Original Article
ISSN (Online): 2582-7472

ShodhKosh: Journal of Visual and Performing Arts
July-December 2023 4(2), 4445–4459

PERFORMANCE MEASURE OF VARIOUS MACHINE LEARNING OPTIMIZERS FOR DIABETES PREDICTION IN INDIAN WOMAN

Surendra Gour [1], Md. Tabrez Nafis [1], Suraiya Parveen [1]

[1] Department of Computer Science and Engineering, Jamia Hamdard New Delhi,110062, India

ABSTRACT

This research paper presents a comprehensive comparative analysis of gradient descent optimization algorithms using a Diabetes Prediction dataset. The study explores their strengths, weaknesses, and performance characteristics under two different conditions, namely with and without feature engineering. The objective is to obtain proper insights into the effectiveness and efficiency of these algorithms in predicting diabetes. The analysis focuses on widely used algorithms, including stochastic gradient descent (SGD) and advanced variants like Nesterov accelerated gradient and adaptive learning rate techniques (e.g., Adam, AdaGrad, AdaMax, and AdaDelta). By evaluating their performance on the dataset under two different scenarios this research provides valuable insights into the performance of these algorithms. The obtained result show that, SGD variants (classic SGD, momentum, Nesterov), RMSProp, Adam, AdaMax, and Nadam outperformed AdaGrad and AdaDelta in minimizing error (lower MAE values) in both scenarios.

Keywords: Optimizers, Learning Rate, Gradient, Iteration, Momentum

Corresponding Author

Surendra Gour,
surendragour@gmail.com

DOI
10.29121/shodhkosh.v4.i2.2023.5206

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

1. INTRODUCTION

The concept of gradient descent has revolutionized machine learning and optimization, serving as a fundamental approach to discovering optimal solutions in a wide range of complex problems [16]. With the ever-increasing volume of data and the need for efficient computational methods, the development and refinement of gradient descent optimization algorithms have become crucial for researchers and practitioners alike [17].

The objective of this research paper is to perform a comprehensive comparative analysis of different gradient descent optimization algorithms. By examining their strengths, weaknesses, and performance characteristics, we aim to shed light on the most effective and efficient algorithms for different problem domains. This analysis will provide valuable insights for researchers, enabling them to make informed decisions while choosing the appropriate optimization algorithm for their specific applications.

The study focuses on investigating a range of widely used gradient descent optimization algorithms, including but not limited to: stochastic gradient descent (SGD) and advanced variants such as momentum-based methods (e.g., Nesterov accelerated gradient), adaptive learning rate techniques (e.g., Adam, AdaGrad, AdaMax and AdaDelta).

2. RELATED WORKS

One popular extension of gradient descent is stochastic gradient descent (SGD). Unlike traditional gradient descent that computes the gradients of the loss function for the entire training dataset, SGD updates the model parameters based on the gradient computed for a random subset of the data. This random sampling introduces noise in the updates, which can lead to faster and more memory-efficient training [18]. However, SGD may also result in noisy updates and reduced accuracy compared to traditional gradient descent [1].

Another variant of gradient descent is Nesterov accelerated gradient (NAG). NAG incorporates a momentum term into the update rule, which allows the algorithm to take into account the direction of the previous update. By considering this momentum factor, NAG can achieve faster convergence and improved performance in certain cases [2].

RMSprop is another optimization technique that has gained attention in recent years. It addresses the challenges posed by non-stationary problems by adapting the learning rate for each parameter based on the historical gradients. This adaptive learning rate prevents the learning rate from oscillating too much and helps the algorithm converge more effectively [3].

AdaGrad is an optimization algorithm that adjusts the learning rate for each parameter based on the historical gradients. It gives larger updates to parameters associated with infrequent features and smaller updates to frequently occurring features. This adaptive learning rate scheme has shown promising results, particularly in scenarios with sparse data [4].

Adadelta is another adaptive learning rate method that addresses some of the limitations of AdaGrad. It seeks to improve the stability of the learning process by using a more sophisticated update rule that takes into account only a window of the most recent gradients. This windowing approach allows Adadelta to adapt more effectively to changing conditions during training [5].

Adam, short for adaptive moment estimation, is a popular optimization algorithm that combines ideas from both momentum-based methods and adaptive learning rate methods. It maintains exponential moving averages of both the gradients and the squared gradients. By utilizing these estimates, Adam can adaptively adjust the learning rate and momentum parameters for each parameter in the model. This adaptive behaviour has contributed to the algorithm's widespread adoption and success across various domains [6].

Adamax is a variant of Adam that introduces an infinity norm constraint on the update step. By using the infinity norm, Adamax can provide better handling of very large gradients, which can be beneficial in scenarios where the magnitude of gradients varies significantly across different parameters [7].

