Student Name: Hassan Nauman

Student Number: 7871016

Course: COMP 3190 – A01

Research Project

Fall 2023

A Comparative Analysis of Stochastic Gradient Descent (SGD) and Adam: Optimization Algorithms in Neural Networks and Backward Propagation

Date: 12/6/2023

Content

1)Abstract

2)Introduction to Neural network

3) Backward Propagation in Neural Networks

4) Practical Applications of Neural Networks

5) Optimization Algorithms: Stochastic Gradient Descent (SGD) vs. Adam Optimizer

6) Comparison between SGD and Adam Optimizer

7) Hybrid Optimization Strategies

8) Conclusion

Abstract

Neural networks are powerful mathematical tools used for many purposes and can help computers make intelligent decisions with limited human assistance (Amazon Web Services, 2023). This paper conducts a comprehensive examination of the optimization algorithms employed in training neural networks, focusing on the widely used Stochastic Gradient Descent (SGD) and Adam optimizer. The paper begins with an overview of neural networks and places particular emphasis on the role of backward propagation, a cornerstone in supervised training. The paper also introduces the practical applications of neural networks. The core of the paper centers on a detailed exploration of the SGD and Adam optimization algorithms. Each optimizer is individually introduced. Additionally, this paper extends to provide insights into the distinctive strengths and weaknesses inherent in each optimizer. The comparative analysis evaluates these algorithms based on critical factors such as convergence speed, training accuracy, etc. In addition to the comparison, the paper explores the concept of hybrid optimization, where both SGD and Adam optimizers are strategically combined. The conclusion synthesizes the findings, providing a summary of the comparative analysis and implications for the optimization of neural networks. This research paper serves as a valuable guide for practitioners and researchers navigating the intricate landscape of neural network optimization.

1) Introduction to Neural network

In the book, An Introduction to Neural Networks, neural networks are intricately explored as interconnected assemblies of basic processing elements, units, or nodes. Inspired by the biological neurons in animals, these networks derive their computational strength from the adaptive connection strengths, or weights, shaped through learning from specific training patterns. Neural networks have evolved beyond mere statistical tools, providing alternative approaches to conventional methods like nonlinear regression and cluster analysis, particularly excelling in statistical modeling and data analysis for classification and forecasting (International Neural Network Society et al., 1988).

The historical context of neural networks dates back to 1943, when McCulloch and Pitts modeled neurons using electrical circuits, followed by Hebb's emphasis on strengthened neural pathways in 1949. The 1950s witnessed challenges in simulating neural networks, but by 1959, Widrow and Hoff introduced the ADALINE and MADALINE models. Despite a decline in interest in the 1960s, the late 1970s and 1980s saw a resurgence with multilayered networks and bidirectional lines. The breakthrough came in 1986 with back-propagation networks, applied across various industries. From 2009 to 2012, Schmidhuber's research team advanced recurrent and deep feedforward neural networks. Ongoing research emphasizes hardware optimization, marking the transition from theoretical concepts to practical applications (Macukow, 2016).

2) Backward propagation in neural networks

In supervised training, a neural network processes inputs, compares actual outputs to expected outputs, and adjusts weights through a process referred to as the back-propagation algorithm.

This algorithm, highly crucial for training neural networks, involves propagating errors backward through the network and iteratively refining the weights controlling the network until errors are minimized. The back-propagation algorithm is particularly associated with training multiple-layer perceptron neural networks also known as MLPs (Liu and Mander, 2010). A backpropagation network is characterized by a layered hierarchical architecture comprising uncomplicated neurons (nodes) intricately connected across different layers. In this specific configuration, termed "simple" and non-recursive, the network strictly permits connections solely between layers, specifically along the synapses (Wythoff ,1993)

To emphasize the importance of backpropagation, an illustrative example that showcases its significance in neural network training is a quadratic loss function. The per-sample loss function (F0) is defined, involving pre- and post-activation variables. The key steps include computing backpropagated errors and partial gradients during both a forward and backward pass. The forward pass involves recursively computing post-activation variables from the bottom to the top layer, storing them for future use. In the backward pass, errors are computed, and partial gradients overweight matrices are obtained. The algorithm optimizes the network using stochastic gradient descent (SGD), updating weights after each backward pass. Backpropagation is a crucial aspect of the learning algorithm, contributing to the iterative optimization of neural networks (Sun, 2019)

