsymbol{\eta}_k^2\parallel g_k\parallel^2$$
$$-\sum_{k=1}^{T}\left(\parallel \boldsymbol{\theta}_{k+1}+\boldsymbol{p}_{k+1}-\boldsymbol{\theta}^*\parallel^2-\parallel \boldsymbol{\theta}_k+\boldsymbol{p}_k-\boldsymbol{\theta}^*\parallel^2\right) \tag{58}$$

The term $\sum_{k=0}^{T}\parallel \boldsymbol{\theta}_{k+1}+\boldsymbol{p}_{k+1}-\boldsymbol{\theta}^*\parallel^2-\parallel \boldsymbol{\theta}_k+\boldsymbol{p}_k-\boldsymbol{\theta}^*\parallel^2$ can be expressed as a telescopic sum, where consecutive terms cancel each other leaving only the difference of the first and final terms.

$$\frac{2\boldsymbol{C}_1}{(1-\beta)}\left(\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)$$
$$\leq \frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\sum_{k=0}^{T}J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})+\frac{1}{(1-\beta)^2}\sum_{k=0}^{T}\boldsymbol{\eta}_k^2\parallel g_k\parallel^2$$
$$-\left(\parallel \boldsymbol{\theta}_{T+1}+\boldsymbol{p}_{T+1}-\boldsymbol{\theta}^*\parallel^2-\parallel \boldsymbol{\theta}_0+\boldsymbol{p}_0-\boldsymbol{\theta}^*\parallel^2\right) \tag{59}$$

since $\boldsymbol{p_0}=\mathbf{0}$, the above equation can be simplified as

$$\frac{2\boldsymbol{C}_1}{(1-\beta)}\left(\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)$$
$$\leq \frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\sum_{k=0}^{T}J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})+\frac{1}{(1-\beta)^2}\sum_{k=0}^{T}\boldsymbol{\eta}_k^2\parallel g_k\parallel^2$$
$$+\parallel \boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{60}$$

Consider the second term of the right hand side of the Eq. 60

$$E\left[\sum_{k=0}^{T}\boldsymbol{\eta}_k^2\parallel g_k\parallel^2\right]=\sum_{k=0}^{T}E\left[\boldsymbol{\eta}_k^2\right]E\left[\parallel g_k\parallel^2\right]$$
$$\leq\sum_{k=0}^{T}\boldsymbol{C}_2\parallel\nabla J(\boldsymbol{\theta}_k)\parallel^2 \tag{61}$$

Now, implementing Lemma 2, gives

$$E\left[\sum_{k=0}^{T}\boldsymbol{\eta}_k^2\parallel g_k\parallel^2\right]\leq\sum_{k=0}^{T}\boldsymbol{C}_2\left(2L\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right) \tag{62}$$

Substituting Eq. (62) into Eq. (60)

$$\frac{2}{(1-\beta)}\left(\sum_{k=0}^{T}\boldsymbol{C}_1\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)$$
$$\leq\frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})\right)+$$
$$\frac{1}{(1-\beta)^2}\sum_{k=0}^{T}\boldsymbol{C}_2\left(2L\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)+\parallel\boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{63}$$

Rearranging the terms to get $J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)$ on one side of the equation

$$\frac{2}{(1-\beta)^2}\left(\sum_{k=0}^{T}\left((1-\beta)\boldsymbol{C}_1-2L\boldsymbol{C}_2\right)\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\right)$$
$$\leq\frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})\right)+\parallel\boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{64}$$

$$\frac{2}{(1-\beta)^2}\left((1-\beta)\boldsymbol{C}_1-2L\boldsymbol{C}_2\right)\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)$$
$$\leq\frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}_{k-1})\right)+\parallel\boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{65}$$

Let $\boldsymbol{C}:=\min_{k=0\ldots T}\left((1-\beta)\boldsymbol{C}_1-2L\boldsymbol{C}_2\right)$

$$\frac{2\boldsymbol{C}}{(1-\beta)^2}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)$$
$$\leq\frac{-2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(J(\boldsymbol{\theta}_T)-J(\boldsymbol{\theta}_0)\right)+\parallel\boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{66}$$

$$\frac{2\boldsymbol{C}}{(1-\beta)^2}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)$$
$$\leq\frac{2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(J(\boldsymbol{\theta}_0)-J(\boldsymbol{\theta}_T)\right)+\parallel\boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \tag{67}$$

Now, consider the first term of the right hand side of the equation. Rearranging the terms and using the convexity property of the loss function, gives

$$\frac{2\beta\boldsymbol{C}_1}{(1-\beta)^2}\left(J(\boldsymbol{\theta}_0)-J(\boldsymbol{\theta}^*)\right)-\left(J(\boldsymbol{\theta}_T)-J(\boldsymbol{\theta}^*)\right)$$
$$=\frac{2\beta\boldsymbol{C}_1}{(1-\beta)^2}\frac{L}{2}\left(\parallel\theta_0-\theta^*\parallel^2-\parallel\theta_T-\theta^*\parallel^2\right) \tag{68}$$

