Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL EXAM REPORT

INTRODUCTION TO MACHINE LEARNING

Instructor: **MR. LE ANH CUONG**

Student: **BUI HUU LOC – 521H0504**

Class : **21H50301**

Year **: 2023-2024**

HO CHI MINH CITY, 2023

Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL EXAM REPORT

INTRODUCTION TO MACHINE LEARNING

Instructor: **MR. LE ANH CUONG**

Student: **BUI HUU LOC – 521H0504**

Class : **21H50301**

Year : **2023-2024**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENT

First, I would like to send my most sincere thanks to Mr. Le Anh Cuong, who enthusiastically taught me to help me gain full knowledge of the subject and complete this report.

Second, I would like to send my thanks to the teachers of the Department of Information Technology at Ton Duc Thang University for giving me the opportunity to write this report.

I look forward to receiving feedback from teachers so I can improve my report.

Finally, I would like to wish the teachers good health and success in their careers.

Ho Chi Minh city, 23th December 2023

Author

(Sign and write full name)

Bui Huu Loc

THIS PROJECT WAS COMPLETED AT TON DUC THANG UNIVERSITY

I fully declare that this is my own project and is guided by Mr. Le Anh Cuong; The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

Besides that, the project also uses several comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

Should any frauds be found, I will take full responsibility for the content of my report. Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

Ho Chi Minh city, 23th December, 2023

Author

(Sign and write full name)

Bui Huu Loc

CONFIRMATION AND ASSESSMENT SECTION

Instructor confirmation section

Ho Chi Minh   December 2023

(Sign and write full name)

Evaluation section for grading instructor

Ho Chi Minh   December 2023

(Sign and write full name)

SUMMARY

This is the final report on Machine Learning of the Faculty of Information Technology of Ton Duc Thang University.

My writing will sometimes have many errors. I am very open to receiving constructive contributions from teachers and will take them as lessons to improve.

INDEX

CHAPTER 1 – EXPLORE AND CONTRAST OPTIMIZER TECHNIQUES FOR TRAINING MACHINE LEARNING MODELS.

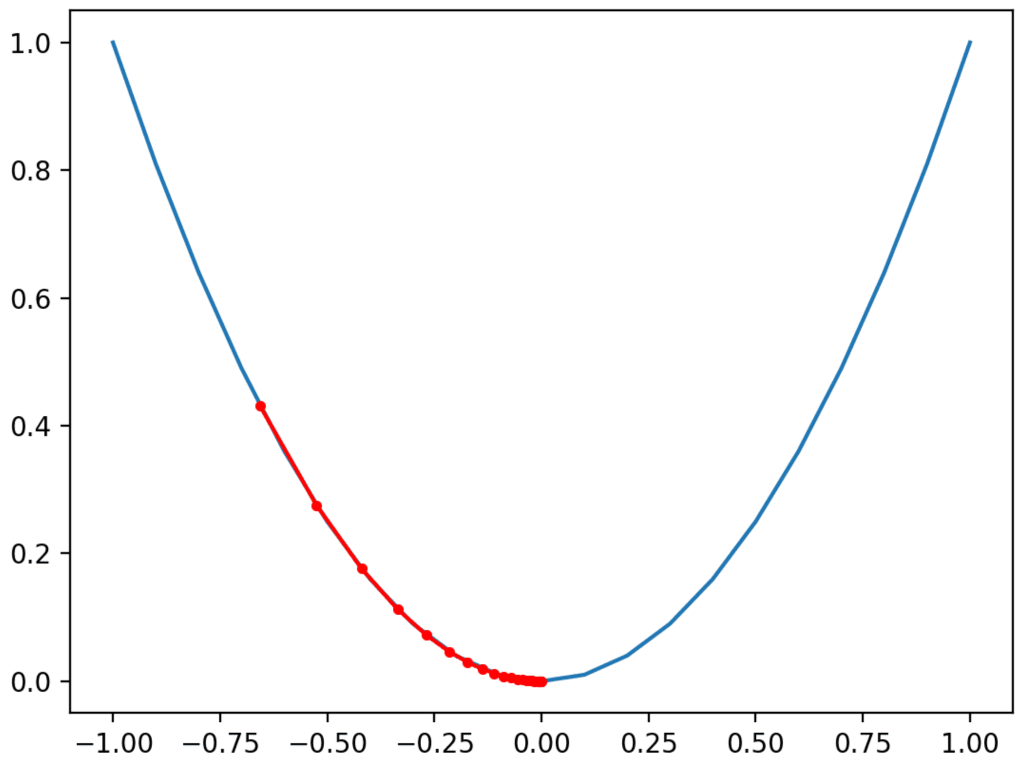
1. Acquire knowledge of optimizer techniques for training ML models.

