Optimisation Algorithms for Deep Learning Method: A Review with a Focus on Financial Applications

Nikhil G. Kurup*, Dr. K. S. Vijula Grace

Department of ECE, Noorul Islam Centre for Higher Education, Kanyakumari, India

*Corresponding author's e-mail: nikhilgkurup@gmail.com

doi: https://doi.org/10.21467/proceedings.160.37

ABSTRACT

In a variety of fields including financial applications like stock market analysis deep learning has achieved amazing success in producing precise forecasts. To train deep learning models for financial forecasts, however, is a difficult undertaking that calls for careful consideration of a variety of hyperparameters and optimisation strategies. Optimisation is a technique that is part of mathematics and is used to solve analytical and numerical problems in minimisation and maximisation of functions. It is thus used for getting improved prediction in terms of quality and performance. In this paper we discuss different techniques like SGD, AdaGram and others, that have proven effective in improving the convergence and generalization performance of deep learning models in finance. Here we focus on financial applications where deep learning algorithms are used for the problem solving were optimization is also a part.

Keywords: Optimisation, Deep learning, Stock market

1 Introduction

Even the most seasoned financial gurus find it difficult to make precise predictions since the stock market is a dynamic system that is complex and subject to many different influences. With the introduction of machine learning, however, there has been an increase in interest in utilising sophisticated algorithms to evaluate and project market patterns. In order to better inform investment decisions and increase profitability, machine learning techniques can be used to find patterns and trends in huge historical data sets. In this regard, using machine learning and soft computing algorithms in stock price forecasting has emerged as an important research area [1], [2].

To forecast stock prices and market patterns, many machine learning methods have been used, includes Artificial neural networks, SVM, decision trees etc. These algorithms have shown encouraging results, which has boosted interest in their use in the finance sector. For instance, a study by Jang *et al.* (2019) that used a deep learning algorithm to estimate stock prices showed how good it was at correctly predicting trends. Lee *et al.*, (2019) employed machine learning algorithms to estimate stock market movements and found that they were more accurate than conventional forecasting methods [5], [18].

Although the application of machine learning algorithms to stock market forecasting is still in its preliminary stages, it has the potential to fundamentally change the way investors make choices. These algorithms may aid investors in reducing risk and maximising returns by offering more precise and trustworthy forecasts. As a result, more research and development in this field are probably going to continue, which will eventually result in better investing methods and more profitability [6], [7].

Because deep learning architectures allow computers to learn from enormous amounts of data and make predictions and judgements with incredible precision, they have completely changed the field of artificial intelligence. Training a complex Deep neural network, however, is a computationally demanding task that consumes big amount of time and resources. That being the case, there is a rising need for effective optimisation methods that can reliably and quickly train neural networks.

Proceedings DOI: 10.21467/proceedings.160; Series: AIJR Proceedings; ISSN: 2582-3922; ISBN: 978-81-965621-9-9

Deep learning relies on optimisation techniques to find out the optimum parameters that will minimise the difference between expected and actual values. With each iteration of the training process, these algorithms adjust the weight and bias of the neural network to find ideal set of parameters [3].

For deep learning, several optimisation algorithms have been introduced, each having unique merits and demerits. One of the most used deep learning optimisation technique is gradient descent, and to enhance its performance, versions like stochastic gradient descent (SGD), Adam, and Adagrad have been created.

In addition to conjugate gradient, L-BFGS, and quasi-Newton approaches, other different optimisation algorithms were also applied to deep learning with varied degrees of success. Deep neural networks' generalisation performance and convergence rate have also recently improved by to recent developments in optimisation algorithms, such as second-order techniques.

In conclusion, optimisation algorithms are essential to deep learning systems and have significantly advanced the area of artificial intelligence through research and use. To increase the effectiveness and efficiency of deep learning algorithms, researchers are constantly investigating new optimisation approaches and algorithms.