And since $\parallel\theta_T-\theta^*\parallel^2$ is always positive, this implies

$$\parallel\theta_0-\theta^*\parallel^2-\parallel\theta_T-\theta^*\parallel^2\leq\parallel\theta_0-\theta^*\parallel^2$$

Thus, the Eq. (67) becomes

$$\frac{2\boldsymbol{C}}{(1-\beta)^2}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right) \leq \frac{2\beta\boldsymbol{C}_1}{(1-\beta)^2}\frac{L}{2}\parallel \boldsymbol{\theta}_T-\boldsymbol{\theta_0}\parallel^2+\parallel \boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \quad (69)$$

$$\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)\leq\left(\frac{2(1-\beta)^2+L\beta\boldsymbol{C}_1}{2\boldsymbol{C}}\right)\parallel \boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \quad (70)$$

The Eq. (70) represents the regret bound $R_J(T)$ of the loss function at instance k, and is bounded by

$$R_J(T)=\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)=\mathcal{O}\parallel \boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2 \quad (71)$$

Applying the Jensen's inequality, i.e. $J(\bar{\boldsymbol{\theta}_T})-J(\boldsymbol{\theta}^*)\leq \frac{1}{T}\sum_{k=0}^{T}\left(J(\boldsymbol{\theta}_k)-J(\boldsymbol{\theta}^*)\right)$, the rate of convergence of the algorithm is given by

$$J(\bar{\boldsymbol{\theta}_T})-J(\boldsymbol{\theta}^*) \leq \frac{1}{T}\left(\frac{L\beta\boldsymbol{C}_1+2(1-\beta)^2}{2\boldsymbol{C}}\right)\parallel \boldsymbol{\theta}_0-\boldsymbol{\theta}^*\parallel^2=\mathcal{O}\left(\frac{1}{T}\right) \quad (72)$$

where $J\left(\bar{\boldsymbol{\theta}}_T\right)=\frac{1}{T}\sum_{k=0}^{T}\left(J\left(\boldsymbol{\theta}_k\right)\right)$ □

Theorem 2:

Proof. From lemma 3

$$E\left[J(\boldsymbol{z}_{k+1})-J(\boldsymbol{z}_k)\right]\leq\frac{1}{2L}E\left[\left(\nabla J(\boldsymbol{z}_k)-\nabla J(\boldsymbol{\theta}_k)\right)^2\right]+\left(\frac{L\boldsymbol{C}_2}{(1-\beta)^2}-\frac{\boldsymbol{C}_1}{(1-\beta)}\right)E\left[\parallel \nabla J(\boldsymbol{\theta}_k)\parallel^2\right] \quad (73)$$

Substitute first term of the right hand side using lemma 4

$$E[J(\boldsymbol{z}_{k+1})-J(\boldsymbol{z}_k)] \leq \frac{1}{2L}\left(\frac{L^2\beta^2}{(1-\beta)^2}\sum_{i=0}^{k-1}\beta^{2i}\boldsymbol{C}_2E\left[\parallel \nabla(J(\boldsymbol{\theta}_{k-1-i})\parallel^2\right]\right)-\left(\frac{\boldsymbol{C}_1}{(1-\beta)}-\frac{L\boldsymbol{C}_2}{(1-\beta)^2}E[\parallel \nabla J(\boldsymbol{\theta}_k)\parallel^2]\right) \quad (74)$$

Summing both sides over $k=0...T$, where the left-hand side of the equation can be expressed as a telescopic sum and consecutive terms cancel each other leaving only the difference of the first and final terms.

$$E[J(\boldsymbol{z}_{T+1})-J(\boldsymbol{z}_0)] \leq -\sum_{k=0}^{T}\left(\frac{\boldsymbol{C}_1}{(1-\beta)}-\frac{L\boldsymbol{C}_2}{(1-\beta)^2}\right)E[\parallel \nabla J(\boldsymbol{\theta}_k)\parallel^2]+\frac{1}{2L}\frac{L^2\beta^2}{(1-\beta)^2}\sum_{k=0}^{T}\left(\sum_{i=0}^{k-1}\beta^{2i}\boldsymbol{C}_2E\left[\parallel \nabla(J(\boldsymbol{\theta}_{k-1-i})\parallel^2\right]\right) \quad (75)$$

Considering the second term of the right-hand side of the equation

$$\frac{1}{2L}\frac{L^2\beta^2\boldsymbol{C}_2}{(1-\beta)^2}\sum_{k=0}^{T}\left(\sum_{i=0}^{k-1}\beta^{2i}\boldsymbol{C}_2E\left[\parallel \nabla(J(\boldsymbol{\theta}_{k-1-i})\parallel^2\right]\right) =\frac{1}{2L}\frac{L^2\beta^2\boldsymbol{C}_2}{(1-\beta)^2}\sum_{k=0}^{T}\left(\sum_{i=0}^{k-1}\beta^{2i}\right)\boldsymbol{C}_2E\left[\parallel \nabla(J(\boldsymbol{\theta}_{k-1-i})\parallel^2\right] \quad (76)$$

