Learning Dynamics of Gradient Descent Optimization in Deep Neural Networks

A Project Report

submitted by

NANDHAKUMARAN M (ME21B2008)

SRIKRISHNAN S (ME21B2008)

MANASA H T(ME21B2012)

MOHAMMAD IMRAN ALAM (ME21B2016)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING KANCHEEPURAM

ABSTRACT

Stochastic gradient descent (SGD)-based optimizers are crucial in deep learning, yet the learning dynamics of complex models remain unclear. While SGD is fundamental for optimizing model parameters, little attention has been given to understanding the learning dynamics of optimizer parameters. This study proposes a perspective using control theory to grasp the model dynamics. By employing the status transfer function, we approximate parameter dynamics for various optimizers, treating them as first- or second-order control systems. This approach sheds light on how these parameters impact the stability and convergence time of deep learning models. Numerical experiments validate our findings

TABLE OF CONTENTS

ABSTRACT **i**

LIST OF FIGURES **1**

1 Problem Definition and Motivation **2**

1.1 Problem Definition 2

2 Overview of mathematical methods **3**

2.1 Introduction 3

2.2 Gradient Descent 3

3 Gradient Descent in Deep Neural Networks **4**

3.1 Overview of Gradient Descent in DNNs 4

3.2 Challenges and Optimization Methods 4

3.3 Advanced Optimization Techniques 4

3.4 Transfer Function Challenges 5

3.5 Mathematical Approaches 5

3.6 Control Model and Theoretical Contributions 5

4 Stochastic Gradient Descent with Momentum **6**

4.1 Overview 6

4.2 Algorithmic Overview 6

4.3 Benefits of Momentum 7

4.4 Iterative Scheme for SGD with Momentum 7

5 Nesterov Accelerated Gradient (NAG) in DNN **8**

5.1 Nesterov Accelerated Gradient (NAG) 8

5.2 Benefits of Nesterov Accelerated Gradient 8

5.3 Iterative Scheme for Nesterov Accelerated Gradient 8

5.4 Algorithm Overview 9

6 Conclusion and Future scope **10**

6.1 Conclusion 10

CHAPTER 1

Problem Definition and Motivation

1.1 Problem Definition

This study delves into the learning dynamics of gradient descent optimization in deep neural networks, specifically employing Stochastic Gradient Descent (SGD) and Nesterov Accelerated Gradient (NAG). The primary objectives include investigating the convergence behavior of SGD and NAG during training, assessing their impact on the model's generalization performance to unseen data, analyzing their sensitivity to hyperparameter choices, and evaluating their robustness in the presence of noisy or adversarial data. By comprehensively exploring these aspects, the research aims to contribute valuable insights into enhancing the efficiency and effectiveness of training deep neural networks.

CHAPTER 2

Overview of mathematical methods

2.1 Introduction

Deep neural networks (DNNs) excel in solving complex recognition problems across various data types. However, when applied to engineering issues, the predefined structure and limited training of DNN models may result in dynamic instability to new inputs, posing a serious risk of unexpected errors and catastrophic consequences. Stochastic gradient descent (SGD) methods aim to optimize parameters quickly, enhancing error backpropagation for accurate output. Despite the focus on obtaining optimal parameter distributions, the dynamics of parameter convergence are often overlooked. To address potential threats to DNNs, this study adopts a control perspective, using transfer functions to analyze and ensure the effective operation of multi-parameter-based optimizers.

Keywords: learning dynamics, deep neural networks, gradient descent, control model, transfer function

2.2 Gradient Descent

In Deep Neural Network (DNN) training, Gradient Descent minimizes a loss function by iteratively adjusting model parameters using computed gradients. Backpropagation efficiently calculates these gradients, enabling the model to learn. The learning rate, crucial for determining parameter update size, guides the optimization process. Stochastic Gradient Descent (SGD) updates parameters with mini-batches, introducing stochasticity. Nesterov Accelerated Gradient (NAG) enhances SGD with momentum for faster convergence. Understanding and tuning these aspects are vital for efficient and accurate DNN training.

CHAPTER 3

Gradient Descent in Deep Neural Networks

3.1 Overview of Gradient Descent in DNNs

Gradient descent optimization is commonly used for learning parameters in deep neural networks (DNNs) for tasks like image, voice, text, and video recognition. DNN structures vary, with examples including Multilayer Perception (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM), each tailored for specific applications.