2 Optimizing Algorthms

2.1 Gradient Descent

For finding the smallest value of a cost function, an optimisation procedure called gradient descent is used. It entails inverse proportionally change a model's parameters with the cost function's gradient corresponding to the parameters [8].

In Gradient Descent (GD) in order to minimise the given function, it uses the gradient (i.e., slope) of the function corresponding to the parameters to iteratively alter the function's parameter values. The GD update rule can be stated as follows:

$k = k - \lambda \nabla J(k)$

Where k is the vector parameter, J(k) the cost function, λ the learning rate and ∇J(k) the gradient of J(k).

2.2 Stochastic Gradient Descent (SGD)

Instead of using the whole training set, SGD a variation of GD, randomly chooses a subset of training samples (also known as a mini-batch) to evaluate the gradient of the cost function. SGD becomes quicker as a result, but the optimisation process also becomes more random as a result [9].

Stochastic gradient descent (SGD), only those data points that were randomly chosen at each iteration were used to compute the gradient. The SGD update rule is written as follows:

$k = k - \lambda \nabla J\_i(k)$

Where i is the index of the randomly selected data point or batch, and ∇J_i(k) is the gradient value of J(k) corresponding to k using only the selected data points.

2.3 Mini-Batch Gradient Descent

A middle ground between SGD and GD is a known mini-batch gradient descent method. Instead of taking the full training set, it computes cost function gradient on very small randomly taken batches of training samples to strike a balance between convergence speed and computing efficiency [9].

A compromise between GD and SGD, Mini Batch Gradient Descent (MBGD) computes the gradient using a few batch of data points at each iteration. The MBGD update rule can be stated as follows:

$k = k - \lambda \nabla J\_B(k)$

Where B is size of batch, and ∇J_B(k) is the gradient of J(k) with respect to k using only the selected batch of data points.

Proceedings DOI: 10.21467/proceedings.160
ISBN: 978-81-965621-9-9

Series: AIJR Proceedings
ISSN: 2582-3922

2.4 Momentum

Gradient Descent's convergence speed and stability are increased using the momentum approach. It introduces a momentum term that, by leveraging knowledge from earlier gradient updates, quickens the optimisation process [10]. A GD variant known as momentum leverages the gradients' moving average to speed convergence and tame oscillations. Momentum's update rule is as follows:

v =g v + (1 - g) ∇J(k) k = k - λv

Where g is the momentum value, which controls the weight of the previous gradient, v is the velocity vector, and λ is the learning rate.

2.5 Nesterov Accelerated Gradient (NAG)

By performing a "look ahead" before calculating the gradient of the cost function, Nesterov Accelerated Gradient (NAG), an extension of Momentum, speeds up the convergence of the optimisation process [11].

Momentum's Nesterov Accelerated Gradient (NAG) uses a "lookahead" update to more accurately predict the future gradient. The NAG update rule can be stated as follows:

v = gv + (1 - g) ∇J (k - gv) k = k - λv

where k - gv is the "lookahead" position

2.6 Adagrad

An adaptive learning rate technique called Adagrad dynamically modifies the learning rate according to how frequently the parameters are updated. For values that are changed frequently, it slows down learning rate while speeding it up for those that are updated infrequently [12].

Based on the information from previous gradients learning rate of all parameters get adjusted. AdaGrad's update rule can be stated as follows:

G = G + ∇J(k)^2 k = k - (λ / √G + ε) ∇J(k)

Where G is the historical sum of squared gradients, ε is the constant that is used to avoid the case of division by zero, and √G is the element-wise square root of G.

2.7 Adadelta

A moving average of gradients and the moving average of squared updates are introduced in the AdaDelta form of Adagrad to address the problem of the learning rate declining too quickly over time [13]. By employing the moving average of historical gradients rather than the historical sum of squared gradients, AdaDelta, a variation of AdaGrad, increases the adaptiveness even more. AdaDelta's update rule can be stated as follows:

E [y2] t = gE[y2] {t-1} + (1 - g)∇J(k2)2

Δkt = - (√E[Δk2] {t-1} + ε) / (√E[y22]t + ε)

∇J(k) k = k + Δkt

where E[y2]t is the moving average value of the historically calculated squared gradients, E[Δk2]t is the moving average value of the historically calculated squared updates, g is the decay rate.

