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

FINAL PROJECT

MACHINE LEARNING

Instructor: **Mr. Le Anh Cuong**

Student: **Le Quang Huy– 521H0238**

Group **:**

HO CHI MINH CITY, 2023

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

FINAL PROJECT

MACHINE LEARNING

Instructor: **Mr. Le Anh Cuong**

Student: **Le Quang Huy – 521H0238**

Group **:**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENT

The first sincere thanks I want to give to Mr. Le Anh Cuong, who enthusiastically taught and worked tirelessly to give me enough tools and skills to complete this report. He played an important role in improving my mathematical logic and knowledge. The second thanks I would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving me the opportunity to do this report.

Because the impact of the epidemic is too great, my report will have some errors, I am very open to receiving feedback from teachers so that I can improve my report writing skills.

Finally, I wish you good health and success in your noble career.

Ho Chi Minh city, 7th January, 2022

Author

(Sign and write full name)

Le Quang Huy

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 a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

Should any frauds were 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, 8th January, 2022

Author

(Sign and write full name)

Le Quang Huy

CONFIRMATION AND ASSESSMENT SECTION

Instructor confirmation section

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh January, 2022

(Sign and write full name)

Evaluation section for grading instructor

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh January 2022

(Sign and write full name)

x

INDEX

x

Introduction to the Importance of Optimizer Methods in Machine Learning

In the realm of machine learning and neural networks, Optimizers play a crucial role in enhancing and optimizing models. An Optimizer is an algorithm or method used to modify attributes of the learning model, such as weights and learning rate, aiming to minimize errors (loss) during the training process. The selection and optimization of an Optimizer can significantly impact the model's performance, learning speed, and convergence ability.

Different Optimizers are suitable for various types of problems and data. For instance, some Optimizers perform well with large datasets, while others are more effective with highly complex data. Understanding the various types of Optimizers and their operation can aid researchers and machine learning engineers in selecting the most appropriate method for specific situations, thereby increasing the effectiveness of machine learning models.

x

Objectives and Scope of the Study

Objectives:

- Exploration and Understanding: The study aims to provide an in-depth view of how different Optimizer methods operate in machine learning.
- Comparison and Analysis: Compare the performance, strengths, and weaknesses of various Optimizers to determine under what conditions they operate most effectively.
- Practical Application: Evaluate how these Optimizers are applied in specific machine learning problems and real-world scenarios.

Scope:

- Key Optimizer Methods: Focus on popular methods like SGD, Momentum, Adagrad, RMSprop, Adam, and their variants.
- Theoretical and Empirical Comparison: Combine both theoretical analysis and case studies to assess effectiveness.
- Considering Both New and Old: Explore both traditional methods and the latest advancements in the field.

x

=> By focusing on these elements, the study will provide a comprehensive view of the importance and application of Optimizers in machine learning, also supporting the academic community and researchers in applying them more effectively.

Theoretical Foundation

Definition and Role of Optimizer in Machine Learning

An optimizer in machine learning is an algorithm or method used to update and adjust the parameters of a model (such as weights and biases) to minimize the loss function. The loss function evaluates the accuracy of the model's predictions compared to the actual data. The goal of the optimization process is to find a set of parameters such that the loss function reaches its lowest value, thereby improving the model's performance.

The Role of Optimizer in Machine Learning and a Detailed Introduction to the Gradient Descent Algorithm

The Role of Optimizer in Machine Learning

Optimizers in machine learning are indispensable, playing a crucial part in the success of machine learning models. Their key roles include:

x

Guiding Learning:

- Optimizers help the model "learn" from data by adjusting its parameters (such as weights and biases) based on training data.
- This process ensures that the model can accurately derive insights from data and improve its predictions or classifications.

Performance Optimization:

- The primary goal of an Optimizer is to minimize the loss function, meaning to make the model more accurate in its predictions.
- Minimizing error helps the model achieve optimal performance, providing more reliable results for practical problems.

Controlling the Learning Process:

- Optimizers also have the responsibility of adjusting the learning rate, a crucial factor in balancing learning speed and depth.
- This helps prevent issues like overfitting (too complex) or underfitting (too simple), ensuring the model learns effectively from data without being influenced by noise or specific characteristics of the training dataset.

Detailed Introduction to the Gradient Descent Algorithm

Gradient Descent (GD) is one of the most basic and widely used optimization algorithms in machine learning. It is based on the principle of adjusting model parameters according to the gradient of the loss function.

x

Calculating the Gradient:

- The gradient is the partial derivative of the loss function with respect to each parameter.
- It indicates the direction and magnitude of the fastest change in the loss function, helping determine the most effective parameter update direction.