3.2 Challenges and Optimization Methods

Training deep networks using gradient descent as a black-box optimizer can be challenging due to the need for intricate parameter settings and the time-consuming fine-tuning process. Batch gradient descent, while aiming for a global optimum, is computationally expensive, and efforts to balance system control and computational burden have been explored.

3.3 Advanced Optimization Techniques

Mini-batch gradient descent improves training speed by using subsets of training instances, showing effective results in a few epochs. Momentum factor, Adagrad, Adadelta, RMSprop, Adam, and AMSGrad are advanced optimization techniques that address issues like learning rate adaptability, stability, and convergence speed.

3.4 Transfer Function Challenges

Expressing the transfer function of a DNN is challenging due to its deep layered structure and matrix operations, making it difficult to infer the system's status space from input data analytically.

3.5 Mathematical Approaches

Mathematical descriptions, such as integral forms, Lyapunov stability theory, and transformations to high-dimensional spaces, bridge the gap between theoretical understanding and practical application of deep learning. Taylor expansion in singularities, non-convex optimization, and connections to nonlinear partial differential equations contribute to robust SGD.

3.6 Control Model and Theoretical Contributions

Theoretical contributions involve proposing first- and second-order control models for SGD, SGD momentum, and NAG optimizers, using transfer functions to provide a systemic understanding of deep learning optimization. Analysis of how learning rate, momentum factor, and gain coefficient impact optimizer dynamics, using time response root locus, provides insights into system characteristics. Comparative studies on popular datasets validate the impact of parameter combinations on system performance, contributing to a deeper understanding of DNN dynamics.

CHAPTER 4

Stochastic Gradient Descent with Momentum

4.1 Overview

Stochastic Gradient Descent (SGD) with momentum is an enhanced optimization algorithm designed to expedite convergence and mitigate oscillations in the parameter space during the training of deep neural networks. Unlike standard SGD, which relies solely on the current gradient for updating parameters, SGD with momentum introduces a moving average of past gradients to the update process.

[linesnumbered,ruled,vlined]algorithm2e

4.2 Algorithmic Overview

Initialize model parameters θ Set the momentum hyperparameter β (commonly between 0 and 1) Set the learning rate α

Update Rule:

1. Compute the gradient of the loss function with respect to the parameters: $\nabla_\theta J(\theta)$
2. Update the moving average of past gradients:
$$v = \beta v + (1 - \beta)\nabla_\theta J(\theta)$$
3. Update the parameters:
$$\theta = \theta - \alpha v$$

Iteration:

- Repeat steps 2 for a specified number of iterations or until convergence.

4.3 Benefits of Momentum

Accelerated Convergence: The inclusion of momentum allows the algorithm to maintain a more consistent direction towards the minimum, leading to faster convergence. Damped Oscillations: Momentum helps dampen oscillations or erratic movements in the parameter space, promoting smoother updates.

4.4 Iterative Scheme for SGD with Momentum

To implement an iterative scheme for SGD with Momentum, consider the following steps: Initialize initial parameters θ Initialize the momentum term v to zero

Iterative Update: each iteration Compute the gradient: $\nabla_\theta J(\theta)$
Update the momentum term using the momentum equation: $v = \beta v + \alpha \nabla_\theta J(\theta)$
Update the parameters using the momentum-enhanced update rule: $\theta = \theta - v$
graphicx graphicx

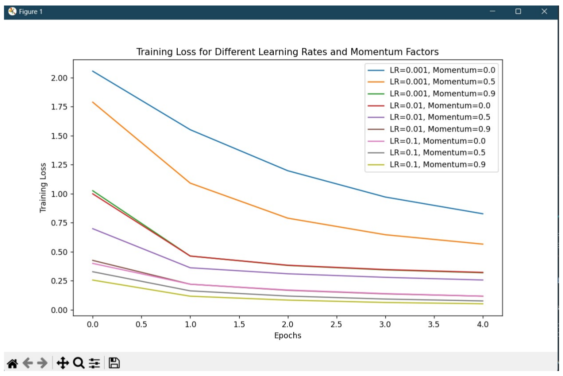
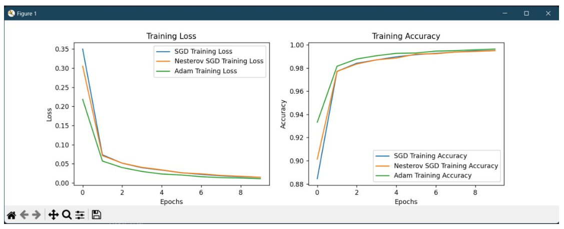