In essence, the optimization algorithm serves as the foundational framework for constructing a neural network model. The primary objective is to enable the model to "learn" intricate features or patterns present in the input data. Through this learning process, the algorithm systematically identifies an optimal set of weights and biases, fine-tuning the model to achieve effective optimization

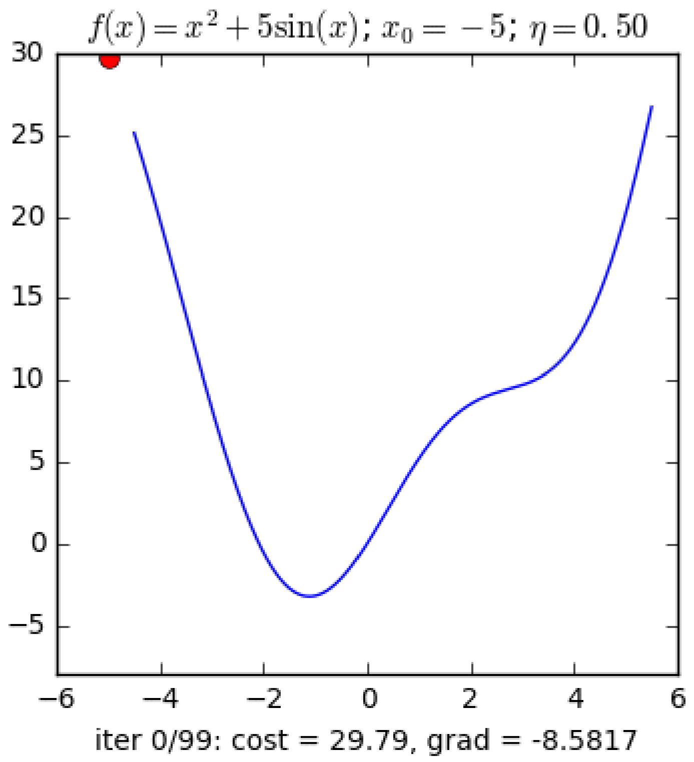
1.1 Gradient Descent (GD)

Gradient Descent (GD) is a fundamental optimization algorithm commonly used in machine learning and neural network training. Its primary objective is to minimize a cost function, which represents the difference between the predicted output of a model and the actual target values in the training dataset.

xnew = xold - learningrate.gradient(x)