Updating Parameters:

- Parameters are adjusted in the direction opposite to the gradient to minimize the loss function.
- This continuous update process during training leads to the discovery of optimal parameters.

Repeating the Process:

- This process of computation and update is repeated multiple times until a point is reached where the loss function no longer decreases significantly.

Variants of Gradient Descent

Batch Gradient Descent:

- Calculates the gradient over the entire dataset.
- Ensures stability but is resource-intensive and time-consuming, not suitable for large datasets.

x

Stochastic Gradient Descent (SGD):

- Calculates the gradient and updates parameters after each data sample.
- Increases learning speed but reduces stability, potentially leading to significant fluctuations in the learning process.

Mini-batch Gradient Descent:

- A combination of Batch and SGD, calculating the gradient over a batch of data.
- Provides a balance between performance and stability, suitable for most machine learning problems.

Gradient Descent and its variants are foundational for many more complex Optimizer methods, playing a vital role in optimizing machine learning models. They help make models more effective and accurate in solving real-world problems.

Optimizer Methods in Machine Learning

Gradient Descent (GD)

- Basic Explanation: GD is a fundamental optimization method where model parameters are updated based on the gradient of the loss function. The key idea is to move the parameters in the opposite direction of the gradient to minimize the loss function.
- Applications: Suitable for problems with simple data and models where computing the global gradient is feasible.

x

Stochastic Gradient Descent (SGD)

- How it Works: In SGD, the gradient is computed and updated after each data point or a small batch of data. This introduces significant fluctuations in the update process, which can help escape local optima.
- Difference from GD: SGD reduces the computational cost per update and often converges faster but can be less stable.

Mini-batch Gradient Descent

- Combination of GD and SGD: Compute the gradient on a fixed-sized batch of data, reducing the update fluctuation compared to SGD.
- Advantages: Balances computational efficiency and stability, suitable for most machine learning tasks.

Momentum and Nesterov Accelerated Gradient (NAG)

- Convergence Speed Improvements: Both algorithms attempt to overcome slow convergence by accumulating previous gradients, speeding up updates. NAG further improves this by predicting the update direction.
- Applications: Particularly effective in situations with many local minima and maxima in the loss surface.

Adagrad and RMSprop

- Learning Rate Adaptation: Adagrad adjusts the learning rate based on the frequency of each parameter, making the optimization more effective for parameters that change infrequently. RMSprop improves this by using a running average of recent gradients.
- Advantages: Suitable for problems with highly varying gradients, offering flexible learning rate adjustments.

Adam and Nadam

- Combining Advantages: Adam combines features from both Momentum and RMSprop, providing a flexible and efficient update mechanism. Nadam combines Adam with NAG for refined parameter updates.
- Applications: Highly popular in the machine learning community, Adam and Nadam are often used in training complex models like deep neural networks.

x

New Methods

- Latest Research: Recently, there have been developments in methods such as Lookahead, AdamW, LARS, and LAMB, aiming to improve convergence speed and adaptability for complex models.
- Characteristics: These methods often focus on addressing the limitations of traditional optimizers, such as optimization issues in large and complex parameter spaces.

Comparison and Evaluation of Optimizer Methods

Optimizer	Performance	Convergence Speed	Handling Specific Problems
Gradient Descent (GD)	Good with less complex data	Slow, especially with large data	Inefficient with large or complex data
Stochastic Gradient Descent (SGD)	Effective with large data	Fast but unstable	Good in reducing overfitting
Mini-batch Gradient Descent	Balance between performance and stability	Good, more stable than SGD	Suitable for both large and small data

x

Momentum/NAG	Accelerates convergence, minimizes risks	Fast and more stable than SGD	Effective in complex terrains
Adagrad/RMSprop	Good with uneven data	Good, self-adjusts learning rate	Highly effective with varying data characteristics
Adam/Nadam	High performance, versatile	Very fast and stable	Optimal for most problems

Detailed Review:

- Gradient Descent (GD): Suitable for simple problems with small datasets. However, for large datasets, GD can be slow and inefficient, especially when encountering local optima.

- Stochastic Gradient Descent (SGD): Provides high efficiency with large datasets due to its approach of updating parameters after each sample. However, the significant fluctuations in the update process can make SGD's convergence unstable.