In addition to the individual analyses of these optimization techniques, several studies have conducted comparative analyses to evaluate their performance against each other [19][20][21]. For instance, in a study focused on wind speed forecasting [8], researchers compared Gradient descent, Momentum, AdaGrad, RMSprop, Adam, and Adadelta. The results showed that Adam and RMSprop outperformed the other techniques in terms of both performance and training time.

Similarly, in the field of ophthalmology, a comparative study [9] compared various optimization techniques and their impact on finding optimal solutions in minimum iterations. The authors found that AdaGrad consistently provided the best solutions within the least number of iterations in this specific domain.

These comparative analyses highlight the importance of selecting an appropriate optimization technique based on the specific problem domain and requirements. The choice of optimization algorithm can significantly impact both the training time and the quality of the resulting models [22]. Researchers continue to explore and propose new extensions and improvements to gradient descent in order to further enhance its performance and applicability in various fields.

3. GRADIENT DESCENT OPTIMIZATION ALGORITHMS

3.1. STOCHASTIC GRADIENT DESCENT (SGD)

SGD minimizes the time to converge by randomly selecting the data points at each iteration to update the parameter. The update parameter for SGD is as follows:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta;\, x(i), y(i))$$

Where x(i) and y(i) are randomly selected data points. This random selection, however, results in less accuracy in comparison to gradient descent.

3.2. MOMENTUM

The momentum technique [10] is used to improve the convergence of SGD by adding a fraction of the previous gradient to the current gradient. This fraction is referred to as the momentum coefficient (γ). The parameter, in this case, is updated as follows:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

Here v_t is the velocity vector. Momentum has been shown to improve the convergence speed of SGD and reduce oscillations, which can lead to better performance in training deep neural networks [11].

3.3. NESTEROV ACCELERATED GRADIENT (NAG)

The NAG algorithm can be seen as a modification of the classical momentum algorithm, where instead of calculating the gradient at the current position, the gradient is calculated at the estimated future position of the parameters. This estimated future position is obtained by adding the current momentum to the current parameter position. The NAG algorithm then uses this estimated future position to calculate the gradient, which is then used to update the momentum and the parameters [2]. So, the update rule for the NAG algorithm can be written as follows:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

where θ and $\nabla_\theta J(\theta)$ are the current parameters and gradient, respectively, v_t is the current momentum, γ is the momentum coefficient, η is the learning rate, and $\nabla_\theta J(\theta - \gamma v_{t-1})$ is the gradient at the estimated future position of the parameters.

3.4. ADAGRAD

The adaptive gradient algorithm (Adagrad) is an optimization algorithm that adapts the learning rate for each parameter in the network based on its previous history of gradients. The main idea behind Adagrad is to use a different learning rate for each parameter, based on the information available for that parameter, rather than using a global learning rate for all parameters [4]. The equation for the updating of parameters is as follows:

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$$

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

where θ_(t,i) and ∇_(θ_t) J(θ_(t,i)) are the current parameters and gradient, respectively, g_(t,i) is the sum of the squares of the gradients for parameter i up to time step t, η is the learning rate, G_t is a diagonal matrix where each diagonal element i,i is the sum of the squares of the gradients for parameter θ_i up to time step t, and ϵ is a small constant added to the denominator to ensure numerical stability.

One of the benefits of Adagrad is that it requires little tuning of hyperparameters and can converge quickly, especially for sparse data sets [12]. However, Adagrad has been shown to have some limitations, such as the accumulation of squared gradients over time, which can lead to a very small learning rate [12].

3.5. RMSPROP

Root Mean Square Propagation (RMSprop) is an optimization algorithm that adapts the learning rate for each parameter in the network based on its moving average of squared gradients [3]. The main idea behind RMSprop is to divide the learning rate for weight by a running average of the magnitudes of recent gradients for that weight [3].

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Where E is moving average of squared gradients and β is moving average parameter.

3.6. ADADELTA

Adadelta is an adaptive learning rate optimization algorithm that is similar to RMSprop. Like RMSprop, Adadelta maintains a moving average of the squared gradient for each weight, but instead of using a fixed learning rate, Adadelta uses the ratio of the root mean squared (RMS) of the gradients and the RMS of the updates to adjust the learning rate [5].

$$g_t = \nabla J(\theta_t)$$

$$E[g^2]t = \rho E[g^2]t - 1 + (1-\rho) g_t^2$$

$$\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]t - 1 + \epsilon}}{\sqrt{E[g^2]t + \epsilon}} g_t$$

$$E[\Delta\theta^2]t = \rho E[\Delta\theta^2]t - 1 + (1-\rho)\Delta\theta_t^2$$
$$\theta t + 1 = \theta t + \Delta\theta_t$$

where g_t is the gradient at time step t, $E[g^2]t$ is the exponentially decaying average of the past squared gradients, $\Delta\theta_t$

is the update at time step t, $E[\Delta\theta^2]t$ is the exponentially decaying average of the past squared updates, ρ is the decay rate, and ϵ is a small constant added to avoid division by zero [12].