2.8 ADAM

The advantages of Momentum and Adagrad are combined in the well-known optimisation method Adam. The learning rate and the momentum term are adaptively adjusted using a moving average of the gradients and the squared gradients [4].

ADAM (Adaptive Moment Estimation) combines concepts from Momentum and AdaGrad to improve performance on a variety of optimisation problems.

The equations for updating parameters can be e given as follows:

mt = g1 * mt-1 + (1 - g1) * yt

vt= g2 * vt-1 + (1 - g2) * yt 2

mhat_t = mt / (1 - g1t)

vhat_t = vt / (1 - g2t)

kt = kt-1 - λ * mhat_t / (√vhat_t + ε)

Where kt is the parameter vector at iteration moment t.yt is the gradient vector at iteration t. λ is the learning rate. g1 and g2 are hyperparameter values that manage the decay rates of moving average of m and v respectively. mhat_t and vhat_t are bias corrected estimates of the primary and secondary moment values of the gradients, respectively. ε is used to avoid the case of division by zero problem.

Adam calculates first two moment values of the gradients with the moving averages. While the second moment estimate vt is comparable to the historical sum of squared gradients used in AdaDelta and the AdaGrad , the first moment estimate mt is much the same to the momentum term used in the Momentum algorithm. The initialization bias of the moving averages, which is more obvious at the start of training when the moving averages are initialised to zero, is corrected for using the bias-correction terms mhat_t and vhat_t.

Adam is superior to other optimisation algorithms in a number of ways, including the fact that it uses less memory and is computationally efficient. It works effectively for issues with huge datasets and highly dimensional parameter spaces. It can withstand gradients that are noisy or sparse. Compared to other algorithms, it is less dependent on the selection of hyperparameters.

2.9 Adamax

Adamax is similar to Adam which varies in the fact that it uses the gradient's L infinity norm as opposed to their squared gradients.

2.10 Rmsprop

Another adaptive learning rate approach, RMSProp, modifies the learning rate using the moving average value of the squared gradients [15]. Using the Nesterov Accelerated Gradient, Nadam extends Adam's optimisation capabilities. In comparison to Adam, it delivers faster convergence and superior generalisation performance.

2.11 NADAM

A variation of Adam called NADAM (Nesterov accelerated ADAM) combines the concepts of Adam and NAG to offer faster convergence and improved generalisation. Like Adam, NADAM updates the parameters using the first two moment estimations of gradient but utilises a different update rule that takes Nesterov Momentum into account [16].

The NADAM update rule might be stated as:

mt = g1 * mt-1 + (1 - g1) * yt

vt = g2 * vt-1 + (1 - g2) * yt2

mt_hat = mt / (1 - g1t)

vt_hat = vt / (1 - g2t)

kt = kt-1- λ* [(1 - g1) * yt / (1 - g1t) + gt * mt_hat / (1 - g1t) + g2 * √vt_hat / (1 - g2t+ε)]

Proceedings DOI: 10.21467/proceedings.160
ISBN: 978-81-965621-9-9

Series: AIJR Proceedings
ISSN: 2582-3922

Where λ is the learning rate. g1 and g2 are the decay rates for the first two moment estimates, respectively.yt is the gradient at time t.mt and vt are the exponentially weighted moving averages of squared gradient and the gradient, respectively. mt_hat and vt_hat are bias-corrected estimates of the first two moment estimates. kt and kt-1 are the updated and the prior parameter values, respectively. ε is a very small constant that is inserted to avoid the problem of division with zero.