- Mini-batch Gradient Descent: Combines the advantages of GD and SGD, minimizing parameter update fluctuations while balancing efficiency and stability. This is a popular choice for many machine learning tasks.

x

- Momentum and Nesterov Accelerated Gradient (NAG): Both methods accelerate convergence and reduce the risk of getting stuck in local minima. NAG, in particular, offers predictive update directions, optimizing the learning process.

- Adagrad and RMSprop: Both are excellent at handling varying gradients, with the ability to adapt the learning rate. RMSprop, with its approach of updating based on recent gradients, addresses some limitations of Adagrad.

- Adam and Nadam: Adam combines the strengths of Momentum and RMSprop, providing a balanced and efficient approach for most types of problems. Nadam, a variant of Adam, adds the predictive capabilities of NAG, making it extremely powerful in complex scenarios.

Real-World Applications

Gradient Descent Optimizer (GD):

Application: Gradient Descent is widely used in machine learning and deep learning to optimize loss functions. For example, in training neural networks, GD is used to update the weights and biases of network layers to minimize prediction errors.
Performance Analysis: The performance of GD depends on the learning rate and the shape of the loss function. If the learning rate is too large, it can lead to instability or non-convergence. If the loss function is not smooth, GD may get stuck in local optima.

x

Stochastic Gradient Descent (SGD):

Application: SGD is popular in training large neural networks with big datasets. It randomly selects a sample from the data to update weights, saving time and memory.
Performance Analysis: SGD can oscillate around the optimal point due to the randomness in data. To improve performance, variants like Mini-batch SGD or learning rate schedules are often used.

Adam Optimizer:

Application: Adam is one of the popular optimizers in deep learning. It combines both Momentum and RMSProp to balance learning rate and momentum.
Performance Analysis: Adam generally works well in most cases but requires tuning hyperparameters like beta1, beta2, and epsilon to achieve the best results.

Challenges and Future Directions

Challenges in Using Optimizers:

- Hyperparameter optimization: Choosing the learning rate and other optimizer parameters is a challenge. Poor choices can lead to loss in performance.

- Non-convex optimization: Non-convex loss functions can result in local optima, making the search for a global minimum difficult.

- Gradient vanishing/exploding: In deep models, gradient vanishing can occur, where gradients decrease significantly when backpropagating to the first layer.

Research Directions and Improvements for the Future:

- Automated hyperparameter optimization: Develop automated methods to optimize hyperparameters for various optimizers.

- Optimizer improvements: Research and develop new optimizers that converge faster and are more efficient in optimization tasks.

x

- Addressing gradient issues: Investigate approaches to handle gradient vanishing/exploding problems, such as using prior knowledge to guide the training process.

Research about Continual Learning and Test Production in building a machine learning solution to solve a particular problem

Continual Learning in Machine Learning

Continual Learning, also known as Lifelong Learning, represents an advanced paradigm in the field of machine learning and artificial intelligence. It emphasizes the development of models that retain and refine their knowledge over time, akin to human learning. This approach is critical in a rapidly changing world where data patterns and distributions are constantly evolving.

Definition and Importance:

- Broad Scope: Continual Learning goes beyond traditional static learning paradigms. In traditional learning, a model is trained once on a dataset and then deployed without any further adaptation. In contrast, Continual Learning models are expected to evolve as they encounter new data.

- Mimicking Human Learning: The approach is inspired by human cognitive abilities. Just as humans don't forget how to ride a bike when they learn how to drive a car, Continual Learning models aim to retain knowledge across tasks. This makes them particularly

x

suited to complex, real-world applications where the data and the tasks can change over time.

- Real-World Necessity: In practical scenarios like recommendation systems, the nature of data changes as user preferences evolve. In fraud detection, tactics employed by fraudsters evolve, requiring models to adapt quickly. Natural language processing systems also need to adapt to new slang, terms, and usage over time.

Challenges:

- Catastrophic Forgetting: This is a significant hurdle where models, upon learning new information, overwrite the old knowledge. This is akin to a person forgetting their native language when they start learning a new one.

- Model Adaptation: It's a delicate balance to adapt to new data without losing performance on the old data. The model must be flexible yet stable.

- Data Imbalance and Availability: Often, older data may not be available due to storage issues or privacy concerns. Even when available, balancing the old and new data without biasing the model is a complex task.

