VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYEN LAM DUY – 521H0499**

**FINAL REPORT**

**INTRODUCTION TO**

**MACHINE LEARNING**

**HO CHI MINH CITY, YEAR 2023**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYEN LAM DUY – 521H0499**

**FINAL REPORT**

**INTRODUCTION TO**

**MACHINE LEARNING**

Advised by

Assoc. Prof.Le Anh Cuong

**HO CHI MINH CITY, YEAR 2023**

**ACKNOWLEDGMENT**

We would like to express our deepest gratitude to Ths. Mai Van Manh for his invaluable guidance and support throughout the preparation of this report. Your expertise and insights have been instrumental in shaping our understanding and approach to machine learning. Thank you for your time, patience, and dedication.

*Ho Chi Minh City, day 19th, month 12, year 2023*

*Author*

*(Signature and full name)*



Nguyen Lam Duy

**DECLARATION OF AUTHORSHIP**

I hereby declare that this thesis was carried out by ourselves under the guidance and supervision of Assoc. Prof. Le Anh Cuong; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

**I will take full responsibility for any fraud detected in our thesis**. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

*Ho Chi Minh City, day 19th, month 12, year 2023*

*Author*

*(Signature and full name)*



Nguyen Lam Duy

# ABSTRACT

This report provides a comprehensive exploration of optimizers in machine learning and delves into their applications and comparisons. Chapter 1 lays the foundation by defining optimizers and introducing common ones, including Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adagrad, and Adam. A thorough comparison of these optimizers is presented, highlighting their advantages and disadvantages.

Chapter 2 shifts the focus to Continual Learning and Testing in Production. It defines Continual Learning, discusses the associated challenges, such as problem identification, data gathering, and model maintenance. The report also introduces Test Production, outlining the testing process for machine learning models.

Contents

[ABSTRACT v](#_Toc154146661)

[CHAPTER 1. Understanding Optimizers in Machine Learning 1](#_Toc154146662)

[1.1 What Are Optimizers? 1](#_Toc154146663)

[1.2 Common optimizers 1](#_Toc154146664)

[1.2.1 Grandient Descent (GD) 1](#_Toc154146665)

[1.2.2 Stochastic Gradient Descent (SGD) 2](#_Toc154146666)

[1.2.3 Momentum 3](#_Toc154146667)

[1.2.4 Adagrad 4](#_Toc154146668)

[1.2.5 Adam 5](#_Toc154146669)

[1.3 Opimizer comparison 6](#_Toc154146670)

[CHAPTER 2. Continual Learning and Testing in Production 7](#_Toc154146671)

[2.1 What is Continual Learning? 7](#_Toc154146672)

[2.2 Continual learning challenges 7](#_Toc154146673)

[2.2.1 Unable to identify problem 7](#_Toc154146674)

[2.2.2 Gather necessary data 8](#_Toc154146675)

[2.2.3 Maintaining machine learning models 8](#_Toc154146676)

[2.3 Test production 8](#_Toc154146677)

[2.3.1 Test production process 8](#_Toc154146678)

[PREFERENCES 10](#_Toc154146679)

# Understanding Optimizers in Machine Learning

## What Are Optimizers?

An optimizer is an algorithm or a set of algorithms used to adjust the parameters of a model (w-weight and b-bias) in order to minimize or maximize a certain objective function. The objective function, often referred to as a loss function or cost function, represents the difference between the predicted output of the model and the actual target values in the training data. The goal of the optimizer is to find the optimal set of parameters that minimize this objective function.

In training phase of machine learning, the machine learning model makes predictions on the input data, and the optimizer evaluates how far these predictions deviate from the actual target values using a predefined objective function. The optimizer then calculates the gradient of this loss function with respect to the model’s parameter and update them. Repeat this step until the parameter converge (insignificant change in values of both bias and weight)

The choice of optimizer can significantly influence the training speed, stability, and final effectiveness of the machine learning model. Researchers and practitioners often experiment with different optimizers to find the one that suits their specific model architecture and dataset characteristics.

## Common optimizers

### Grandient Descent (GD)

Gradient descent is an iterative optimization algorithm employed to minimize a function by adjusting its parameters. In the context of machine learning, this function is often associated with the error or loss of a model. The primary objective is to find the minimum of the cost function, representing the optimal set of parameters for the given problem.