3.7. ADAM

Adam is a popular optimization algorithm that combines momentum-based optimization and adaptive learning rate methods. Adam maintains a running estimate of the first and second moments of the gradient (i.e., the mean and variance of the gradient), which are then used to update the parameters. Specifically, Adam computes a moving average of the gradient and its squared values, which are then used to adjust the learning rate for each weight. The algorithm includes two key hyperparameters: the learning rate (α) and the momentum decay rate (β) [6].

The update rule for Adam is as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

To correct the bias from the first and second moments:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t}$$

3.8. ADAMAX

Adamax is another variant of the Adam optimization algorithm that is designed to address certain shortcomings of Adam when dealing with very large datasets. While Adam calculates the exponentially weighted moving average (EWMA) of both the first and second moments of the gradient, Adamax only calculates the EWMA of the first moment and uses the L-infinity norm (maximum absolute value) of the gradients for the second moment [7] [13].

The update rule for Adamax is as follows:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)|g_t|^2$$

$$v_t = \beta_2^p v_{t-1} + \left(1-\beta_2^p\right)|g_t|^p$$

$$u_t = \beta_2^\infty v_{t-1} + (1-\beta_2^\infty)|g_t|^\infty$$

$$= (\beta_2 \cdot v_{t-1}, |g_t|)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t} \widehat{m_t}$$

The key difference between Adamax and Adam is that Adamax uses the L-infinity norm to normalize the gradients instead of the L2 norm. This makes Adamax more resilient to outliers, which can be particularly useful when working with large datasets that may contain noisy or corrupted data [7] [13].

3.9. NADAM

Nadam (Nesterov-accelerated Adaptive Moment Estimation) [14] is another variant of the Adam optimization algorithm that combines the Nesterov accelerated gradient (NAG) method with Adam. NAG is a variant of gradient descent that uses a momentum term to accelerate the convergence of the optimization algorithm. Nadam builds on Adam by incorporating the NAG method to achieve faster convergence and better generalization.

The updated rule for Nadam [14] is as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t = \theta t - 1 - \frac{\alpha}{\sqrt{\widehat{v_t}} + \epsilon} \left(\beta_1 \widehat{m_t} + \frac{(1 - \beta_1)\left(1 + \beta_1 (m_t - \widehat{m_t})\right)}{1 - \beta_1^t} g_t \right)$$

Where $\beta_1, \beta_2, \widehat{m_t}\ and\ \widehat{v_t}$ are have significance as in adam.

4. CASE STUDY

In the following study we have used tensorflow optimizers to compare the performance of SGD, momentum, NAG, RMSProp, AdaGrad, AdaDelta, Adam, AdaMax and Nadam algorithms. The performance is measured on speed of convergence and accuracy. We have considered two cases, (i) Fixed number of iterations and (ii) No constraint on iterations.

Data

The dataset in both the cases is for diabetes prediction [15]. The Pima Indian Women Diabetes Prediction dataset comprises a collection of 768 instances, all of which pertain to female individuals of Pima Indian heritage. Each instance is characterized by eight different diagnostic measurements and a binary outcome variable. The dataset features are described as follows:

Table 1 Features of Dataset

Feature	Description
Pregnancies	Number of times the patient has been pregnant
Glucose	Plasma glucose concentration measured 2 hours after an oral glucose tolerance test
Blood Pressure	Diastolic blood pressure recorded in millimetres of mercury (mm Hg)
Skin Thickness	Thickness of the triceps skin fold measured in millimetres (mm)
Insulin	2-Hour serum insulin level measured in microunits per millilitre (mu U/ml)

BMI	Body mass index calculated as weight in kilograms divided by the square of height in meters (kg/m^2)
Diabetes Pedigree Function	A function that quantifies the hereditary risk of diabetes

The target variable is 'Outcome': A binary variable indicating the presence (1) or absence (0) of diabetes.

