VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

****

**INTRODUCTION TO MACHINE LEARNING**

**FINAL REPORT**

*Instructor*: **Mr. LE ANH CUONG**

*Implementer*: **HUYNH TRAN MINH TIEN – 520H0583**

**NGUYEN TRUNG TIN – 520H0589**

*Class***: 20H50202**

*Course***: 24**

**HO CHI MINH CITY, YEAR 2023**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

****

**INTRODUCTION TO MACHINE LEARNING**

**FINAL REPORT**

Instructor: **Mr. LE ANH CUONG**

*Implementer*: **HUYNH TRAN MINH TIEN – 520H0583**

**NGUYEN TRUNG TIN – 520H0589**

*Class***: 20H50202**

Course**: 24**

**HO CHI MINH CITY, YEAR 2023**

ACKNOWLEDGEMENT

In order to complete the report completely, we would like to express our sincere thanks to Mr. Le Anh Cuong for enthusiastically supporting us to complete this report.

During the process of making the report, we tried to focus on researching and understanding relevant sources. However, in the process of implementation, errors cannot be avoided, we are looking forward to the guidance and suggestions from the teacher to make the report more complete.

**REPORT COMPLETED**

**AT TON DUC THANG UNIVERSITY**

We hereby declare that this is our own report and is guided by Mr. Le Anh Cuong. The research contents and results on 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 themselves from different sources, clearly stated in the reference section.

In addition, the report also uses a number of comments, assessments as well as data from other authors, other agencies, and organizations, with citations and source annotations.

**If we find any fraud, we will take full responsibility for the content of our report.** Ton Duc Thang University is not related to copyright and copyright violations caused by us during the implementation process (if any).

*Ho Chi Minh City, December 24th 2023*

*Author*

*Huynh Tran Minh Tien*

*Nguyen Trung Tin*

TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

**TEACHER'S CONFIRMATION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, …………………

(sign and write name)

**TEACHER'S ASSESSMENT**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, …………………

(sign and write name)

SUMMARY

Individual Research Paper: Evaluation of Optimizer Methods and Continual Learning

The individual research focuses on understanding and comparing various optimizer methods used in training machine learning models. Methods such as SGD, Adam, and RMSProp are extensively evaluated based on crucial criteria such as convergence speed and the ability to avoid local optima. Additionally, the paper provides technical insights into Continual Learning and Test Production when constructing machine learning solutions, emphasizing continuous learning capabilities and ensuring model accuracy across testing scenarios.

Group Machine Learning Problem: Prediction with Diverse Data

The group's machine learning problem poses a challenge with diverse data, incorporating both numerical and categorical data. Initially, statistical analysis and graphs are employed to comprehend the problem and data. Subsequently, the group applies basic machine learning models and Ensemble Learning, exploring the performance of both Feed Forward Neural Network and Recurrent Neural Network in the context of complex data. Measures to prevent overfitting are implemented to ensure model generalization and stability. Finally, the group examines error cases, proposes solutions, and evaluates the results, improving the model's accuracy.

Collectively, both studies contribute to a comprehensive understanding of optimization methods and continuous knowledge in the field of machine learning. Furthermore, they provide flexible and applicable solutions for diverse prediction problems

TABLE OF CONTENT

[ACKNOWLEDGEMENT 1](#_Toc27473)

[TEACHER'S CONFIRMATION AND ASSESSMENT SECTION 3](#_Toc27545)

[SUMMARY 4](#_Toc6184)

[TABLE OF CONTENT 5](#_Toc8762)

[LIST OF TABLES, PICTURES, GRAPHS 7](#_Toc19699)

[CHAPTER 1. INDIVIDUAL RESEARCH 8](#_Toc18793)

[1.1 Optimizer methods 8](#_Toc6095)

[1.1.1 Definition 8](#_Toc29749)

[1.1.2 How do Optimizers work? 9](#_Toc3948)

[1.1.3 Types of Optimization methods 10](#_Toc22273)

[1.1.4 Stochastic Gradient Descent (SGD) 11](#_Toc30023)

[1.1.5 Mini-batch Gradient Descent 12](#_Toc14674)

[1.1.6 Momentum 13](#_Toc22221)

[1.1.7 Adam 13](#_Toc12022)