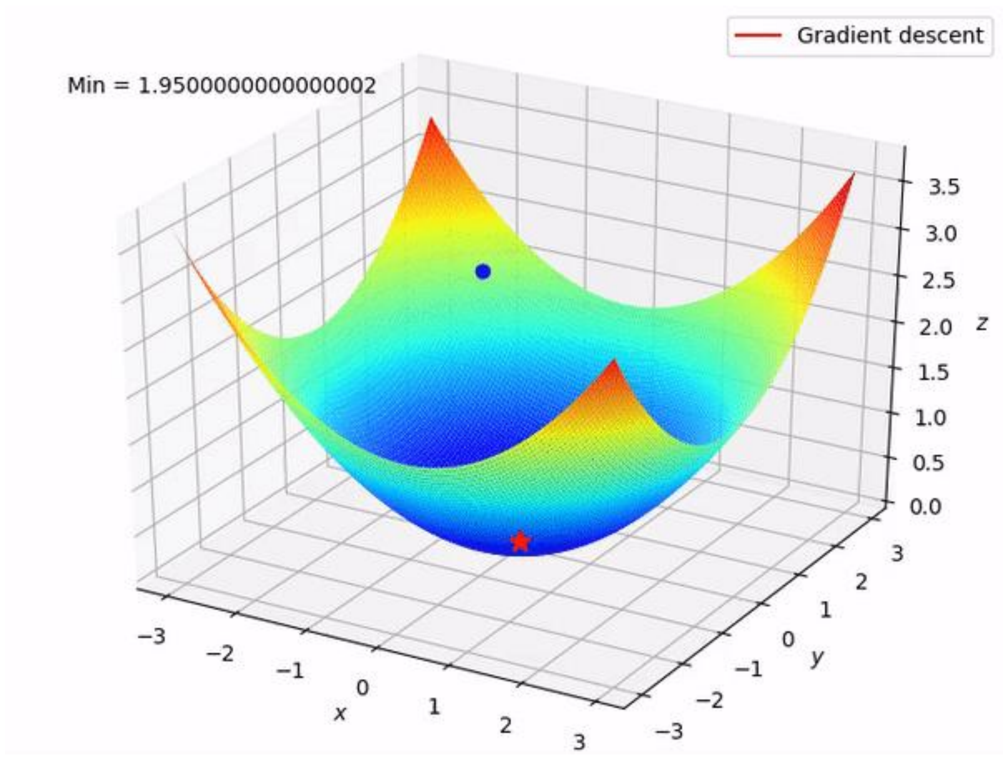


1.1.1 Gradient for 1-variable function:



Gradient descent depends on many factors: for example, choosing different initial x points will affect the convergence process; or the learning rate is too large or too small, it also has an impact: if the learning rate is too small, the convergence speed is very slow, affecting the training process, and if the learning rate is too large, it will quickly reach the target after a few minutes. However, the algorithm does not converge and loops around the destination because the jump is too large.

1.1.2 Gradient Descent for multi-variable functions:



1.1.3 Advantages and disadvantages of Gradient Descent

Advantages:

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

Disadvantages:

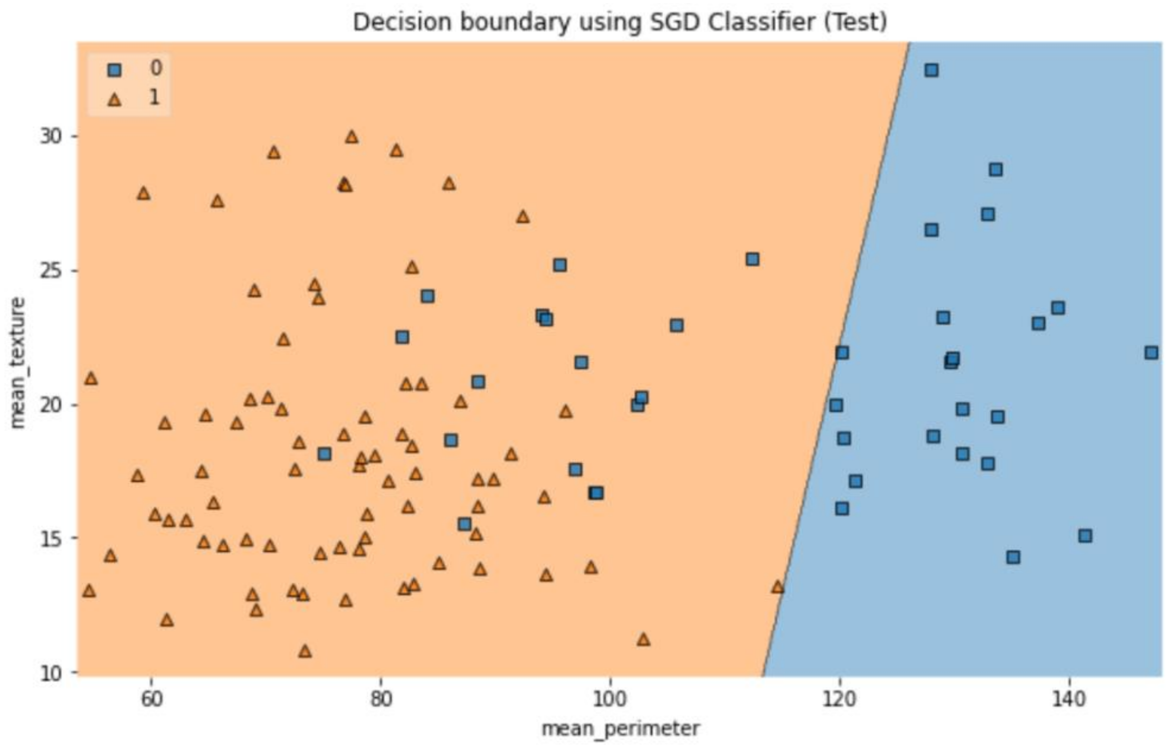
- The Gradient Descent algorithm has many limitations such as depending on the initial initial solution and learning rate.
- A learning rate that is too large will cause the algorithm to fail to converge and hover around the target because the jump is too large; or the small learning rate affects the training speed