3) Practical applications

The practical applications of neural networks are vast and extend across various fields due to their ability to recognize patterns and make predictions based on training data. One notable application is in the classification of data, exemplified by the following wine classification scenario. Neural networks can be trained on a dataset containing characteristics of wines and their corresponding types, enabling the system to predict the types of new, unseen wines. This classification capability is immensely valuable in industries such as quality control in winemaking. Furthermore, neural networks find extensive use in predictive analysis. Once trained, they can predict outputs for future data within the provided domain. This predictive power is harnessed in fields like finance for stock market predictions, in healthcare for disease diagnosis, and in weather forecasting for predicting climate patterns. (Oken, 2017)

Revisiting our paper's title, we pivot from exploring the practical applications of neural networks to a detailed examination of optimization algorithms—specifically, Stochastic Gradient Descent (SGD) and Adam.

4) Optimization Algorithms: Stochastic Gradient Descent (SGD)

Stochastic gradient algorithms are commonly employed for optimizing functions, specifically addressing problems like empirical loss minimization in supervised learning. The objective is to minimize a function ,$f(x)$, representing the expectation of $f\gamma(x)$, where $f\gamma(x)$ denotes the loss function for a given training sample identified by γ. The conventional gradient descent (GD) scheme, $xk+1=xk-\eta\nabla Ef\gamma(xk)$, encounters computational challenges when dealing with the gradient of an expectation. To alleviate these challenges, the stochastic gradient descent (SGD) algorithm is introduced. In its basic form, SGD substitutes the expectation of the gradient with a sampled gradient, resulting in the iteration $xk+1=xk-\eta\nabla f\gamma k(xk)$. Here, γk is an independent and identically distributed random variable sharing the same distribution as γ. This algorithm provides a computationally more manageable solution, particularly beneficial for tasks involving large datasets (n)is large), such as empirical risk minimization in supervised learning. (Li et al., 2019). In a supervised learning setup, SGD updates parameters for each training example, making it faster and suitable for online learning. It performs frequent updates with large variance, leading to significant variability in the objective function. This variability allows SGD to explore new and potentially superior local minima. However, as SGD overshoots, achieving convergence to the exact minimum becomes more challenging without an appropriate learning rate. The reduction of the learning rate gradually aligns SGD's convergence behavior with batch gradient descent, ensuring convergence to a local or global minimum (Alagözlü , 2022).

SGD, or Stochastic Gradient Descent, stands out as an optimization algorithm that optimizes the loss function on a random sample of training data in each iteration, providing a notable advantage in terms of accelerated parameter update speed. This approach offers efficiency benefits, but it comes with inherent trade-offs that warrant consideration. In the context of discussing the pros and cons of SGD, this strategy allows for quicker updates, enhancing computational efficiency. However, the random optimization on subsets introduces challenges related to convergence and potential overshooting, factors that need careful management for optimal performance in neural networks (Alagözlü, 2022).

5) Optimization Algorithms: Adam Optimizer

The Adam optimization method, introduced by Kingma and Ba (2014), has become widely utilized in stochastic optimization, Adam is a highly efficient stochastic optimization method that operates with minimal memory. This technique calculates adaptive learning rates for distinct parameters based on estimates derived from the first and second moments of gradients. The name "Adam" itself reflects its adaptive moment estimation approach (Kingma and Ba, 2014). Moreover, Kingma and Ba (2014), also mention that Adam offers a stable gradient descent process, making it well-suited for a wide range of non-convex optimization problems, especially those involving large datasets and high-dimensional spaces. However, it may face challenges in achieving convergence in certain cases (Kingma and Ba, 2014). The Adam algorithm incorporates both first-order momentum, which preserves the historical gradient direction, and second-order momentum, which sustains the adaptive state of the learning rate. Additionally, it directly addresses a sequential setting, where samples are presented one after another, as opposed to assuming the availability of a large number of pre-existing training samples. These characteristics contribute to the commendable performance of the Adam algorithm, marked by not only high computational efficiency but also, as previously noted, minimal memory requirements (Liu et al., 2023). The Adam algorithm stands out as a popular choice for tackling parameter optimization challenges due to its efficient computation, fewer tuning parameters, and broad applicability. Nevertheless, this algorithm is not without its drawbacks. One notable limitation is the sluggish convergence speed of the model. This can be attributed to the first-order momentum in the Adam algorithm, characterized by the exponentially weighted average of historical gradients. This momentum, crucial for directing optimization updates, is highly susceptible to the influence of gradient deviation values, resulting in suboptimal searchability and a tardy convergence pace (Liu et al., 2023).