Case I: Fixed number of iterations

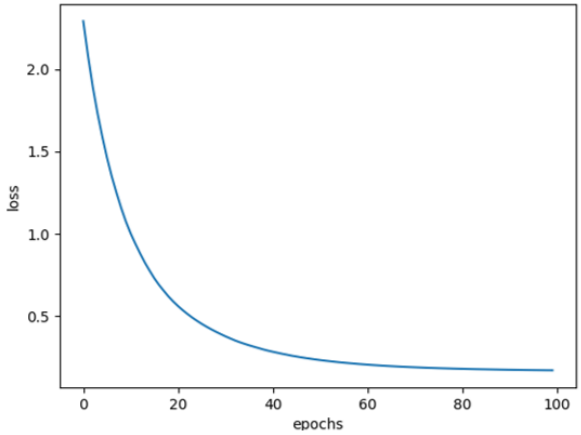

Figure 1 SGD Error Function

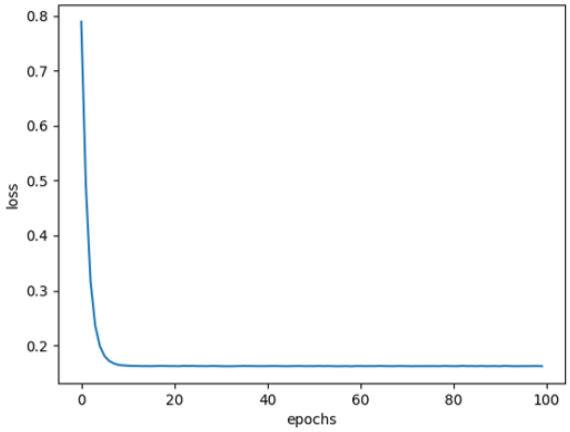

Figure 2 Momentum Error Function

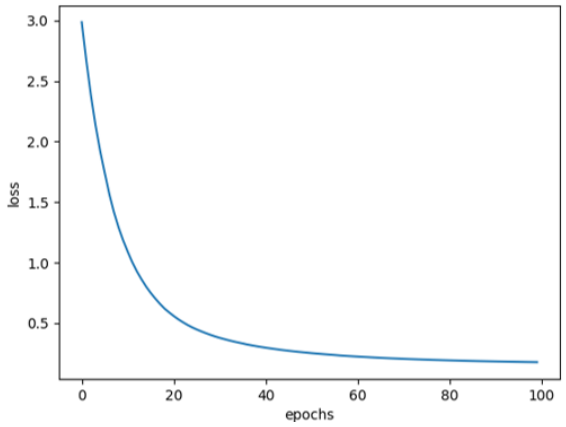

Figure 3 NAG Error Function

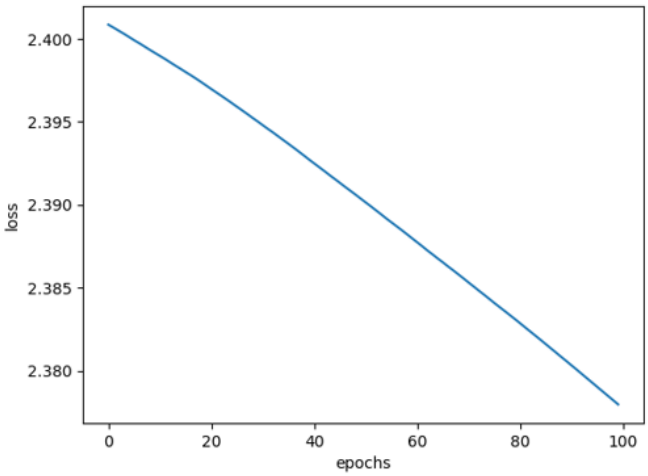

Figure 4 Ada Delta Error Function

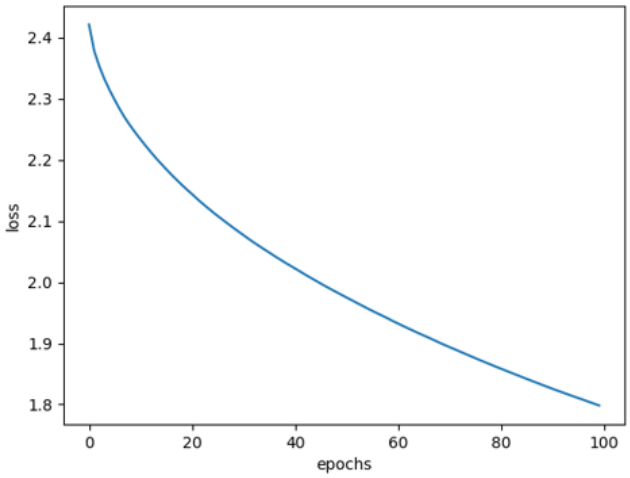

Figure 5 Ada Grad Error Function

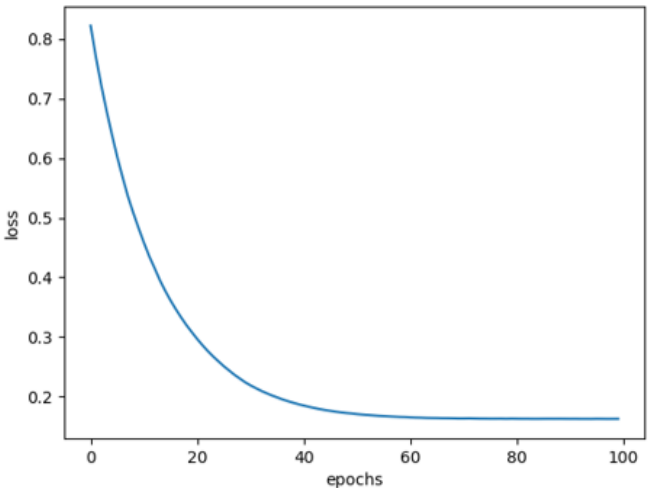

Figure 6 Adam Error Function

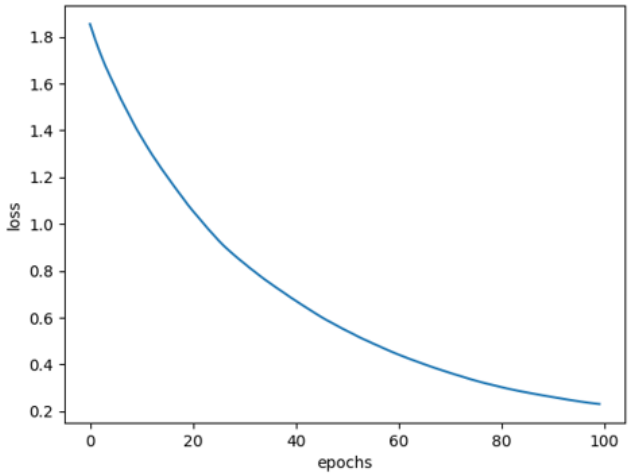

Figure 7 Ada Max Error Function

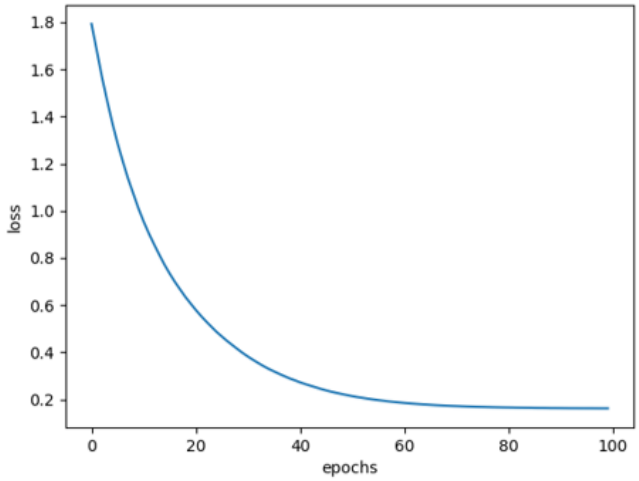

Figure 8 Nadam Error Function

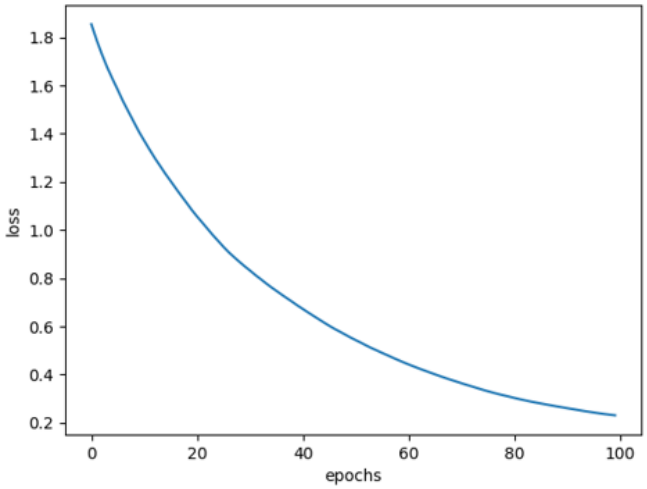

