VIET NAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

LÂM GIA BỘI – 520H0519

FINAL REPORT
MACHINE LEARNING

HỒ CHÍ MINH CITY, 2023

VIET NAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

LÂM GIA BỘI – 520H0519

FINAL REPORT
MACHINE LEARNING

Instructor:

TS. LÊ ANH CƯỜNG

HỒ CHÍ MINH CITY, 2023

ACKNOWLEDGEMENTS

I would like to say sincere thanks to Mr. Lê Anh Cường for his dedicated guidance and contribution to the completion of this report. Your enthusiasm and heart have been an enormous motivation of encouragement, assisting me in understanding the subject and developing the necessary skills. I want to thank you for all that you've done to help me develop. I hope you will continue to share your enthusiasm and knowledge with future generations of practitioners.

Hồ Chí Minh city, 20 December 2023

Author

(Signature)

WORK COMPLETED AT TON DUC THANG UNIVERSITY

I hereby declare that this research project is my own work, conducted under the scientific guidance of TS. Lê Anh Cường. The research contents and results presented in this project are truthful and have not been published in any form before. The data presented in the tables, serving for analysis, comments, and evaluations, were collected by the author from various sources as explicitly indicated in the reference section.

Furthermore, the project also incorporates some comments, evaluations, as well as data from other authors and different organizing entities, all of which are properly cited and referenced.

If any form of academic dishonesty is identified, I take full responsibility for the content of our project. Ton Duc Thang University is not associated with any copyright violations or infringements that may occur during the execution of this project (if any)."

Ho Chi Minh city, 20 December 2023

Author

(Signature)

WEB SOCKET

SUMMARY

This report will provide an overview of the information we have studied and learned about optimizer model in training learning models. This includes:

1. What is optimizer in training learning models?
2. Overview and study about this method?
3. Comparison between these methods.
4. Application of these methods
5. Continual Learning
6. Test Production

TABLE OF CONTENT

LIST OF IMAGES

LIST OF TABLES

LIST OF ABBREVIATIONS

GD	Gradient Descent
SGD	Stochastic Gradient Descent
Adagrad	Adaptive Gradient Algorithm
RMSprop	Root means square propagation
NLP	Natural Language Processing
CNNs	Convolutional Neural Network

CHAPTER 1: THEORY

1.1 What is the optimizer in training learning models?

- In machine learning model training, optimization is an important part of updating the weight of the model so that the model achieves the best performance on the training data. Optimizers assist in minimizing the loss function. Different optimizers offer different strategies for this process. There are some common methods which are used for optimizing the data:
    - Gradient Descent
    - Stochastic Gradient Descent
    - Momentum
    - Adam
    - Adagrad
    - Root means square propagation.
- Learning rates play an important role in optimizing. For example, when the learning rate is very high, we never reach the true minimum. On the other hand, if the learning rate is too small, we can find the closer minimum point, but in reality, the global minimum is down here, which leads the model to end up at a false minimum.