6) Comparison between SGD and Adam Optimizer

Table 1 shows the results from the CNN training on the Iris dataset using various gradient descent methods.

	training accuracy	validation accuracy	epochs	running time (3000 epochs)
SGD	94%	94%	3000	126s
ADAM	94%	98%	300	142s

Table 1. Comparison of different variants on CNN

In the training process of a Convolutional Neural Network (CNN) on the Iris dataset, different gradient descent methods were employed, yielding varied results. Stochastic Gradient Descent (SGD) demonstrated a training accuracy of 94%, matching the validation accuracy, after 3000 epochs with a relatively efficient running time of 126 seconds. On the other hand, the ADAM optimizer achieved a comparable training accuracy of 94%, but notably outperformed in validation accuracy, reaching 98% within a significantly reduced epoch count of 300. However, this improvement in accuracy came at the cost of a slightly longer running time of 142 seconds. The results suggest that while both optimizers achieved high accuracies, ADAM displayed a quicker convergence to a superior validation accuracy, emphasizing its efficiency in this particular training scenario. The results suggest that, in this specific scenario, ADAM appears to be a more effective optimization algorithm for achieving higher accuracy with reduced computational effort (Alagözlü, 2022).

7) Hybrid optimization strategy

Hybrid optimization techniques employ a dynamic approach during the compilation process to selectively choose from a diverse set of optimization algorithms, all designed to enhance the performance of a given piece of code. This dynamic selection is based on heuristics, which act as predictive tools to determine the most suitable algorithm for a specific code segment. By leveraging these heuristics, hybrid optimization aims to strike a balance between different optimization strategies, adapting its choice according to the unique characteristics and requirements of the code at hand. This adaptive and heuristic-driven approach enables the compiler to make informed decisions, ultimately optimizing the overall efficiency and execution speed of the compiled software (Cavazos, Moss, & O'Boyle, 2006).

According to the study by Landro, Gallo, and La Grassa, a combined optimization method involving ADAM and SGD shows promising results where MAS, a novel optimization method, was introduced which designed to automatically amalgamate the benefits of ADAM, an adaptive method, with those of SGD throughout the entire learning process. The MAS algorithm combines displacements on the loss function surface from ADAM (w1) and SGD (w2), resulting in a combined displacement (w1 + w2). The algorithm is elucidated in Algorithm 3, with only two hyper-parameters, λa and λs, governing the contributions of ADAM and SGD, respectively. While a single learning rate η is used for both ADAM and SGD in experiments, the option to differentiate the learning rates exists. Additionally, MAS requires the typical hyper-parameters of SGD and ADAM, assuming the use of mini-batch gradient descent in this context. This approach divides the training dataset into small batches to compute model errors and update coefficients wk. The MAS algorithm calculates the contribution from both ADAM and SGD for each mini-batch, updating coefficients accordingly in three subsequent subsections. To provide clarity on MAS behavior, we present a toy example illustrating its main characteristics (Landro, Gallo, & La Grassa, 2020)

In another paper, The SoftAdam optimizer, a novel hybrid of SGD and Adam update steps, demonstrates superior or comparable performance to traditional optimization methods across diverse tasks. Empirical results, using default parameters ($\eta = 1$), showcase its effectiveness in image classification and language modeling tasks, surpassing both SGD and Adam. Testing on various deep learning problems maintains strong performance at the default setting. SoftAdam's adaptability with default parameters suggests its efficiency without extensive tuning, and its unique ability to create a learning schedule on η opens new research avenues. The algorithm's potential for faster and more accurate training of larger models makes it a valuable addition to optimization techniques ("SOFTADAM: UNIFYING SGD AND ADAM," 2020)

In summary, the exploration of hybrid optimization strategies, exemplified by MAS and SoftAdam, reflects a broader trend in dynamically selecting and blending diverse optimization algorithms based on specific task characteristics. These approaches showcase adaptability and efficiency, emphasizing the significance of simplicity and default settings. The MAS algorithm's minimalistic design and SoftAdam's superior performance with default parameters underscore the industry's pursuit of more intelligent and self-adjusting optimization processes. This trend signals a paradigm shift toward versatile, efficient, and adaptable optimization methodologies, promising enhanced efficiency and speed in compiled software and machine learning model training.