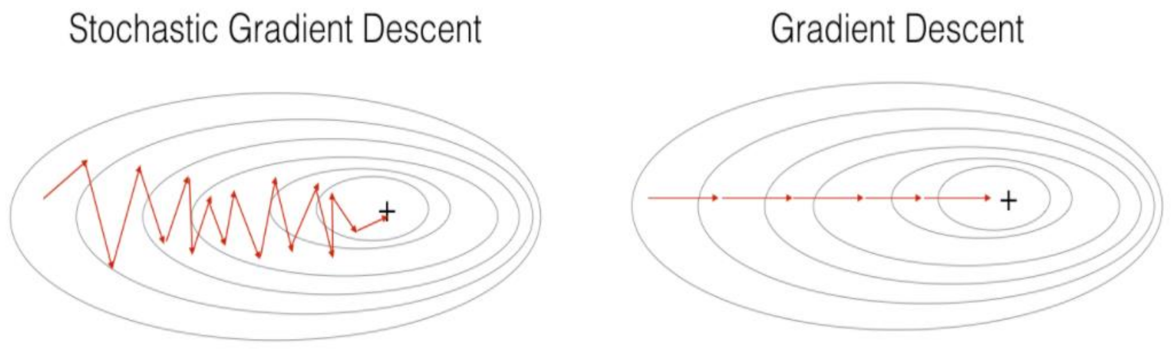
1.2 Stochastic Gradient Descent (SGD)

A specific version of the gradient descent algorithm is known as stochastic gradient descent (SGD). Unlike traditional gradient descent, where we update the model's weights once after processing the entire dataset in one epoch, SGD introduces a more frequent weight update strategy. In each epoch, the weights are updated N times, corresponding to the N data points in the training set.

While this approach may initially appear to slow down the progress in each epoch, it offers a notable advantage in terms of convergence speed. SGD tends to converge rapidly within just a few epochs compared to its traditional counterpart. Despite the more frequent updates, the core formula of SGD remains akin to that of traditional gradient descent, with the algorithm's adjustments being applied to each individual data point during the training proce.

.





Gradient Descent (GD) exhibits a smoother trajectory compared to Stochastic Gradient Descent (SGD), as evident in the visual comparison above. The reason behind this difference lies in the fact that representing the entirety of the dataset by a single data point is impractical. GD encounters challenges, particularly with extensive databases containing millions of data points, where calculating the derivative for all the data in each iteration becomes computationally burdensome.

The unwieldiness of GD becomes more pronounced, especially when dealing with large datasets, making it less suitable for scenarios involving continuous learning. The algorithm's non-online nature and the need for extensive computation time make it less optimal for applications requiring real-time adjustments, such as in educational settings.

To address these challenges, Stochastic Gradient Descent (SGD) emerges as a solution. SGD efficiently tackles the issue of updating parameters when new data is introduced by updating only one data point at a time. This characteristic makes SGD well-suited for online learning scenarios where continual updates are essential, allowing the algorithm to adapt quickly to evolving datasets

.

1.2.1 Advantages and disadvantages of Stochastic Gradient Descent

Advantages:

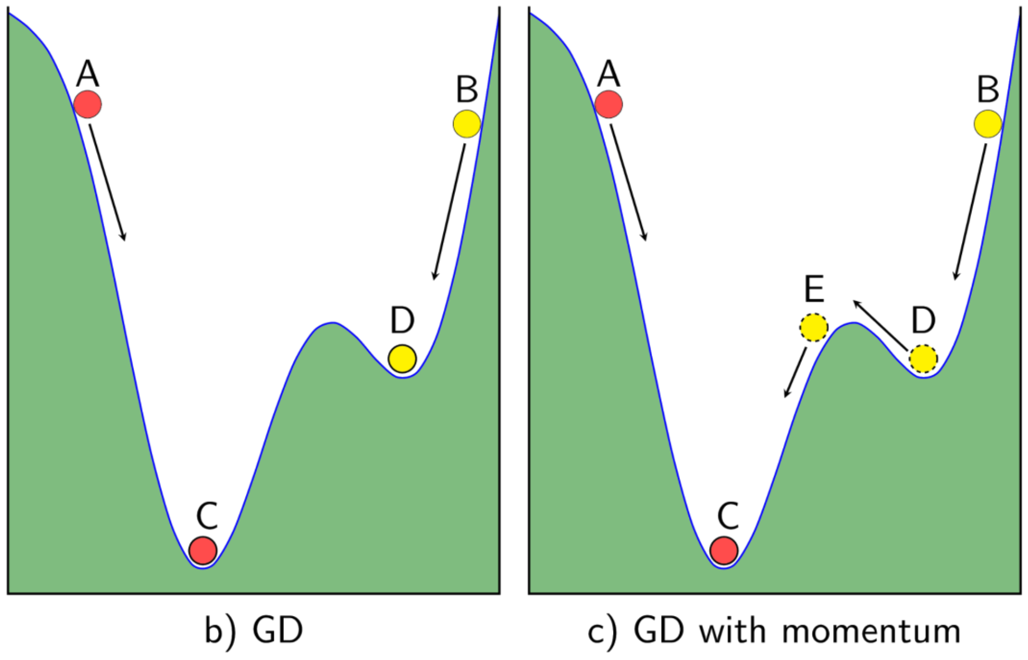
- Large databases are handled by the algorithm in a way that GD cannot. Even now, this optimization algorithm is frequently employed.