θ represents the model parameters, α is the learning rate, is the gradient of the cost function

* Pros:
  + **Efficiency:** Gradient descent is computationally efficient, particularly for large datasets and high-dimensional parameter spaces.
  + **Ease of Implementation:** The basic concept is simple and easy to implement, providing a foundational understanding of optimization.
* **Cons:**
  + **Sensitivity to Learning Rate:** The choice of the learning rate is crucial and can affect convergence speed and stability.
  + **Local Minima and Saddle Points:** Gradient descent may struggle in regions with local minima or saddle points, leading to slow convergence or suboptimal solutions.

### Stochastic Gradient Descent (SGD)

Stochastic is a variant of Gradient Descent. Instead of updating the weights once after each epoch, in each epoch, with N data points, we update the weights N times. In another word, SGD processes individual training examples, making it well-suited for scenarios with vast datasets and computationally intensive tasks.

The core principle of SGD involves updating model parameters using the gradient of the loss function with respect to a single training example. SGD formular is similar to GD but is performed on each data point

* Pros:
  + **Computational Efficiency:** SGD processes one training example at a time, reducing computational requirements compared to batch gradient descent, making it suitable for large datasets.
  + **Faster Convergence in Noisy Data:** The stochastic nature of SGD allows it to escape noisy or fluctuating cost surfaces more easily, potentially leading to faster convergence.
  + **Parallelization Potential:** Unlike batch gradient descent, SGD can be parallelized, making it well-suited for distributed computing and parallel processing environments.
* **Cons:**
  + **High Variability:** The stochastic nature of SGD introduces variability in the training process, leading to fluctuations in the cost function and less predictable convergence.
  + **Noise Sensitivity:** The randomness can cause noisy updates, making the convergence path less smooth and potentially leading to suboptimal solutions.

### Momentum

Momentum is a modification to the classical gradient descent algorithm designed to expedite convergence by incorporating a dynamic element into weight updates. This dynamism helps overcome challenges such as slow convergence in certain scenarios and oscillations around the minimum of the cost function.

This version of gradient descent helps to overcome local minimum problem of the original optimizer

The Momentum update is characterized by the introduction of a momentum term (*β*), usually between 0 and 1, which represents the contribution of the past update to the current update. The update rule is given by:

*vt*​ is the momentum term at time *t*, ∇*J*(*θt*​) is the gradient of the cost function at time *t*, *α* is the learning rate, and *θt*​ represents the model parameters at time *t*.

* Pros:
  + **Escape from Saddle Points:** Momentum assists in escaping saddle points, preventing the algorithm from getting stuck in these regions and promoting progress toward the minimum.
  + **Accelerated Convergence:** Momentum speeds up the convergence of the optimization process, especially in scenarios with complex, elongated, or narrow valleys in the cost function.
* **Cons:**
  + **Hyperparameter Tuning:** Selecting an appropriate value for the momentum hyperparameter (*β*) is crucial. Poor choices can lead to overshooting or slow convergence, requiring careful tuning.
  + **Dependence on Learning Rate:** The performance of Momentum is influenced by the choice of the learning rate (*α*). Inappropriate learning rates can impact convergence behavior.

### Adagrad

Adagrad, short for Adaptive Gradient Algorithm, represents a breakthrough in optimization algorithms by introducing adaptivity to the learning rate. It aims to address challenges associated with manual tuning of learning rates, offering a dynamic approach to optimize the convergence process. That is, Adagrad will let the learning rate change after each t time.

*α* is the learning rate, ∇(∇*J*(*θt*​) is the gradient of the cost function with respect to *θt*, *Gt*​ is the sum of squared gradients for *θt*​, and *ϵ* is a small constant to prevent division by zero

* Pros:
  + **Reduction in Manual Hyperparameter Tuning:** Adagrad reduces the need for manual tuning of learning rates, simplifying the optimization process. The algorithm automatically adjusts the learning rates based on the gradients, mitigating the need for extensive hyperparameter tuning.
  + **Adaptive Learning Rates:** Adagrad adapts the learning rates for each parameter based on the historical gradient information. This adaptivity allows the algorithm to handle different parameters with varying characteristics, leading to more efficient convergence.
* **Cons:**
  + **Learning Rate Decay:** Adagrad may suffer from diminishing learning rates over time. As the historical gradient information accumulates, the learning rates for some parameters can become very small, slowing down the convergence process. This issue has led to the development of alternative algorithms like Adadelta and RMSprop.
  + **Memory Requirements:** Adagrad requires storing and updating historical gradients for each parameter. This can lead to increased memory requirements, especially for long training sessions or in scenarios with a large number of parameters.