The updation of initial instant estimate is where NADAM and Adam diverge. While NADAM corrects the first moment estimate by conducting a lookahead step using the most recent estimates of the primary and secondary moments, Adam updates the initial moment estimate using the unbiased one of the gradient.

The optimizer can more accurately follow the curvature of the loss surface thanks to this lookahead step. To further enhance the optimisation procedure, NADAM also employs an Adam's second moment estimate that is bias corrected. Generally speaking, NADAM is a strong optimizer that combines the advantages of Adam and Nesterov Momentum to offer quicker convergence and improved generalisation.

2.12 EADAM

The extragradient method and adaptive momentum are combined in the optimisation algorithm EADAM (Extragradient with Adaptive Momentum), which effectively resolves non-convex optimisation issues. Escape from High-Dimensional Non-Convex Mazes with Adaptive Momentum [14, 22].

During the optimisation process, the momentum parameter is adaptively adjusted using a technique called adaptive momentum. The contribution of the prior gradient to the current update is controlled by the momentum parameter. Adaptive momentum techniques can efficiently balance the exploration and exploitation of the search space while accelerating convergence by modifying the momentum parameter. Extragradient and momentum phases make up the two primary steps of the EADAM algorithm.

The final update is obtained using the momentum step after an intermediate point has been obtained using the extragradient step. The EADAM update rule is stated as:

Intermediate update:

$ki+1 = Proj\_Ω\ (ki - λ∇f(ki))$

Momentum update: $ki+1 = ki + g\ (ki+1 - ki)$

where ki is the current vector, ki+1 is the intermediate vector, ∇f(ki) is the gradient vector of the objective function f at ki, λ is the step size, g is the momentum parameter, and Proj_Ω (ki) is the projection of ki onto the feasible set Ω.

There are multiple EADAM variations that alter the momentum update or alter the momentum parameter using various methods. These variations consist of:

A condensed variant of EADAM that omits the projection step and employs an extragradient step with a fixed step size is called EAM (Extragradient with Adaptive Momentum).

Extra gradient with Adaptive Momentum and Stochastic Gradient Descent (EAMSGD), is a variation of EAM that calculates the gradient in the extragradient step using stochastics gradient descent method.

Extragradient with Adaptive Momentum and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (EAM-LBFGS): A variation of EAM that adjusts the momentum parameter and approximates the Hessian matrix using the L-BFGS approach.

A variation of EAM called EAM-BB (Extragradient with Adaptive Momentum and Barzilai-Borwein) uses the Barzilai-Borwein method to modify the momentum parameter in light of prior updates.

A variation of EAM that uses AdaGrad to adaptively change the extragradient step's step size is known as EAM-AdaGrad (Extragradient with Adaptive Momentum and AdaGrad).

3 Literature Review

Quadir *et al.* used adam optimizer for optimizing a multi layered sequential LSTM [17]. Saurabh *et al.* in their work based on LSTM-RNN for the prediction of stock prices future values tested with different optimizers and found adam optimizer came out to be the best among them [27]. Das *et al.* used adam optimizer along with a multilayer perceptron for the prediction of stock values [28].

Rana et.al used different optimizers and compared the results using LSTM recurrent networks [19]. Bhandari *et al.* used Adam, adagrad, Nadam along with a LSTM network [20]. Rao *et al.* in his paper suggested that GRU with gradient descent and adam optimizer produces better results [29]. Deepika *et al.* used Artificial bee colony algorithm as the optimizer algorithm and used with LSSVM for stock trend anticipation [21].

Upadhyay et.al used market sentiment as a data and adam optimizer for optimization and LSTM as the algorithm for prediction [30]. Aiswarya *et al.* used adam optimizer in a machine learning model for stock price forecasting [31]. Rekha *et al.* used RNN, LSTM and GRU algorithms for the stock market prediction and Adam optimizer was used as the optimizer in these setups [23]. Jiang *et al.* used different methods including Adam optimizer in Bitcoin price prediction along with LSTM and GRU [24].