Approaches:

- Regularization Techniques: Advanced methods like Elastic Weight Consolidation (EWC) and Synaptic Intelligence impose constraints on the update of weights, particularly those crucial for previous tasks, thereby preserving old knowledge.

x

- Rehearsal Methods: These involve retaining a subset of the old data (replay) or creating synthetic versions of it (pseudo-rehearsal). The model is then trained on a mix of old and new data, helping it remember and adapt.

- Architectural Strategies: Methods like Progressive Neural Networks construct a new neural network for each task while sharing weights with previous networks, creating a web of knowledge that grows with each new task.

- Dynamic Architectures: Some approaches dynamically modify the network architecture by adding new neurons or paths to accommodate new knowledge, thereby preserving the old structure and knowledge.

Advanced Applications:

- Personalized Medicine: Continual Learning can be pivotal in creating personalized treatment plans that evolve with the patient's changing medical history and conditions.

- Robotic Autonomy: In robotics, Continual Learning helps in adapting to new environments and tasks, crucial for search and rescue missions, planetary exploration, and more.

x

- Evolving Threat Detection: In cybersecurity, as threats evolve, models must adapt in real-time to detect and mitigate new kinds of attacks.

- Adaptive Content Curation: For platforms like news aggregators and social media, Continual Learning helps in adapting to the ever-changing interests and behaviors of users, providing personalized content that evolves daily.

Test Production in Machine Learning

Test Production, often termed as model deployment or productionizing, is a fundamental yet intricate phase in the lifecycle of a machine learning (ML) project. It's the stage where a trained ML model transitions from a controlled development setting to an operational environment, facing real-world data and users.

Expanded Definition and Context:

- Beyond Development: Test Production isn't just about deploying a model; it's about integrating it into the broader context of an application or a business process. This integration demands the model to perform reliably under varying operational conditions.

- Operationalizing ML: This stage bridges the gap between a theoretical model and a practical, usable tool. It involves not only the deployment of the model but also the setup of necessary infrastructure for data ingestion, processing, and handling user requests.

x

- End-to-End Workflow: The process encompasses everything from initial deployment to updating the model in production. It requires careful planning, as the way data flows through the system and how predictions are delivered can significantly affect the overall performance and user experience.

Challenges:

1. Model Integration:

- Integration into existing systems often requires ensuring compatibility with various software components and data formats.

- Ensuring that the deployed model can communicate effectively with other parts of the system, such as databases and front-end interfaces.

2. Scalability:

- The model must handle varying loads, potentially scaling from a few requests per day to thousands per second.

- Scalability challenges also involve managing resources efficiently to balance cost and performance.

3. Latency:

- Particularly crucial for real-time applications like fraud detection or autonomous vehicles, where decisions must be made rapidly.

- Low latency is often a balancing act against model complexity and accuracy.

x

4. Monitoring and Maintenance:

- Models in production can degrade over time due to changes in underlying data patterns, a phenomenon known as concept drift or data drift.

- Regular monitoring is required to ensure the model remains accurate and relevant.

Best Practices:

1. Continuous Monitoring:

- Implementing tools and processes for tracking model performance metrics in real-time.

- Setting up alerts for anomalies in model predictions or performance metrics.

2. Version Control:

- Maintaining versions of both the model and the data it was trained on.

- Version control systems help roll back to previous versions if a new model version performs poorly.

3. Automated Retraining Pipelines:

- Automating the process of retraining models with new data to ensure they remain up-to-date.

- These pipelines can include validation steps to verify the performance of the retrained model before it is pushed to production.

x

4. A/B Testing:

- Testing different versions of models under real-world conditions to evaluate performance improvements or changes.

- A/B testing helps in making data-driven decisions about which model version should be deployed.

5. Tools and Frameworks:

- Utilizing advanced tools such as TensorFlow Serving for serving machine learning models, AWS SageMaker for end-to-end machine learning workflows, Azure ML for robust cloud-based deployment, and Docker for containerizing applications.

- These tools facilitate not just the deployment but also the monitoring, scaling, and updating of models in production environments.

6. Documentation and Knowledge Sharing:

- Maintaining comprehensive documentation on the deployment process, model architecture, and operational procedures.

- Encouraging knowledge sharing among team members for a better understanding of the deployment nuances and quicker problem resolution.

x

REFERENCES

Tiếng việt:

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

https://csdlkhoahoc.hueuni.edu.vn/data/2021/5/BaiDangHoiThao.pdf

Tiếng anh:

https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html

https://paperswithcode.com/task/continual-learning#:~:text=Continual%20Learning%20(also%20known%20as,anymore%20during%20training%20new%20ones