TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN LÊ GIA BẢO**

**OPTIMIZER IN MACHINE LEARNING**

**FINAL REPORT**

**MACHINE**

**LEARNING**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN LÊ GIA BẢO – 520H0516**

**OPTIMIZER IN MACHINE LEARNING**

**FINAL REPORT**

**MACHINE**

**LEARNING**

Người hướng dẫn

**PGS.TS. LÊ ANH CƯỜNG**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Before I finish my final report, I would want to sincerely thank PGSTS. Lê Anh Cường for his guidance and support during the writing process. The information gained throughout the course of the study serves as both a basis for the research for the midterm report and a means of preparing for what is ahead. I would want to express our gratitude to all of the instructors and staff at Ton Duc Thang University's Department of Information Technology once more for their hard work in making it possible for me to finish my midterm report.

There are occasionally a lot of mistakes in our report. We're hoping that professors will understand and offer advice on how we might get better at producing reports.

*TP. Hồ Chí Minh, ngày 23 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Bao*

*Trần Lê Gia Bảo*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

We hereby declare that we are the principal investigator on this study, working under the scientific direction of PGSTS. Lê Anh Cường. The truthful study findings and information on this subject have never previously been released in any format. The author gathered the information in the tables for analysis, comments, and assessment from several sources, and it is made explicit in the reference section.

The Project furthermore incorporates a variety of evaluations, remarks, and statistics from other writers and organizations, all of which are annotated and referenced back to the original source.

**If any fraud is detected, We will take full responsibility for the content of my Project**. Ton Duc Thang University is not involved in copyright violations caused by me during the implementation process (if any).

*TP. Hồ Chí Minh, ngày 23 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Bao*

*Trần Lê Gia Bảo*

**MỤC LỤC**

[DANH MỤC HÌNH VẼ iv](#_Toc154254157)

[DANH MỤC CÁC CHỮ VIẾT TẮT v](#_Toc154254158)

[CHƯƠNG 1. QUESTION 1 - a 1](#_Toc154254159)

[1.1 REQUIREMENT 1](#_Toc154254160)

[1.2 What is optimizer ? 1](#_Toc154254161)

[1.3 Gradient Descent 1](#_Toc154254162)

[1.4 Stochastic Gradient Descent (SGD) 2](#_Toc154254163)

[1.5 Stochastic Gradient Descent with Momentum 3](#_Toc154254164)

[1.6 AdaGrad 4](#_Toc154254165)

[1.7 Root Mean Square Prop (RMSProp) 5](#_Toc154254166)

[1.8 Adam (Adaptive Moment Estimation) 6](#_Toc154254167)

[1.9 Conclusion 7](#_Toc154254168)

[CHƯƠNG 2. QUESTION 1-b 8](#_Toc154254169)

[2.1 What is Continuous Learning? 8](#_Toc154254170)

[2.2 What is Test Production ? 8](#_Toc154254171)

[2.3 Integrating Continuous Learning with Test Production 9](#_Toc154254172)

[TÀI LIỆU THAM KHẢO 11](#_Toc154254173)

# DANH MỤC HÌNH VẼ

[Hình 1.1 Compare Stochastic Gradient Descent and Gradient Descent 3](#_Toc154254106)

[Hình 1.2 SGD without Momentum 3](#_Toc154254107)

[Hình 1.3 SGD with Momentum 4](#_Toc154254108)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| BERT | Stochastic Gradient Descent |
| AdaGrad | Adaptive Gradient Algorithm |
| Adam | Adaptive Moment Estimation |
| RMSProp | Root Mean Square Prop |
|  |  |

# QUESTION 1 - a

## REQUIREMENT

Yêu cầu: Trình bày một bài nghiên cứu, đánh giá về các vấn đề:

1. Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy
2. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

## What is optimizer ?

To get the most accurate and optimal predictions, it is crucial to adjust the model's weights throughout the training phase. But just how do you go about doing it? When, how, and how much do you adjust the parameters in your model?

Optimizers are the best response to all of the aforementioned questions. By updating the model in response to the loss function's output, they link the loss function and model parameters together. To put it another way, optimizers tinker with the weights to shape and mold your model into the most accurate form possible. The optimizer is guided across the terrain by the loss function, which indicates when it is traveling in the correct or incorrect path.

Some factors commonly used to evaluate an optimizer algorithm:

* Fast (during training)
* High generalization (still recognizes untrained samples)
* High precision

## Gradient Descent

Gradient Descent is a fundamental optimization algorithm used in machine learning and deep learning to minimize a function. It is particularly used to update the parameters (like weights) of a model to minimize the cost (or loss) function.

1. Start with initial values for the parameters that you want to optimize. In the context of neural networks, these are typically the weights and biases.
2. Calculate the gradient of the cost function with respect to each parameter. The gradient is a vector that points in the direction of the steepest increase of the function. In optimization, we are interested in the opposite direction (to decrease the cost).
3. Adjust the parameters in the opposite direction of the gradient. The size of the step you take in the direction opposite to the gradient is determined by the learning rate, a hyperparameter.

new parameter=old parameter−learning rate×gradient

old parameter : weight and bias ( example in neural network )

gradient is calculated with respect to the parameter, indicating how much a change in the parameter will change the function's output.