[1.1.8 RMSProp 14](#_Toc3165)

[1.1.9 Adagrad 14](#_Toc7027)

[1.1.10 Pros and Cons 15](#_Toc11648)

[1.2 Continual Learning 16](#_Toc23008)

[1.2.1 Introduction 16](#_Toc3032)

[1.2.2 The role of Continual Learning 17](#_Toc29847)

[1.2.3 The catastrophic forgetting phenomenon 17](#_Toc4756)

[1.2.4 Techniques help prevent catastrophic forgetting 18](#_Toc29536)

[1.2.5 Types of Continual Learning 19](#_Toc5735)

[1.2.6 Process of Continual Learning 20](#_Toc15361)

[1.2.7 Why should ML models be retrained? 21](#_Toc16981)

[1.2.8 Advantages of Continual Learning 21](#_Toc8289)

[1.2.9 Limitations of Continual Learning 22](#_Toc28631)

[1.2.10 Applications of Continuous Learning 22](#_Toc20119)

[1.3 Test Production 24](#_Toc27936)

[1.3.1 Introduction 24](#_Toc15005)

[1.3.2 Step of Test Production 25](#_Toc2697)

[CHAPTER 2. GROUP PROBLEMS 27](#_Toc5768)

[REFERENCE 28](#_Toc20597)

LIST OF TABLES, PICTURES, GRAPHS

**LIST OF PICTURES**

**LIST OF TABLES**

CHAPTER 1. INDIVIDUAL RESEARCH

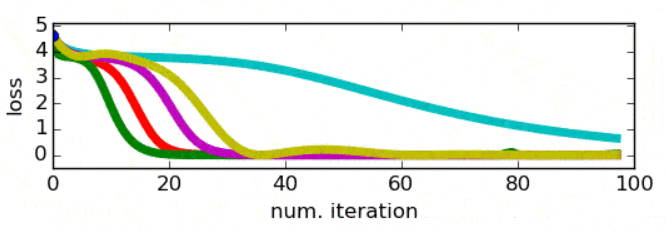
* 1. Optimizer methods
     1. Definition

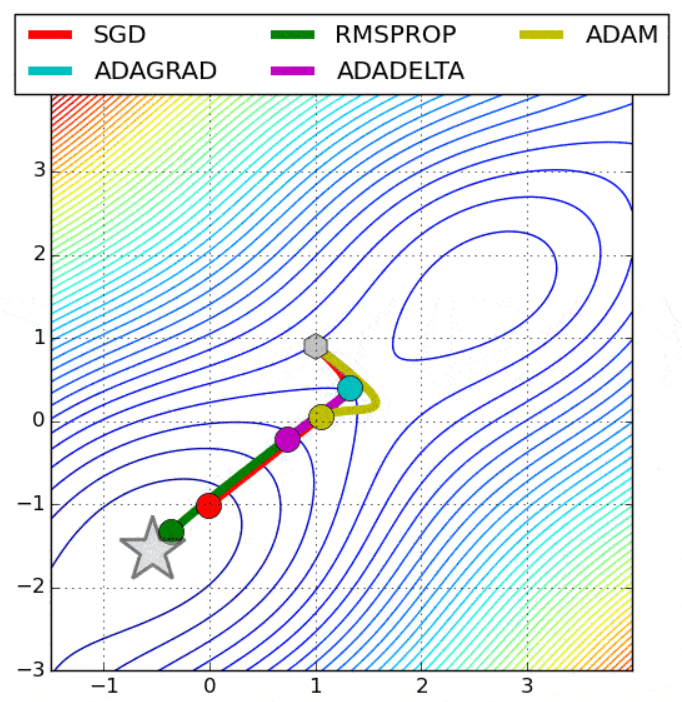
The primary objective of machine learning is to construct a model that demonstrates high performance and delivers accurate predictions within a specific set of scenarios. Previously, our understanding has been centered around how loss functions provide insights into the model's fit with the current dataset. Now, upon recognizing inadequate performance in the current instance, we employ optimization methods to create a precise model with a reduced error rate.

In machine learning optimization, the adjustment of hyperparameters is undertaken to minimize the loss and maximize accuracy. Minimizing the cost function is crucial as it signifies the disparity between the actual value of the estimated parameter and the model's prediction.