Figure 4.1: Results



CHAPTER 5

Nesterov Accelerated Gradient (NAG) in DNN

5.1 Nesterov Accelerated Gradient (NAG)

Nesterov Accelerated Gradient is an optimization algorithm that builds upon the foundation of Stochastic Gradient Descent (SGD) with momentum. Introduced by Yurii Nesterov, this method aims to improve the convergence behavior by adjusting the momentum term in a way that provides a more accurate estimate of the gradient.

5.2 Benefits of Nesterov Accelerated Gradient

Improved Accuracy: The lookahead gradient helps in providing a more accurate estimate of the gradient, contributing to improved convergence, especially in scenarios with high curvature. Faster Convergence: NAG often converges faster compared to traditional SGD with momentum.
Iterative Scheme for Nesterov Accelerated Gradient:

[linesnumbered,ruled,vlined]algorithm2e

5.3 Iterative Scheme for Nesterov Accelerated Gradient

To implement an iterative scheme for Nesterov Accelerated Gradient, follow these steps: Initialize initial parameters θ Initialize the momentum term v to zero

Iterative Update: each iteration Compute the lookahead parameters: $\theta' = \theta - \beta v$
Compute the gradient: $\nabla_{\theta'} J(\theta')$
Update the momentum term using the adjusted gradient: $v = \beta v + \alpha \nabla_{\theta'} J(\theta')$
Update the parameters using the Nesterov update rule: $\theta = \theta - v$

5.4 Algorithm Overview

[linesnumbered,ruled,vlined]algorithm2e

Initialize model parameters θ Set the momentum hyperparameter β (commonly between 0 and 1)

Update Rule:

1. Compute a "lookahead" gradient using the momentum term: $\theta' = \theta - \beta v$
2. Compute the gradient of the loss function with respect to the parameters at the lookahead position: $\nabla_{\theta'} J(\theta')$
3. Update the momentum term and adjust the parameters: $v = \beta v + \alpha \nabla_{\theta'} J(\theta')$ $\theta = \theta - v$
4. The final update of θ incorporates both the momentum term and the lookahead gradient.

Iteration:

- Repeat steps 2 for a specified number of iterations or until convergence.

graphicx

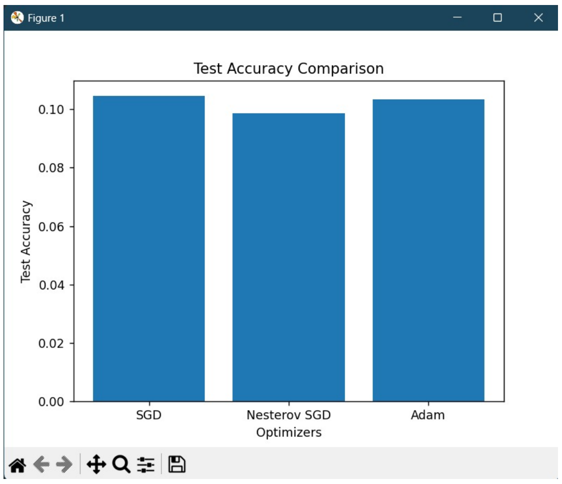


Figure 5.1: Test Accuracy Comparison

CHAPTER 6

Conclusion and Future scope

6.1 Conclusion

DNNs obtain promising results from the given input to the desired target, and benefit from the fine-tuned parameters. In this paper, the transfer function is applied to approximate three SGD-based optimizers. Basically, the SGD optimizer is a traditional first-order system, and it needs more time than a momentum optimizer or NAG optimizer to reach stability. The momentum optimizer changes SGD by adding one inertial element, transforming the system to a second-order system. Furthermore, the NAG optimizer puts one more zero on the second-order system of Momentum way. How to simulate and simplify those optimizers is our next move.

REFERENCES

WU, Wei, et al. “Learning Dynamics of Gradient Descent Optimization in Deep Neural Networks.” SciEngine, May 2021, https://www.sciengine.com/doi/10.1007/s11432-020-3163-0.