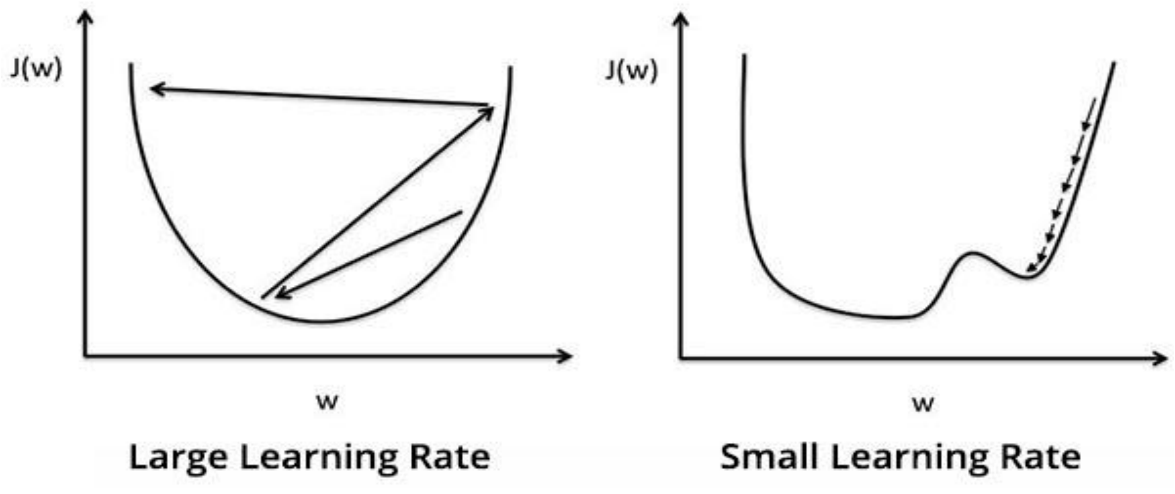

Figure 1.1.1 Difference between large learning rate and small learning rate

1.2 Overview and study

1.2.1 Gradient Descent

- **Definition:** Gradient Descent is an optimization algorithm used to minimize a cost function iteratively. It calculates the gradient (partial derivative) of the cost function with respect to the model parameters and adjusts the parameters in the opposite direction of the gradient to reduce the cost.
- **Formula:**

$$\theta^{new} = \theta^{old} - \underset{\boxed{\alpha = \textit{step size} \text{ or } \textit{learning rate}}}{\underset{\uparrow}{\alpha}} \nabla_\theta J(\theta)$$

Figure 1.2.1 Formula of Gradient Descent

While:

θ: Parameters of the model.

$J(\theta)$: The cost function (or loss function) that we want to minimize.

α: Learning rate.

1.2.2 Stochastic Gradient Descent

- **Definition:** Stochastic Gradient Descent is a variant of gradient descent where the model parameters are updated after each training example. Unlike batch gradient descent, which processes the entire dataset for each parameter update, SGD is faster but introduces more noise due to its reliance on individual data points.

- **Formula:**

$$\theta^{new} = \theta^{old} - \underset{\substack{\uparrow \\ \boxed{\alpha = \textit{step size} \text{ or } \textit{learning rate}}}}{\alpha}\nabla_{\theta} J(\theta)$$

Figure 1.2.2 Formula of Stochastic Gradient Descent

While:

θ: Parameters of the model.

$J(\theta)$: The cost function (or loss function) that we want to minimize.

α: Learning rate.

1.2.3 Momentum

- **Definition:** Momentum is an optimization technique that enhances gradient descent by adding a momentum term. It helps accelerate the convergence by considering the past gradients and smoothing out the updates. This helps to overcome oscillations and converge faster, especially in the presence of noisy or sparse gradients.
- **Formula:**

$$v_t = \gamma v_{t-1} + \eta \nabla J(w_t)$$

$$w_t = w_{t-1} - v_t$$

Figure 1.2.3 Formula of momentum

While:

W_t represents the parameters of the model.

α is the learning rate.
β is the momentum parameter (a hyperparameter typically set between 0 and 1).
v is the velocity vector that accumulates the gradients over time.
$\nabla J(W_t)$ is the gradient of the cost function with respect to the parameters.

1.2.4 Adam

- **Definition:** Adam is an adaptive optimization algorithm that combines ideas from both momentum and RMSprop. It maintains moving averages of the gradients and squared gradients for each parameter. The algorithm adapts the learning rates for each parameter based on their past gradients and squared gradients, providing robust performance across different types of data and architectures.
- **Formula:**

$$
\begin{aligned}
\boldsymbol{g}_n &\longleftarrow \nabla f(\boldsymbol{\theta}_{n-1}) \\
\boldsymbol{m}_n &\longleftarrow (\beta_1/(1-\beta_1^n))\,\boldsymbol{m}_{n-1} + ((1-\beta_1)/(1-\beta_1^n))\,\boldsymbol{g}_n \\
\boldsymbol{v}_n &\longleftarrow (\beta_2/(1-\beta_2^n))\,\boldsymbol{v}_{n-1} + ((1-\beta_2)/(1-\beta_2^n))\,\boldsymbol{g}_n \odot \boldsymbol{g}_n \\
\boldsymbol{\theta}_n &\longleftarrow \boldsymbol{\theta}_{n-1} - a\,\boldsymbol{m}_n/(\sqrt{\boldsymbol{v}_n} + \epsilon)\,,
\end{aligned}
$$