Disadvantages:

- The two primary drawbacks of gradient descent (learning rate and starting data points) have not yet been resolved by the algorithm.

1.3 Momentum

In order to address the constraints associated with the Gradient Descent algorithm, a modified version called gradient descent with momentum is often employed. This adaptation was devised to combat a particular challenge faced by the original GD algorithm: the tendency to get stuck in undesirable local minimum points. This solution, known as Momentum, was introduced to enhance the optimization process and prevent the algorithm from converging to suboptimal solutions.



xnew = xold -(gama.v + learningrate.gradient)

** In there :*

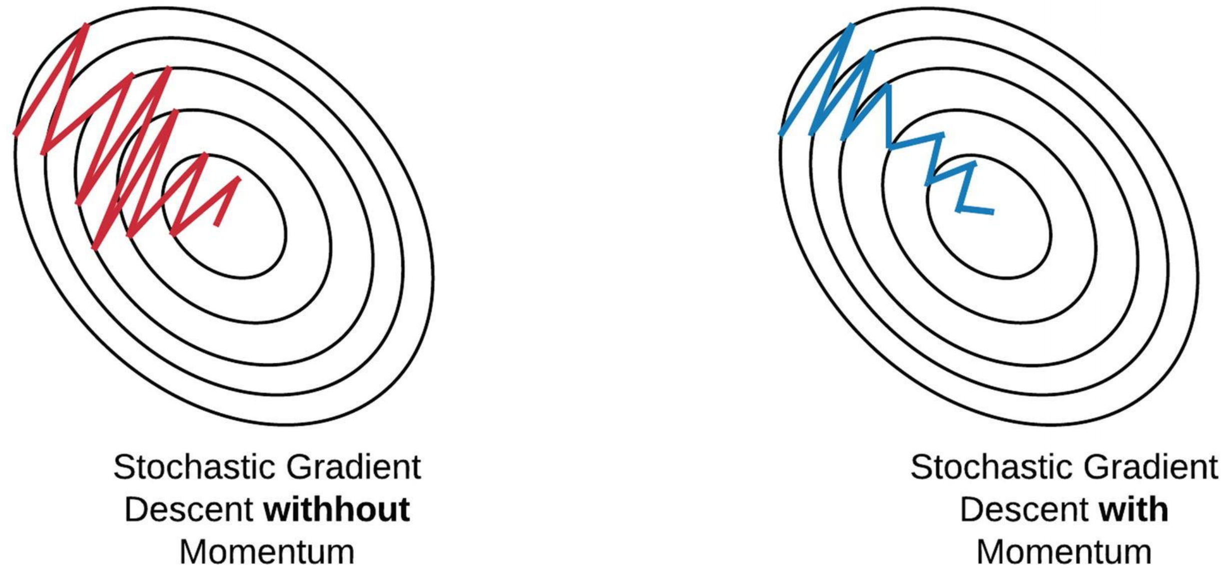
xnew: new coordinates

xod : old coordinates

gamma: parameter , usually =0.9

learningrate: learning speed

gradient : derivative of function f


Stochastic Gradient Descent **withhout** Momentum
Stochastic Gradient Descent **with** Momentum

1.3.1 Advantages and disadvantages of Momentum

Advantages:

- The optimal algorithm solves the problem: Gradient Descent does not reach the global minimum point but only stops at the local minimum.