Figure 9 RMSprop Error Function

Case II: No constraint on iterations

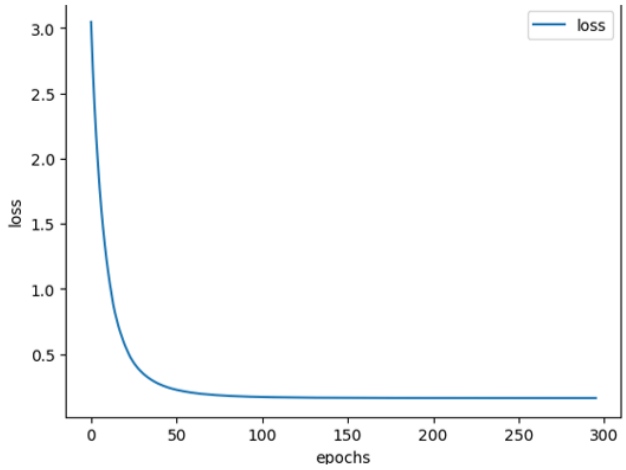

Figure 10 SGD Error Function

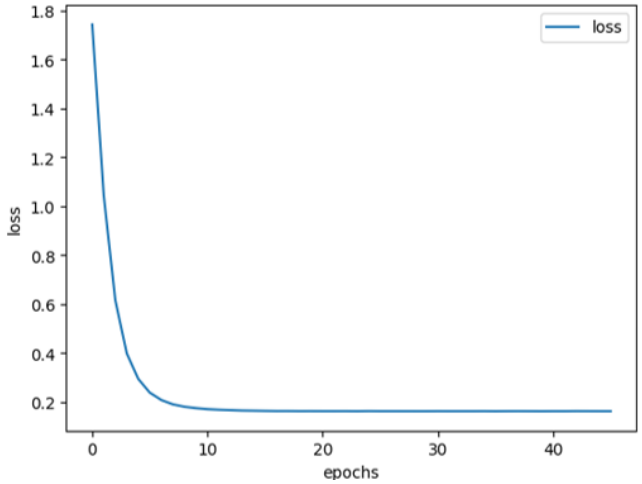

Figure 11 Momentum Error Function

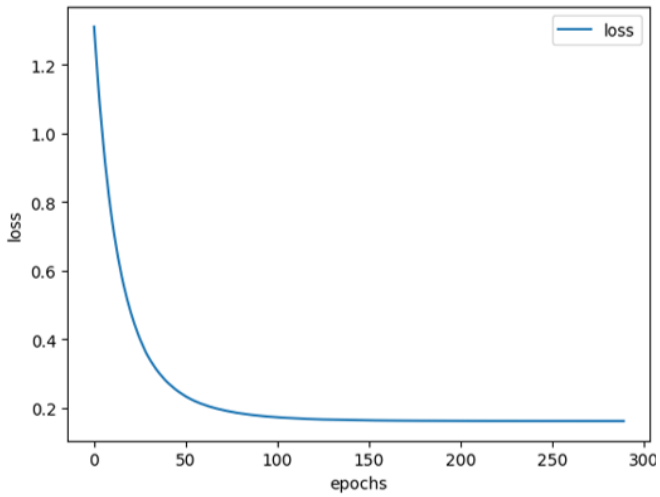

Figure 12 NAG Error Function

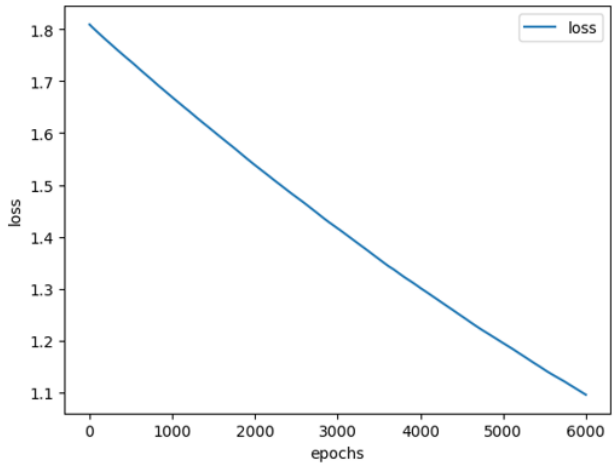

Figure 13 Ada Delta Error Function

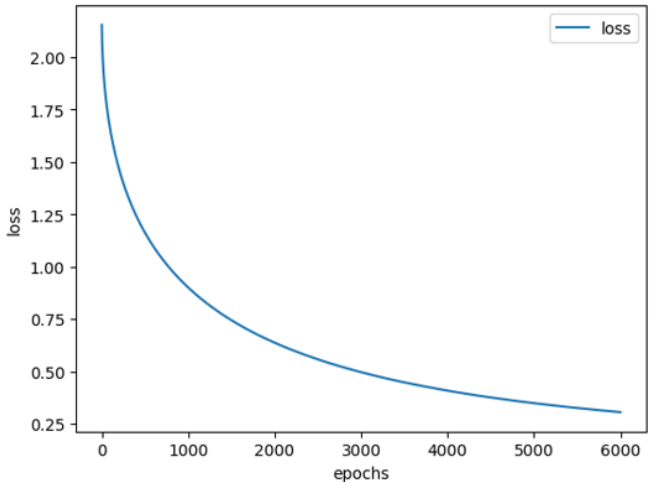

Figure 14 Ada Grad Error Function

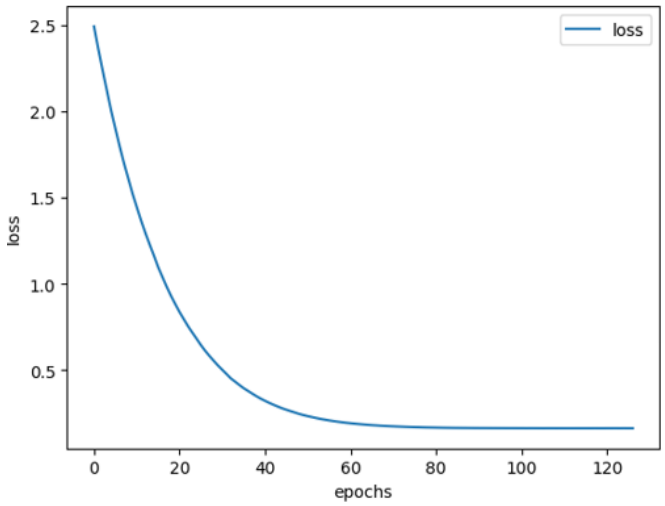

Figure 15 Adam Error Function

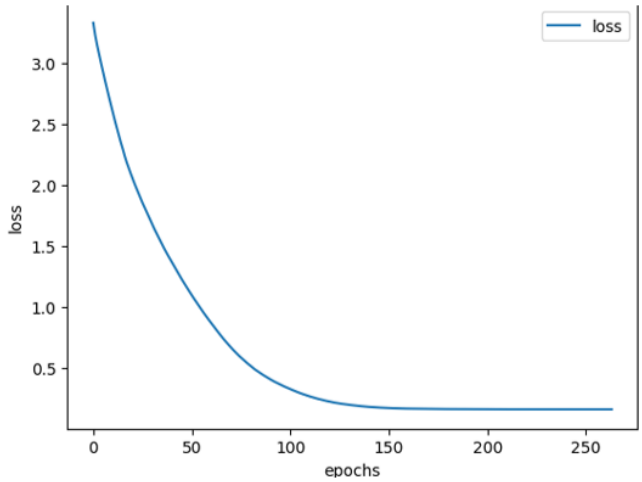

Figure 16 Ada Max Error Function

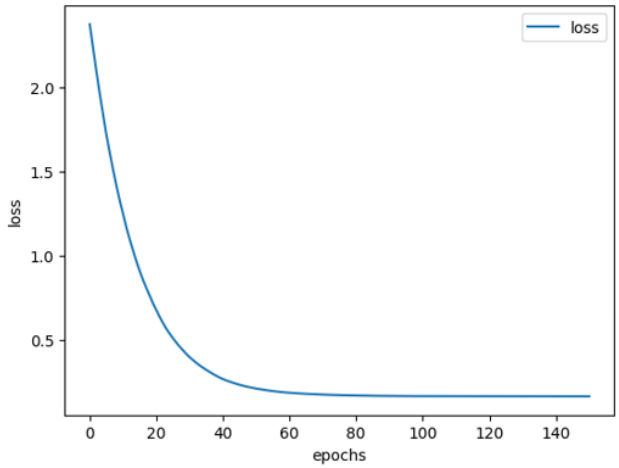