Figure 1.2.4 Formula of Adam

1.2.5 Adagrad

- **Definition:** Adagrad is an optimization algorithm that adapts the learning rates for each parameter based on the historical gradient information. It individually scales the learning rates for each parameter, giving larger updates to parameters with smaller gradients and vice versa. While this can be beneficial for sparsely occurring features, it may lead to overly aggressive updates and a diminishing learning rate over time.

- **Formula:**

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

Figure 1.2.5 Formula of Adagrad

While:

n is a constant.

g_t is the gradient at time t.

ϵ is an error avoidance coefficient (used to avoid division by zero).

G is a diagonal matrix where each element on the diagonal is the square of the partial derivative of the parameter vector at time t.

1.2.6 Root means square propagation

- **Definition:** RMSprop is an optimization algorithm that addresses the diminishing learning rate problem of Adagrad. It uses a moving average of squared gradients to normalize the learning rates. This helps to prevent the learning rates from becoming too small, allowing the algorithm to converge faster, especially in non-convex optimization problems.
- **Formula:**

$$E[g^2]_t = 0{,}9E[g^2]_{t\text{-}1} + 0{,}1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Figure 1.2.6 Formula of RMSprop

1.3 Comparison between these method

Table 1.3.1 Advantages and Disadvantages of these methods

Algorithms	**Advantages**	**Disadvantages**
GD	- Simple and easy to implement.	- Computationally expensive for large datasets - May get stuck in local minima.
SGD	- Faster convergence for large datasets - Less memory requirements.	-High variance in parameter updates, leading to noisy convergence. - May oscillate.
Momentum	- Accelerates convergence, especially in the presence of oscillations. - Helps overcome saddle points.	- Requires tuning of the momentum parameter - May overshoot the minimum.
Adam	- Combines benefits of momentum and RMSprop. - Adaptive learning rates for each parameter - Robust performance.	- Requires tuning of hyperparameters - Computational complexity.

Adagrad	- Adaptive learning rates for each parameter - Effective for sparse data.	- Cumulative sum of squared gradients may lead to diminishing learning rates. - May not perform well on non-convex problems.
RMSprop	- Addresses diminishing learning rates in Adagrad. - Adaptively scales learning rates for each parameter.	- Requires tuning of hyperparameters. - May not perform well with high-dimensional parameter spaces.

1.4 Application

- **Machine Learning and Deep Learning:**

Application: Training neural networks for image recognition, natural language processing, and other machine learning tasks.

Optimization Method: Adam, SGD, RMSprop, and their variants are commonly used to optimize the parameters of deep learning models.

- **Natural Language Processing (NLP):**

Application: Language translation, sentiment analysis, and chatbots.

Optimization Method: Adam and SGD are often employed to optimize models in NLP applications.

- **Computer Vision:**

Application: Object detection, facial recognition, and image segmentation.

Optimization Method: Adam and SGD are frequently used for optimizing convolutional neural networks (CNNs) in computer vision tasks.

- **Recommendation Systems:**

Application: Personalized content recommendations in platforms like Netflix and Amazon.

Optimization Method: SGD and variants are used to optimize collaborative filtering models in recommendation systems.

- **Financial Modeling:**

Application: Predictive modeling for stock prices, risk assessment, and algorithmic trading.

Optimization Method: Various optimization algorithms, including Gradient Descent, are used to optimize financial models.

- **Healthcare:**

Application: Disease prediction, medical image analysis, and drug discovery.

Optimization Method: Gradient Descent and its variants (e.g., Adam, RMSprop): Used in training machine learning models for disease prediction, medical image analysis, and drug discovery.