Kamalov *et al.* used RMSprop and compared with SGD and Adam optimizers in deep learning algorithm for stock price estimation [25]. Chen in his work on forecasting financial and economic market used RMSProp as the optimizer [26].

4 Results and Discussion

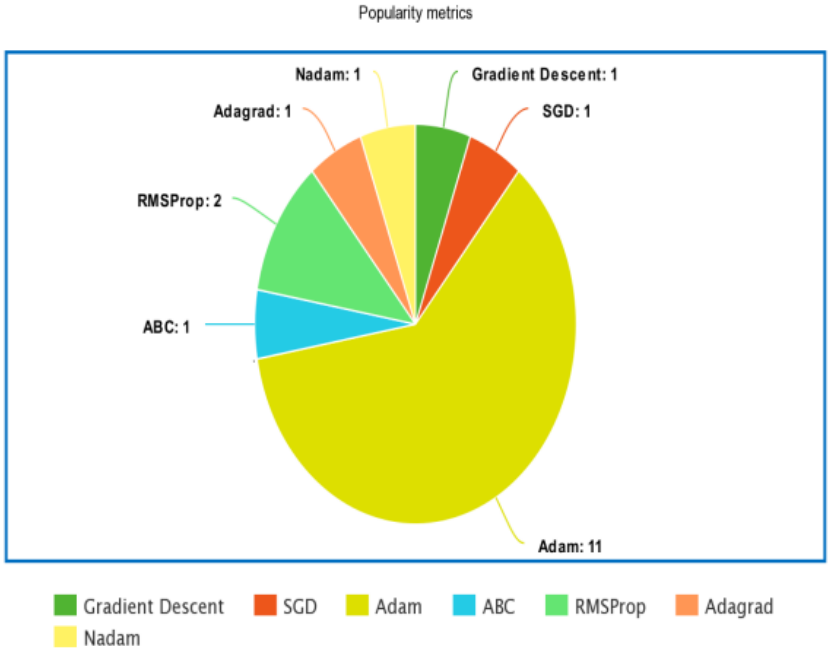

Figure 1: *Popularity Metrics of the methods*

Taking account of the works that are considered in the study it can be seen from Figure 1 that Adam algorithm is the most used and is considered as the pioneer method for optimising the deep learning parameters when they are used in financial applications.

5 Conclusion

This paper concludes by reviewing the latest trends in optimizing machine learning algorithms especially deep learning which are used in financial applications. In the latest researches it is found that Adam algorithm and its variants are

Proceedings DOI: 10.21467/proceedings.160
ISBN: 978-81-965621-9-9

Series: AIJR Proceedings
ISSN: 2582-3922

much used. most algorithms are giving a good result after optimizing. But there is still room for improvements in the research as the problem of getting stucked in local minima is still there and finding a good approach to make the loss function ideal is still open to improvement.

6 Publisher's Note

AIJR remains neutral with regard to jurisdictional claims in institutional affiliations.

How to Cite

Kurup & Grace (2023). Optimisation Algorithms for Deep Learning Method: A Review with a Focus on Financial Applications. *AIJR Proceedings*, 289-296. https://doi.org/10.21467/proceedings.160.37