### Adam

Adam is a combination of Momentum and RMSprop. If explained in terms of physical phenomena, Momentum is like a ball rushing downhill, and Adam is like a very heavy ball with friction, so it easily overcomes the local minimum to the global minimum and when it reaches the global minimum it does not. It takes a long time to oscillate back and forth around the target because it has friction so it's easier to stop.

Adam computes adaptive learning rates for each parameter and incorporates momentum to prevent oscillations. The update rules for the parameters *θi*​ are given by:

*α* is the learning rate, ∇*J*(*θt*​) is the gradient of the cost function, *β*1​ and *β*2​ are decay rates, and *ϵ* is a small constant to prevent division by zero.

* Pros:
  + **Adaptive Learning Rates:** Adam adapts learning rates individually for each parameter, allowing for efficient convergence in high-dimensional and non-convex optimization spaces.
  + **Effective Handling of Sparse Gradients:** Adam effectively handles sparse gradients, making it suitable for scenarios with irregular or missing data.
* **Cons:**
  + **Sensitivity to Learning Rate:** Adam can be sensitive to the choice of the learning rate. Careful tuning is required to achieve optimal performance.
  + **Memory Requirements:** Similar to other adaptive algorithms, Adam involves storing and updating additional parameters, leading to increased memory requirements

## Opimizer comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aspect | Gradient Descent | Stochastic Gradient Descent | Momentum | Adagrad | Adam |
| Adaptivity | No | Yes | Yes | Yes | Yes |
| Handling sparse data | No | Yes | Yes | Very effective | Very effective |
| Learning rate adaptation | Manual tuning | Adaptive | Adaptive | Adaptive | Adaptive |
| Memory requirement | Low | Low to moderate | Moderate | Moderate to high | Moderate to high |
| Sensitive to learning rate | High | Moderate to high | Moderate | Moderate | Moderate |

# Continual Learning and Testing in Production

## What is Continual Learning?

Continual Learning, alternatively recognized as Incremental Learning or Life-long Learning, encapsulates the paradigm of sequentially acquiring proficiency in a multitude of tasks without succumbing to the phenomenon of knowledge decay concerning previously encountered tasks. This paradigm is characterized by the absence of access to data from earlier tasks during the training of subsequently introduced tasks.

## Continual learning challenges

### Unable to identify problem

Identifying the problem is a critical challenge in software engineering and holds particular significance in applied Machine Learning projects. This is underscored by the fact that only 20% of large corporations' AI pilots successfully transition to production, often failing to meet intended goals. The issue lies in either misaddressing the problem or overlooking variables during development.

### Gather necessary data

Acquiring and organizing data for Continuous Machine Learning is inherently challenging, as the collection of high-quality and precise real-world data is difficult. Unlike academia, where data is often accessible, public datasets are not well-curated for applied Machine Learning. Obtaining relevant data involves collecting or purchasing, both presenting their own set of difficulties.

### Maintaining machine learning models

Maintaining Machine Learning models is essential due to the dynamic nature of data environments and the occurrence of "concept drifts." Human intervention is crucial for constant maintenance, management, and course correction. The COVID-19 pandemic exemplifies how unforeseen events can impact machine models, necessitating retraining to adapt to changes in consumer behavior.

## Test production

"Test Production" in machine learning refers to the deployment and testing process of a machine learning model in a real-world environment, especially when the model is used to address a specific problem. It involves ensuring that the model performs correctly and effectively against new or out-of-distribution data when deployed in a specific application or system.

### Test production process

* Model Deployment: The machine learning model is deployed in a real-world environment, running on actual data and interacting with a specific system or application.
* General Testing: The model undergoes testing with scenarios and data representing a range of cases that may occur in real-world applications, ensuring it operates correctly and responds well to variations in the environment.
* New Data Testing: The model is tested with new data it may not have encountered before, ensuring independence and confidence in handling unfamiliar cases.
* Performance Evaluation: The model's performance is assessed based on project-specific metrics, including accuracy, reliability, response time, and other relevant factors.

**PREFERENCES**

Chapter 1:

[Viblo - optimizers](https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8#_1-gradient-descent-gd-2)

Chapter 2:

<https://levity.ai/blog/what-is-continuous-machine-learning>