- **Operations Research:**

Application: Supply chain optimization, logistics planning, and resource allocation.

Optimization Method: Gradient Descent and variants are applied to optimize objective functions in operations research problems.

1.5 Continual Learning

- **Definition**: In computer science, incremental learning is a method of machine learning in which input data is continuously used to extend the existing model's knowledge i.e. to further train the model. It represents a dynamic technique of supervised learning and unsupervised learning that can be applied when training data becomes available gradually over time or its size is out of system memory limits. Algorithms that can facilitate incremental learning are known as incremental machine learning algorithms.

- **How it works?**

Continual Learning is an approach in machine learning that enables models to learn from a continuous stream of data or tasks, sequentially over time. It addresses scenarios where new information arrives, and the model needs to adapt without forgetting previously learned knowledge.

Sequential Learning:

Continual learning involves learning tasks or concepts sequentially, one after another, rather than in a batch setting. This is particularly relevant in dynamic environments where new data becomes available over time.

Retaining Knowledge:

Models trained through continual learning should be able to retain knowledge learned from previous tasks while adapting to new information. This is crucial to prevent catastrophic forgetting, where the model forgets previously learned patterns when exposed to new data.

Adaptability:

Continual learning models need to adapt to changing data distributions and be flexible enough to incorporate knowledge from new tasks without significantly degrading performance on previous tasks.

Resource Efficiency:

In real-world scenarios, resources such as memory and computation are often limited. Continual learning models aim to efficiently use these resources while accommodating new knowledge.

Transfer Learning:

Transfer learning plays a significant role in continual learning. Knowledge gained from solving one task can be beneficial for solving related tasks, promoting knowledge transfer and reuse.

Task Boundaries:

Identifying task boundaries is crucial in continual learning to manage transitions between tasks. The model needs to recognize when a new task is introduced and adjust its parameters accordingly.

- **Pros and Cons**

Table 1.5.1 Pros and Cons of Continual Learning

Pros	Cons
Adaptability - Continual Learning models can adapt to changing data distributions and evolving tasks	**Catastrophic Forgetting** There is a risk of catastrophic forgetting, where the model may lose knowledge from earlier tasks when learning new ones.
Resource Efficiency Models can efficiently use resources by updating parameters selectively, avoiding the need for retraining on the entire dataset.	**Task Interference** Interference between tasks can occur, where learning a new task affects the performance on previous tasks.
Long-Term Knowledge Retention Models can retain knowledge from earlier tasks, preventing	**Increased Complexity** Implementing continual learning systems can be more complex

degradation of performance on previously learned tasks.	compared to traditional static learning.
Flexibility Well-designed continual learning systems are flexible and capable of handling a variety of tasks over time.	**Hyperparameter Tuning** Tuning hyperparameters for continual learning models can be challenging and requires careful consideration.
❖ **Application:** Robotics, Autonomous Vehicles, Natural Language Processing, Computer Vision, Healthcare, Financial Modeling	

1.6 Test production

Test production is a crucial part of the machine learning solution-building process. It is often conducted after the model has been trained to evaluate the model's performance on new data not used during training.

How it Works?

Model Quality:

Test production helps assess the quality and performance of the model on real-world data, providing information about the overall capabilities of the model.

Error Detection and Improvement:

If the model does not perform accurately on new data, test production helps detect errors and provides an opportunity to improve the model.

Biased Data and Adjustment Management:

Test production can help identify issues related to biased or non-representational data and manage the process of model adjustment.

Continuous Testing:

Continuous testing can be deployed to ensure that the model maintains high performance after each update or when new data is introduced.

Performance Management:

Test production provides insights into the model's performance in a production environment, helping manage performance and respond quickly to changes.

REFERENCES

Vietnamese:

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

2. https://machinelearningcoban.com/2017/01/12/gradientdescent/

3. https://ndquy.github.io/posts/gradient-descent-2/

English

1. TDTU Slides

2. https://www.youtube.com/watch?v=nra0Tt3a-Oc&t=98s

3. https://www.youtube.com/watch?v=vjaq03IYgSk