References

[1] Jang, D., Kim, J., & Kang, J. (2019). Stock price prediction using deep learning on Korean stock market index. Journal of Information Processing Systems, 15(6), 1386-1395.
[2] Lee, D., Kim, D., & Kim, H. (2019). A study on stock price prediction using machine learning. International Journal of Advanced Science and Technology, 28(13), 457-466.
[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
[4] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[5] Martens, J., & Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 1033-1040).
[6] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408.
[7] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
[8] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
[9] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Springer.
[10] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5), 1-17.
[11] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k^2) . In Doklady ANSSSR (Vol. 269, No. 3, pp. 543-547).
[12] Duchi, J., Hazan, E., & Singer, Y. (2011). "Adaptive subgradient methods for online learning and stochastic optimization". Journal of Machine Learning Research, 2121-2159.
[13] Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701.
[14] Luo, P, & Ding J. (2021)."eAdam: Escaping from saturated solutions". arXiv preprint arXiv:2102.12898.
[15] Tieleman, T, & Hinton, G. (2012). "Lecture on rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning, 26-31.
[16] Dozat, T. (2016). Incorporating Nesterov momentum into Adam. arXiv preprint arXiv:1609.04747.
[17] Abdul Quadir Md, Sanjit Kapoor, Chris Junni A.V,"Novel optimization approach for stock price forecasting using multi-layered sequential LSTM" , Applied Soft Computing,Volume 134,2023,109830,ISSN 1568-4946,
[18] Wang, Qiaoyu & Kang, Kai & Zhang, Zhihan & Cao, Demou. (2021). "Application of lstm and CONV1D lstm Network in Stock Forecasting Model". Artificial Intelligence Advances. 3. 10.30564/aia.v3i1.2790.
[19] Rana, Masud & Uddin, Mohsin & Hoque, Md. (2019). Effects of Activation Functions and Optimizers on Stock Price Prediction using LSTM Recurrent Networks. 354-358. 10.1145/3374587.3374622.
[20] Hum Nath Bhandari, Keshab R. Dahal Ramchandra Rimal. Binod Rimal, Nawa Raj Pokhrel,"Predicting stock market index using LSTM" ,Machine Learning with Applications, Volume 9,2022,100320,ISSN 2666-8270,
[21] Deepika, N., Nirupamabhat, M. (2020). An optimized machine learning model for stock trend anticipation. Ingénierie des Systèmes d'Information, Vol. 25, No. 6, pp. 783-792. https://doi.org/10.18280/isi.250608
[22] Liu, Huicheng. (2018). Leveraging Financial News for Stock Trend Prediction with Attention- Based Recurrent Neural Network. arXiv:1811.06173
[23] KS Rekha and S. MK, "Stock Market Prediction Using Deep Learning techniques," International Conference on Communication, Control and Information Sciences (ICCISc), India, 2021, pp. 1-6, doi: 10.1109/ICCISc52257.2021.9484960
[24] Jiang, X. (2020). Bitcoin Price Prediction Based on Deep Learning Methods. Journal of Mathematical Finance.
[25] Kamalov, Gurrib. I, Smail, L., and "Stock price forecast with deep learning", , 2021. doi:10.48550/arXiv.2103.14081.
[26] Fan Chen. "Deep Neural Network Model Forecasting for Financial and Economic Market." 2022 Hindawi
[27] Saurabh, Nikitha. (2020). LSTM -RNN Model to Predict Future Stock Prices using an Efficient Optimizer. 7. 672-677.
[28] Das, S., & Mishra, S. (2019). "Advanced deep learning framework for stock value prediction". International Journal of Innovative

Technology and Exploring Engineering, 8(10), 2358–2367.

[29] Joshi, Kalyani & N, Bharathi & Rao, Jyothi. (2016). Stock Trend Prediction Using News Sentiment Analysis. International Journal of Computer Science and Information Technology. 8. 67-76. 10.5121/ijcsit.2016.8306.

[30] Yash Upadhyay, Nitin Dixit, Vinayak Pujari ."stock price prediction using lstm and markets sentiment analysis", International Journal of Emerging Technologies and Innovative Research ISSN:2349-5162, Vol.8, Issue 5, page no. ppf734-f738,2021

[31] N. aiswarya & N. divya "stock price forecasting: a machine learning model". International Journal of Engineering Applied Sciences and Technology, 2020, Issue 4, Vol. 5, ISSN No. 2455-2143.

Proceedings DOI: 10.21467/proceedings.160
ISBN: 978-81-965621-9-9

Series: AIJR Proceedings
ISSN: 2582-3922