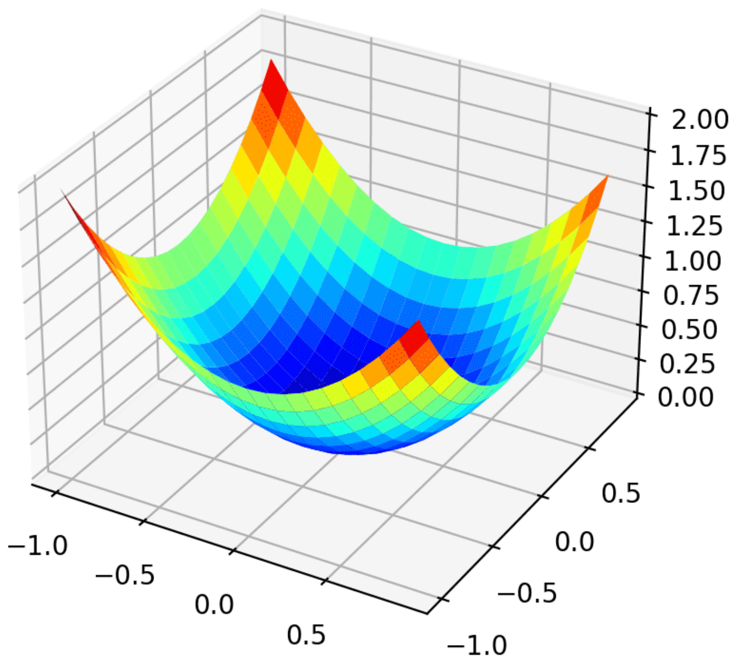
Disadvantages:

- Although the momentum helps the marble climb uphill towards the destination, when it gets close to the destination, it still takes a lot of time to oscillate back and forth before stopping completely.

1.4 AdaGrad

The Adaptive Gradient Algorithm, or AdaGrad for short, is an extension of the gradient descent optimization algorithm.

It is designed to speed up the optimization process, such as reducing the number of function evaluations needed to reach the optimum, or to improve the ability of the optimization algorithm, e.g. to yield the result same better.



$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{G_t + \epsilon}} * g_t$$

In there:

n: Constant.

g_t: Gradient at time t.

ϵ: Error avoidance factor (divided by sample equal to 0).

G: Diagonal matrix where each element on the diagonal (i,i) is the square of the parameter vector derivative at time t.

1.4.1 Advantages and disadvantages of AdaGrad

Advantages:

- AdaGrad has the advantage of avoiding adjusting the learning rate manually. Just set the default learning rate to 0.01 and the algorithm will automatically adjust.
- Efficient in processing multidimensional data.

Disadvantages:

- Because the square of the gradient accumulates, the learning rate decreases over time, which can slow down the convergence process.
- Parameters with large gradients are often updated quickly, which can lead to too large a learning rate.

1.5 RMSprop

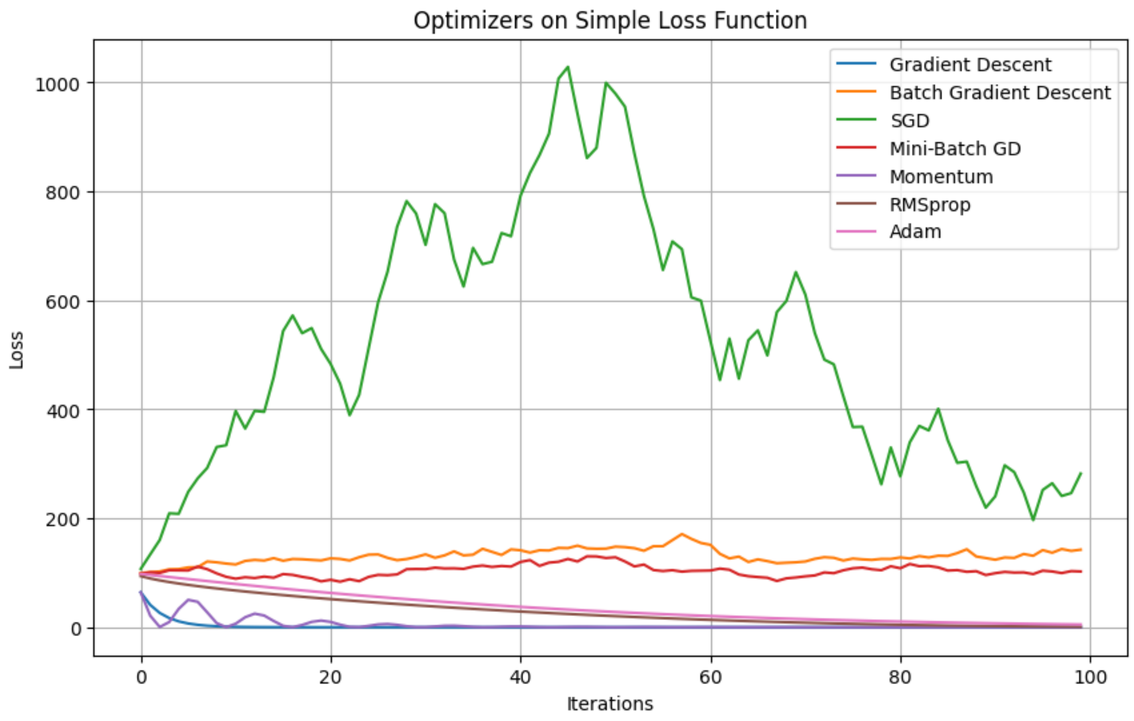
RMSprop solves Adagrad's decreasing learning rate problem by dividing the learning rate by the average of the squares of the gradient.