8) Conclusion:

In conclusion, this research paper explores into the intricacies of neural networks, emphasizing their historical evolution, fundamental components, and practical applications across diverse fields. The backbone of supervised training, backward propagation, is explored showcasing its pivotal role in optimizing neural networks through iterative weight adjustments. The paper then shifts focus to optimization algorithms, with a comprehensive analysis of two prominent methods—Stochastic Gradient Descent (SGD) and the Adam optimizer. The strengths and weaknesses of each algorithm are thoroughly examined, paving the way for a comparative analysis based on critical factors such as convergence speed and training accuracy. The exploration extends to a hybrid optimization strategy, combining the benefits of both SGD and Adam. The comparative analysis on a Convolutional Neural Network (CNN) training scenario underscores the efficiency of the Adam optimizer in achieving higher validation accuracy with reduced computational effort. Furthermore, the insights gained from hybrid optimization strategies, such as MAS and SoftAdam, underscore their potential to elevate efficiency and adaptability in neural network optimization. This paradigm shift towards intelligent, self-adjusting techniques holds promise for broader applications, marking a progressive evolution in the field of optimization methodologies.

This research paper serves as a valuable guide for practitioners and researchers navigating the intricate landscape of neural network optimization. It not only enhances understanding of the theoretical underpinnings but also provides practical insights into the application of optimization algorithms. As the field of neural networks continues to evolve, the findings presented herein contribute to the ongoing dialogue surrounding optimization methods, opening avenues for further exploration and refinement in the quest for more efficient and effective neural network training.

References

Alagözlü, M. (2022). *Stochastic Gradient Descent Variants and Applications.* https://doi.org/10.13140/RG.2.2.12528.53767

Amazon Web Services. (2023). Neural Network. AWS. Retrieved from https://aws.amazon.com/what-is/neural-network/ © 2023, Amazon Web Services, Inc.

Cavazos, J., Moss, J.E.B., O'Boyle, M.F.P. (2006) 'Hybrid Optimizations: Which Optimization Algorithm to Use?'. In: Mycroft, A., Zeller, A. (eds) Compiler Construction. CC 2006. Lecture Notes in Computer Science, vol 3923. Springer, Berlin, Heidelberg. Available at: https://doi.org/10.1007/11688839_12

International Neural Network Society, European Neural Network Society, & Nihon Shinkai Kairoku Gakkai. (1988). Neural networks. Oxford ; New York: Pergamon.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv*, arXiv:1412.6980. [Google Scholar]

Landro, N., Gallo, I., La Grassa, R. (Year of Publication), 'Mixing ADAM and SGD: a Combined Optimization Method', University of Insubria, Available at : https://arxiv.org/pdf/2011.08042.pdf

Li, Q., Tai, C. and Weinan, E., 2019. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. The Journal of Machine Learning Research, 20(1), pp.1474-1520.\

Liu, H.-W., & Mander, L. (Eds.). (2010). Comprehensive Natural Products II. Elsevier Ltd. ISBN 978-0-08-045382-8.

Liu, M., Yao, D., Liu, Z., Guo, J., & Chen, J. (2023). An Improved Adam Optimization Algorithm Combining Adaptive Coefficients and Composite Gradients Based on Randomized Block Coordinate Descent. Computational Intelligence and Neuroscience, 2023, 4765891. https://doi.org/10.1155/2023/4765891

Macukow, B. (2016). "Neural Networks – State of Art, Brief History, Basic Models and Architecture." In: Saeed, K., Homenda, W. (eds) Computer Information Systems and Industrial Management. CISIM 2016. Lecture Notes in Computer Science, vol 9842. Springer, Cham. https://doi.org/10.1007/978-3-319-45378-1_1

Oken, A. (2017). An Introduction To and Applications of Neural Networks. Retrieved from https://www.whitman.edu/Documents/Academics/Mathematics/2017/Oken.pdf

OpenReview. (2020). SOFTADAM: UNIFYING SGD AND ADAM FOR BETTER. Retrieved from https://openreview.net/pdf?id=Skgfr1rYDH

Sun, R. (2019). Optimization for deep learning: Theory and algorithms. arXiv, arXiv:1912.08957. [Google Scholar]

Wythoff, B. J. (1993). Backpropagation neural networks: A tutorial. Chemometrics and Intelligent Laboratory Systems, 18(2), 115-155. https://doi.org/10.1016/0169-7439(93)80052-J.