Learning rate: Choosing the right learning rate is crucial. A learning rate that's too large can cause the algorithm to overshoot the minimum, while a too-small learning rate can make the convergence very slow.

Advantage:

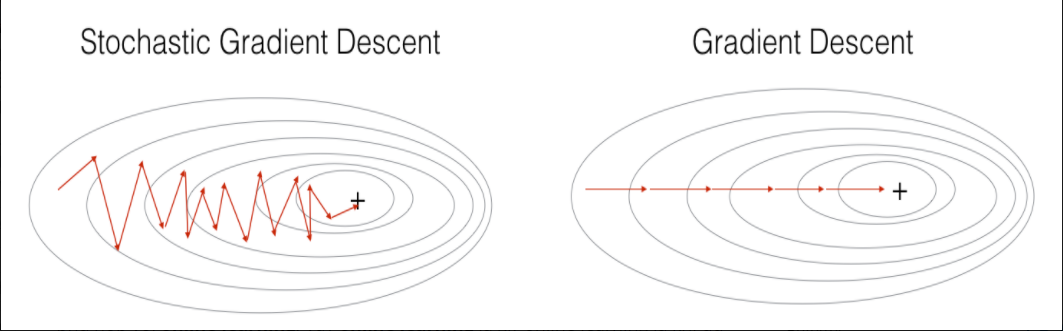
Basic, easy to understand gradient descent algorithm. The algorithm solved the problem of optimizing the neural network model by updating the weights after each loop.

Disadvantage:

Because of its simplicity, the Gradient Descent algorithm has many limitations such as depending on the initial initial solution and learning rate.

## Stochastic Gradient Descent (SGD)

SGD stands for Stochastic Gradient Descent.In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy or less random manner, but the problem arises when our datasets get really huge.

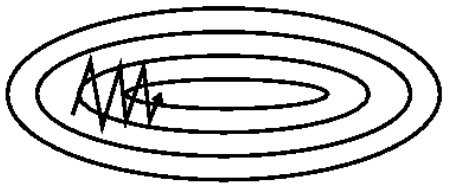


Hình 1.1 Compare Stochastic Gradient Descent and Gradient Descent

This problem is solved by Stochastic Gradient Descent. In SGD, it uses only a single sample to perform each iteration. The sample is randomly shuffled and selected for performing the iteration.

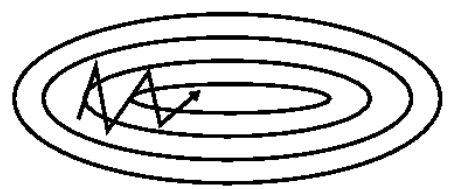
Since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn’t matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with significantly shorter training time.

## Stochastic Gradient Descent with Momentum



Hình 1.2 SGD without Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum as image 1.2.



Hình 1.3 SGD with Momentum

Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations as can be seen in Image 3. It does this by adding a fraction of the update vector of the past time step to the current update vector

* is the momentum term at time *t*, which is a combination of the current gradient and the previous momentum term.
* *γ* is the momentum coefficient, typically set to a value close to 1 (e.g., 0.9).
* is the learning rate.
* is the gradient of the loss function with respect to the parameter at the current position in the parameter space.

## AdaGrad

AdaGrad adapts the learning rates of all parameters by scaling them inversely proportional to the square root of the sum of all historical squared values of the gradient. This means that parameters with frequent large gradients will have their effective learning rate reduced, while parameters with small or infrequent gradients will have their effective learning rate increased.

AdaGrad adapts the learning rates of all parameters by scaling them inversely proportional to the square root of the sum of all historical squared values of the gradient. This means that parameters with frequent large gradients will have their effective learning rate reduced, while parameters with small or infrequent gradients will have their effective learning rate increased.

* - the gradient of a parameter, :math: 'Theta ' at an iteration t.
* - the learning rate
* - very small value to avoid dividing by zero

## Root Mean Square Prop (RMSProp)

RMSprop solves Adagrad's decreasing learning rate problem by dividing the learning rate by the average of the squares of the gradient.

* - the exponentially weighted average of past squares of gradients
* - cost gradient with respect to current layer weight tensor
* - weight tensor
* - hyperparameter to be tuned
* - the learning rate
* - very small value to avoid dividing by zero

## Adam (Adaptive Moment Estimation)

Combining ideas from Momentum and RMSprop, Adam computes adaptive learning rates for each parameter. It's one of the most popular optimizers due to its effectiveness in a wide range of machine learning problems.

First, it computes the exponentially weighted average of past gradients ()

Second, it computes the exponentially weighted average of the squares of past gradients ()

Third, these averages have a bias towards zero and to counteract this a bias correction is applied ,

Lastly, the parameters are updated using the information from the calculated averages.