Figure 17 Nadam Error Function

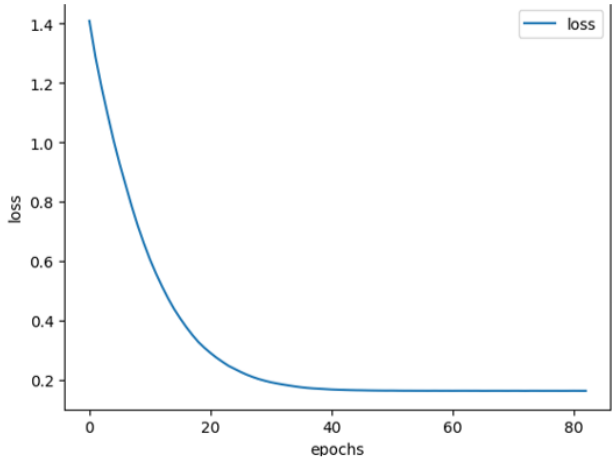

Figure 18 RMSprop Error Function

5. RESULT AND DISCUSSION

Table 2 Comparison of performance of algorithms – Case I

Optimization Algorithm	Number of iterations	MAE	TensorFlow Keras Optimizer
Stochastic	100	0.1687	SGD
Momentum	100	0.1623	SGD
Nesterov	100	0.1788	SGD
AdaGrad	100	1.7982	Adagrad
RMSProp	100	0.1626	RMSprop
Adam	100	0.1623	Adam
AdaDelta	100	2.3780	Adadelta
AdaMax	100	0.2310	Adamax
Nadam	100	0.1629	Nadam

Table 3 Comparison of performance of algorithms – Case II

Optimization Algorithm	Number of iterations	MAE	TensorFlow Keras Optimizer
Stochastic	296	0.1621	SGD
Momentum	46	0.1625	SGD
Nestereov	290	0.1621	SGD
AdaGrad	6000	0.3056	Adagrad
RMSProp	83	0.1626	RMSprop
Adam	127	0.1622	Adam
AdaDelta	6000	1.0958	Adadelta
AdaMax	264	0.1622	Adamax
Nadam	151	0.1624	Nadam

In case I, the observations are made for fixed number of iterations (100). The obtained results (Table 1), show that among the stochastic gradient descent (SGD) variants, both the classic SGD and momentum-based methods achieved similar performance in terms of Mean Absolute Error (MAE). The MAE values obtained were 0.1687 and 0.1623, respectively, indicating that both algorithms effectively minimized the error during the optimization process.