Optimizers, which are techniques or algorithms, play a vital role in altering neural network attributes like weights and learning rate to diminish the loss. Following the calculation of loss, the optimization process involves adjusting weights and biases within the same iteration.

Optimization unfolds as an iterative training process, leading to the evaluation of maximum and minimum functions.





* + 1. How do Optimizers work?

Optimizers function to reduce the discrepancy between the model's predictions and actual values, aiming to minimize the overall error. Through iterative adjustments of the model parameters, guided by gradients calculated during backpropagation, optimizers steer the model towards a configuration where the loss is minimized. This process continues until the model converges to an optimal state, achieving the best possible fit to the training data.

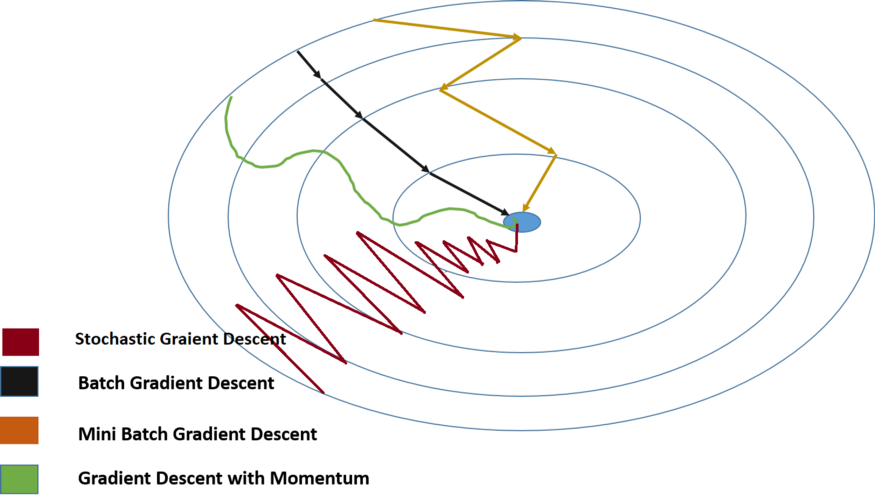
Step of Optimization

1. Commencement: The optimizer starts with the model's current state, represented by parameters like weights and biases, analogous to knobs controlling predictions.
2. Loss Computation: The optimizer calculates the loss function, a measure of the variance between model predictions and actual values, akin to a gauge indicating proximity to the summit.
3. Gradient Descent: In response to the loss, the optimizer takes a small step in the direction that minimizes it, guided by the gradient pointing towards the steepest downhill slope on the loss landscape.
4. Parameter Adjustment: Utilizing step information, the optimizer refines the model's parameters, mirroring a climber adjusting grip and footwork based on slope direction.
5. Iterative Refinement: This cyclical process of calculating loss, taking steps, and updating parameters continues iteratively. With each iteration, the optimizer strives to enhance the model's predictions and move closer to peak accuracy.
   * 1. Types of Optimization methods

**There are some common types:**

**First-order methods: are a class of algorithms that use only the first derivative (gradient) of the objective function to guide their search for an optimal solution. They are widely used in machine learning due to their simplicity, efficiency, and scalability to large-scale problems**

* **Gradient Descent: adjusting model parameters along the steepest gradient direction.**
* **Stochastic Gradient Descent (SGD): updates model parameters using the gradient of the loss function with respect to a randomly chosen individual data point.**
* **Batch Gradient Descent: calculating the gradient of the loss function using the entire training dataset in each iteration to update parameters.**
* **Mini-batch Gradient Descent: computing the gradient on a randomly selected small batch of data points.**
* **Momentum: enhances gradient descent by accumulating past gradients, promoting faster convergence.**



**Adaptive Learning Rate methods: are a family of optimization techniques that intelligently adjust the learning rate during model training, rather than using a fixed value throughout.**

* **Adam (Adaptive Moment Estimation): combines features of RMSProp and Momentum. It adapts learning rates per parameter, incorporating a moving average of past gradients for efficient model training.**
* **RMSProp (Root Mean Square Propagation): adjusts learning rates by normalizing the gradient using the moving root mean square of past squared gradients.**
* **Adagrad: adapts learning rates for each parameter based on the historical sum of squared gradients. It is particularly suitable for sparse data.**