2. Compare Optimizer methods in training machine learning models.

2.1 Compare using table:

METHOD	EFFICIENCY	CONVERGENCE SPEED
Gradient Descent (GD)	Solve small and simple problems well.	Can get stuck at local minima without corrective techniques.
Stochastic Gradient Descent (SGD)	Effective with large and heterogeneous data sets.	Highly volatile and can be unstable due to randomness.
Momentum	Reduces oscillations and provides faster convergence on highly oscillatory loss function surfaces. Effective in passing through the minimum point.	Parameters need to be adjusted to achieve best performance.
AdaGrad	Efficient in handling high-dimensional data.	Excessive updates for frequently updated parameters.

2.2 Compare graphically on simple and complex problem types:



CHAPTER 2 – LEARN ABOUT CONTINUOUS LEARNING AND TEST PRODUCTION WHEN BUILDING A MACHINE LEARNING SOLUTION TO SOLVE A CERTAIN PROBLEM

2.1 Continuous learning

2.1.1 The concept of continuous learning

Continuous learning is an area of machine learning in which the learning model must continuously learn and update its knowledge over time, regularly and automatically based on new data. This is necessary to ensure that the model always reflects changes in the data and environment.

Continuous learning faces two main challenges: forgetting old knowledge and interfering with already learned knowledge (the phenomenon of "catastrophic forgetting"). To solve this problem, the Continuous Learning method focuses on developing mechanisms to retain old knowledge and control the learning of new knowledge.

2.1.2 Mechanism of Continuous Learning

- Replay-based methods

In this method, historical data is "repeated" to train the model on old and new data. Old data is stored and reused as an expanded training set.

Advantages:

- Helps the model retain important information from old data.
- Make sure that the model does not completely forget what it has learned.

Disadvantages:

- Requires large storage to retain old data.
- May not be effective if the old data set is too large.

- Regularization-based Methods:

Regularization is the addition of regularization terms to the loss function to stabilize and control the learning process. In the context of continuous learning, regularization can be designed to ensure the model does not forget too much information from old data.

Advantages:

- Increase the generality of the model.

● Dynamic Architecture Methods:

This method focuses on changing the model's architecture over time to adapt to changes in data or tasks. The architecture can change automatically or be controlled by a management structure.

Advantages:

- Allows the model to flexibly adapt to changes in data and tasks.
- Helps reduce model complexity.

● Application of Continuous Learning:

Continuous Learning can be applied in many situations in building machine learning solutions.

2.1.3 Why Continual Learning

The main purpose is to help your model adjust to shifts in data distribution. There are specific situations where quickly adapting to changing distributions is crucial. Here are some examples:

- Use cases with unpredictable and sudden changes: This applies to services like ride-sharing. For instance, a spontaneous concert in a random area on a Monday may cause the "Monday pricing ML model" to struggle, as it wasn't designed to handle such unexpected events.
- Use cases without training data for a specific event: Consider e-commerce models during Black Friday or other unprecedented sales events. Gathering

historical data for predicting user behavior on Black Friday is challenging, making it essential for the model to adapt dynamically throughout the day.