The Nesterov accelerated gradient descent algorithm showed a slightly higher MAE value of 0.1788 compared to the SGD and momentum variants. This suggests that the Nesterov algorithm might not have performed as well in minimizing the error in this experiment.

Among the adaptive learning rate techniques, Adam, RMSProp, and Nadam all achieved similar MAE values around 0.1623-0.1629. This indicates that these algorithms were effective in optimizing the model and producing accurate predictions.

AdaGrad and AdaDelta algorithms, on the other hand, yielded higher MAE values of 1.7982 and 2.3780, respectively. This suggests that these algorithms might not have been as effective in reducing the error compared to other optimization techniques in this experiment.

The AdaMax algorithm achieved an MAE value of 0.2310, which was higher compared to the Adam algorithm but lower than AdaGrad and AdaDelta. This suggests that AdaMax performed reasonably well in this experiment but might not have been as effective as Adam or some other algorithms.

In case II, there is no constraint on number of iterations, however a patience value of 10 iterations is applied to the model. This helps in early stopping the iterations if the value of mean absolute error remains same for 10 continuous iterations. The results of this experiment show that, the stochastic gradient descent (SGD) variants, including classic SGD,

momentum, and Nesterov, all achieved very similar MAE values ranging from 0.1621 to 0.1625. This suggests that these algorithms were effective in minimizing the error, with no significant difference in performance observed among them.

The AdaGrad algorithm required a much larger number of iterations (6000) to achieve a comparable MAE of 0.3056. This indicates that AdaGrad took more time to converge to a similar level of accuracy compared to the SGD variants.

RMSProp, Adam, AdaMax, and Nadam all achieved MAE values close to each other, ranging from 0.1622 to 0.1626. This suggests that these algorithms were effective in optimizing the model and producing accurate predictions, similar to the performance of the SGD variants.

AdaDelta, on the other hand, yielded a higher MAE value of 1.0958, indicating that it might not have been as effective in reducing the error compared to other optimization techniques in this experiment.

6. CONCLUSION

In this study we compared existing gradient descent optimization algorithms to predict the diabetes. We observed that both time and computational capacity play an important role in selection of the optimization algorithm for models. The SGD variants (classic SGD, momentum, and Nesterov), along with RMSProp, Adam, AdaMax, and Nadam, demonstrated relatively better performance in terms of minimizing the error (lower MAE values) compared to AdaGrad and AdaDelta algorithms in both scenarios. However, it is important to note that these conclusions are specific to the experiment conducted, and further analysis and evaluation may be required to generalize these findings to different datasets or problem domains. The performance of optimizers also vary with some other factors like size of dataset and models used for testing.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010, 177-186.

Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. Proceedings of ICML'2013, 1139-1147.

Tieleman, T., & Hinton, G. (2012). Lecture 6

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12, 2121-2159.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR'2015).

Kingma, D. P., & Lei Ba, J. (2017). Adamax: A variant of the Adam optimizer. arXiv preprint arXiv:1412.6980.

H. Tao and X. Lu, "On Comparing Six Optimization Algorithms for Network-based Wind Speed Forecasting," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 8843-8850, doi: 10.23919/ChiCC.2018.8482567.

Aatila Mustapha et al 2021 J. Phys.: Conf. Ser. 1743 01200

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR computational mathematics and mathematical physics, 4(5), 1-17.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. arXiv preprint arXiv:1904.09237.

Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. ICLR Workshop.

Mehmet Akturk, accessed 22 June 2023 https://www.kaggle.com/datasets/mathchi/diabetes-data-set
Bishop, C.M. (2006) Pattern Recognition and Machine Learning. Springer, Berlin.
Dean, J. and Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51, 107-11. https://doi.org/10.1145/1327452.1327492
Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. In: Montavon, G., Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_26
Chandra M and Matthias M 2017 Variants of RMSProp and Adagrad with Logarithmic Regret Bounds (arXiv:1706.05507)
Yazan, E., & TALU, M. F., (2017). Comparison of the Stochastic Gradient Descent Based Optimization Techniques . 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey
Voronov, Sergey & Voronov, Ilya & Kovalenko, Roman. (2018). Comparative analysis of stochastic optimization algorithms for image registration. 123-130. 10.18287/1613-0073-2018-2210-123-130.
Hassan, E., Shams, M.Y., Hikal, N.A. Elmougy Samir. The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study. Multimedia Tools Appl 82, 16591–16633 (2023). https://doi.org/10.1007/s11042-022-13820-0