Using the formula for the sum of geometric series[2] and rearranging the indices, gives

$$=\frac{1}{2L}\frac{L^2\beta^2\boldsymbol{C}_2}{(1-\beta)^2}\sum_{k=0}^{T-1}\left(\sum_{i=k}^{T-1}\beta^{2(i-k)}\right)E\left[\parallel \nabla(J(\boldsymbol{\theta}_k)\parallel^2\right] \quad (77)$$

Thus the Eq. (75) becomes,

$$E[J(\boldsymbol{z}_{T+1})-J(\boldsymbol{z}_0)] \leq -\sum_{k=0}^{T}\left(\frac{\boldsymbol{C}_1}{(1-\beta)}-\frac{L\boldsymbol{C}_2}{(1-\beta)^2}\right)E[\parallel \nabla J(\boldsymbol{\theta}_k)\parallel^2]+\frac{1}{2L}\frac{L^2\beta^2\boldsymbol{C}_2}{(1-\beta)^2}\sum_{k=0}^{T-1}\left(\sum_{i=k}^{T-1}\beta^{2(i-k)}\right)E\left[\parallel \nabla(J(\boldsymbol{\theta}_k)\parallel^2\right] \quad (78)$$

Let $\bar{\boldsymbol{C}}=\left(\frac{\boldsymbol{C}_1}{(1-\beta)}-\frac{L\boldsymbol{C}_2}{(1-\beta)^2}\right)$

and $\bar{\boldsymbol{D}}=\frac{1}{2L}\frac{L^2\beta^2\boldsymbol{C}_2}{(1-\beta)^2}\sum_{i=k}^{T-1}\beta^{2(i-k)}$

After simplifying and substituting the given expressions for $\bar{\boldsymbol{C}}$ and $\bar{\boldsymbol{D}}$, the Eq. (78) can be expressed as:

[2]The representation of the sum of a finite geometric series is given by $S_n=\sum_{i=0}^{n-1}r^i=\left(\frac{1-r^n}{1-r}\right)$, where n is the number of terms in the series and "S_n" represents the sum of the first "n" terms.

$$E[J(\boldsymbol{z}_{T+1}) - J(\boldsymbol{z}_0)]$$
$$\leq -\sum_{k=0}^{T-1} \left(\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}\right) E\left[\| \nabla J(\boldsymbol{\theta}_k) \|^2\right] - \bar{\boldsymbol{C}} \| \nabla (J(\boldsymbol{\theta}_T) \|^2) \tag{79}$$

Rearranging the terms

$$\sum_{k=0}^{T-1} \left(\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}\right) E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq E[J(\boldsymbol{z}_0) - J(\boldsymbol{z}_{T+1})] - \bar{\boldsymbol{C}} \| \nabla (J(\boldsymbol{\theta}_T) \|^2) \tag{80}$$

Since $\bar{\boldsymbol{C}} \| \nabla (J(\boldsymbol{\theta}_T) \|^2) \geq 0$ is always true, the above inequality is equivalent to

$$\sum_{k=0}^{T-1} \left(\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}\right) E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq E[J(\boldsymbol{z}_0) - J(\boldsymbol{z}_{T+1})] \tag{81}$$

Since $\boldsymbol{p}_0 = 0$ implies $J(\boldsymbol{z}_0) = J(\boldsymbol{\theta}_0)$, and $J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{z}_{T+1})$, the following relation can be derived[3]:

$$E[J(\boldsymbol{z}_0) - J(\boldsymbol{z}_{T+1})] = E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{z}_{T+1})] \leq E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \tag{82}$$

Thus the Eq. (81) can be written as

$$\sum_{k=0}^{T-1} \left(\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}\right) E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \tag{83}$$

$$\sum_{k=0}^{T-1} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq \frac{1}{\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}} E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \tag{84}$$

The subsequent equation follows from the fact that the minimum of a set of values is always less than or equal to their average. i.e.,

$$\min_{k=0 \ldots T-1} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq \sum_{k=0}^{T-1} \frac{1}{T} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq \frac{1}{T} \frac{1}{\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}} E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \tag{85}$$

This inequality proves that the minimum of the expected sum of squared gradients of the loss function $E[\| \nabla J(\boldsymbol{\theta}_k) \|^2]$ over iterations $k = 0, \ldots, T-1$ is upper bounded by the difference between the value of the objective function at the start and end of the iterations. Finally, the following equation completes the proof.

$$\min_{k=0 \ldots T-1} E[\| \nabla J(\boldsymbol{\theta}_k) \|^2] \leq \frac{1}{\bar{\boldsymbol{C}} - \bar{\boldsymbol{D}}} \frac{1}{T} E[J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] \tag{86}$$

□

[3] This inequality holds because $J(\boldsymbol{\theta}^*)$ is the minimum achievable value of $J(\cdot)$

Ethical Statement

Acknowledgments