- Use cases susceptible to the cold start problem: This occurs when your model needs to make predictions for a new or logged-out user with no historical or outdated data. Adapting the model as soon as you receive some data from that user is crucial to providing relevant recommendations

2.1.3 Continual Learning Challenges

- Updating Models Frequently

To update your model every hour, you require quality labeled training data at the same frequency. The more often you update, the more crucial this challenge becomes.
Many companies retrieve training data from warehouses like Snowflake or BigQuery. However, data from various sources enters the warehouse at different speeds and through different mechanisms.
For instance, some data arrives in real-time through events, while other portions come in through daily or weekly ETL processes copying data from other sources.
One solution is to pull data directly from real-time transport for training before it reaches the warehouse, especially if the real-time transport is connected to a feature store. However, challenges arise:
Not all data may flow through events, particularly external vendor systems beyond your control. If you need fresh data from these systems, you must find a way to capture changes using events, employing methods like web-hooks or polling APIs.
Some companies heavily process and join data through batched ETLs in the warehouse to enhance usability. Shifting to a full real-time transport strategy requires figuring out how to perform the same processing on a data stream.

- Evaluation Challenge

Embracing continual learning has a downside – the risk of major model failures increases with more frequent updates. Also, continual learning makes models vulnerable to coordinated attacks aimed at poisoning them.
To mitigate these risks, thorough testing before releasing models to a larger audience is crucial. However, testing takes time and could be a limiting factor for achieving the fastest model update frequency. For instance, a new fraud detection model might need around 2 weeks of traffic for a confident evaluation

- Data scaling challenge

Scaling data for feature calculation is a challenge. It involves using global data statistics like min, max, average, and variance. When employing stateful training, these statistics must consider both previous data used for training and the new data refreshing the model, making tracking global statistics tricky.
A common approach is to calculate or approximate these statistics incrementally as new data is observed, rather than loading the full dataset at training time. An example is the "Optimal Quantile Approximation in Streams."

2.2 Testing models in Production

2.1.1 Pre-deployment offline evaluations

There are two common ways to test models: (1) using a static test split for benchmarking and (2) running backtests. Test splits provide a trusted benchmark, but good performance on an old split doesn't guarantee success with current data distribution in production.

Backtesting involves using the latest labeled data not seen during training to assess performance, like using the last hour's data if the model was trained on the last day. While backtesting is introduced here, it's not a sole solution. Production testing involves more than label-related performance; factors like latency, user behavior, and system integration correctness must be observed for a safe model rollout

2.2.2 Testing in Production Strategies

- Shadow Deployment

Intuition	Pros	Cons
Deploy the challenger model alongside the existing champion model. Route all requests to both models but only use the champion's predictions. Log predictions for both models for later comparison	Safest deployment method; even if the new model has bugs, predictions won't be served. Simple conceptually. Gathers enough data quickly for statistical significance since all models get full traffic.	Not suitable when observing user interactions is crucial for measuring model performance (e.g., shadow recommender models). Expensive as it doubles predictions and compute requirements. Challenges with online prediction modes, like

		dealing with delays or failures in the shadow model and deciding if the primary model should also fail in such cases.

- A/B Testing

Intuition	**Pros**	**Cons**
Deploy the challenger model alongside the champion model (model A) and direct a portion of traffic to the challenger (model B). Users see predictions from the challenger, and monitoring and analysis determine if the challenger outperforms the champion	Captures user reactions to different models since predictions are served. Simple and well-documented with available libraries. Cost-effective, involving only one prediction per request.	Less safe than shadow deployments; requires a stronger offline evaluation to avoid potential failures with real traffic. Balancing risk and sample size is a trade-off in the analysis

- Canary Release

Intuition	**Pros**	**Cons**
Deploy challenger and champion models side by side, with the challenger initially receiving no traffic. Gradually shift traffic from the champion to the challenger (the canary). Monitor challenger performance, and if it looks good, continue until all traffic goes to the challenger.	Easy to understand and implement, especially with existing feature flagging infrastructure. Suitable for models requiring user interaction. Cost-effective with one inference per request. Dynamic traffic adjustment with paired A/B testing	Potential for less rigorous performance determination. Requires careful supervision to avoid accidents. Arguably less safe but easy to rollback

References

[1] Viblo - Optimizer- Hiểu sâu về các thuật toán tối ưu (GD,SGD,Adam,..)
[2] Design Machine Learning System - Continual Learning and Test in Production