* - the exponentially weighted average of past gradients
* - the exponentially weighted average of past squares of gradients
* - hyperparameter to be tuned
* - hyperparameter to be tuned
* - cost gradient with respect to current layer
* - the weight matrix (parameter to be updated)
* - the learning rate
* - very small value to avoid dividing by zero

## Conclusion

| **Algorithm** | **Learning Rate** | **Best For** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- |
| Gradient Descent | Fixed or manually changed | Simple and small datasets | Simple and easy to implement | Slow convergence on large datasets; sensitive to the choice of learning rate |
| Stochastic Gradient Descent (SGD) | Typically fixed | Large datasets, online learning | Fast update, good for large datasets | High variance in updates can lead to unstable convergence |
| SGD with Momentum | Typically fixed, with momentum as an additional hyperparameter | Problems with noisy gradients or non-stationary objectives | Reduces oscillations, faster convergence than basic SGD | Requires careful tuning of the momentum hyperparameter |
| AdaGrad | Adaptive | Sparse data and large-scale learning problems | Automatically adjusts learning rate, good for sparse data | Learning rate decreases monotonically, can become infinitesimally small |
| Root Mean Square Prop (RMSProp) | Adaptive | Non-convex optimization problems (e.g., neural networks) | Addresses AdaGrad's radically diminishing learning rates | Still requires tuning of learning rate |
| Adam | Adaptive | General-purpose, works well on most problems | Combines benefits of RMSProp and Momentum; generally performs well | More complex and computationally expensive than simple SGD |

# QUESTION 1-b

## What is Continuous Learning?

Continuous Learning, also known as Continuous Machine Learning (CML), is a process where a model learns from new data streams without requiring explicit retraining. This approach contrasts with traditional machine learning models that are trained on a static dataset and periodically retrained. Continuous Learning models iteratively update their parameters to accommodate new data distributions, thus staying relevant and adaptable over time. The process involves standard modeling principles like preprocessing, model selection, hyperparameter tuning, training, deployment, and monitoring, along with additional steps like data rehearsal and the implementation of a continuous learning strategy​​.

**Data Stream Processing**: Continuous learning systems often deal with data streams rather than fixed datasets. Efficiently processing and learning from these streams is crucial.

**Model Updating**: Regularly updating the model with new data. This could involve retraining the model from scratch or incrementally updating it.

**Handling Catastrophic Forgetting**: As new data is incorporated, it's essential to ensure the model does not forget previously learned information. Techniques like experience replay, regularization, and architectural strategies are used to mitigate this.

**Evaluation and Monitoring**: Continuously evaluating the model's performance to detect any degradation or improvement in its predictions. This often involves setting up metrics and monitoring systems.

## What is Test Production ?

Test Production in machine learning involves a series of steps to ensure that a model is reliable, accurate, and ready for deployment in a real-world environment. The process generally includes automated testing, manual validation, and continuous evaluation in a production setting.

Automated tests for model verification are crucial. They typically involve running a suite of tests every time a change is made to the model. These tests can include smoke tests, which verify that the entire pipeline functions correctly on a small set of actual data, and unit tests, which check specific aspects of the model and data. For example, a smoke test might run the entire end-to-end pipeline with a small dataset to ensure nothing breaks, while unit tests could check data quality or model predictions within specific bounds​

Testing in production is another key aspect, especially when dealing with Continual Learning systems. Since these systems are designed to adapt to data distribution shifts, it's necessary to test them with live data in production. This approach helps ensure that the model performs as expected in the real world and can adapt to changing data patterns​​.

Additionally, testing the model under high-load conditions is vital to understand its performance and scalability. This can reveal how the model responds under extreme workloads, helping identify any potential bottlenecks or performance issues​.

## Integrating Continuous Learning with Test Production

Integrating Continuous Learning with Test Production in machine learning involves a series of steps and strategies to ensure that models not only learn effectively from ongoing data streams but also perform reliably and accurately when deployed in production. The core aspects of this integration involve:

**AutoML and Hyperparameter Optimization**: This is important for adapting to changing data. AutoML helps in selecting the right algorithms and tuning their parameters, making the model adaptable to new data patterns.

**Monitoring and Control**: Continuous monitoring of predictions in the deployment area is crucial. This includes data cleaning, labeling, and ensuring that the learning system adapts to new data while maintaining overall control over the pipeline.

**Data Engineering and DevOps Automation**: Utilizing tools like Kubernetes for automating the machine learning infrastructure is recommended for efficient deployment and management of experiments.

# TÀI LIỆU THAM KHẢO

Tiếng Việt

https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

Tiếng Anh

https://machinelearningcoban.com/2017/01/16/gradientdescent2/

<https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html#adam>

<https://arxiv.org/pdf/1502.03167.pdf>

<https://ruder.io/optimizing-gradient-descent/>

https://www.datacamp.com/blog/what-is-continuous-learning

# 