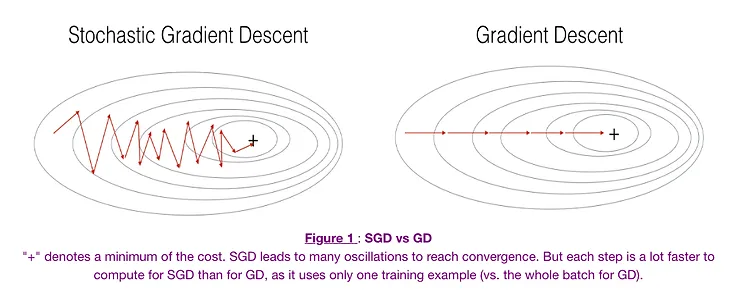
**Second-order methods: are a class of optimization techniques that leverage more information about the shape of the objective function (the function you're trying to minimize or maximize) than first-order methods. They do this by incorporating not only the gradient (the first derivative), but also the Hessian (the second derivative) of the objective function.**

* **Newton's Method: Newton's method is a classic second-order optimization algorithm. It involves computing the Hessian matrix, which contains second-order partial derivatives, and using it to determine the optimal step size and direction for parameter updates.**
* **Quasi-Newton Methods: Due to the computational cost of computing the full Hessian matrix, quasi-Newton methods provide approximations of the inverse Hessian without explicitly calculating it. Examples include the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the Limited-memory BFGS (L-BFGS) algorithm.**
  + 1. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) stands as an iterative optimization technique widely employed in machine learning and deep learning. It represents a deviation from traditional gradient descent by updating model parameters (weights) based on the gradient of the loss function, computed on a randomly selected subset (mini-batch) of the training data rather than the entire dataset.

The fundamental concept of SGD involves sampling a small, random subset (mini-batch) from the training data and computing the gradient of the loss function relative to the model parameters using solely that subset. This gradient guides the parameter update, and the process repeats with new random mini-batches until the algorithm converges or meets a predefined stopping criterion.

SGD offers several advantages compared to standard gradient descent, including faster convergence and lower memory requirements, particularly for extensive datasets. It also exhibits resilience to noisy and non-stationary data, enabling it to navigate away from local minima. However, it may necessitate more iterations for convergence than gradient descent, and careful tuning of the learning rate is essential to ensure successful convergence.

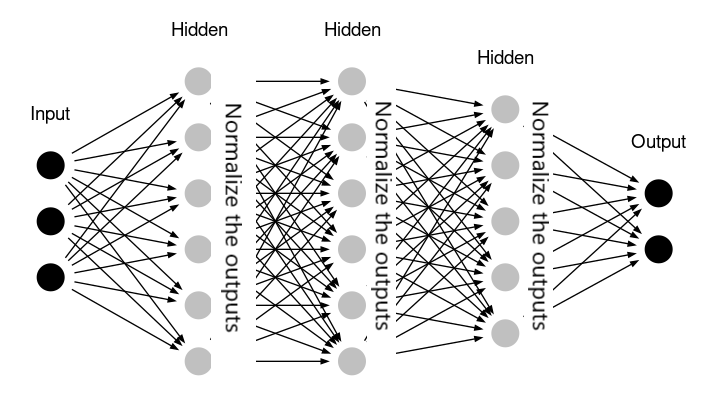


* + 1. Mini-batch Gradient Descent

Mini-batch Gradient Descent (MBGD) strikes a balance between stochastic and batch gradient descent, leveraging the advantages of both. It randomly selects training samples in mini-batches from the entire dataset, aiming to emulate Batch Gradient Descent.

By utilizing only a subset of the data in each iteration, mini-batch SGD requires fewer rounds compared to batch gradient descent. This approach proves more efficient and reliable than both stochastic and batch gradient descent algorithms. Notably, the batching strategy enhances efficiency by negating the need to load the entire training dataset into memory.

Mini-batch Gradient Descent introduces a noise level in the cost function between batch and stochastic gradient descent. While noisier than batch gradient descent, it is smoother than stochastic gradient descent. This balance allows mini-batch SGD to achieve an excellent compromise between speed and precision, making it a widely preferred choice in practical applications.

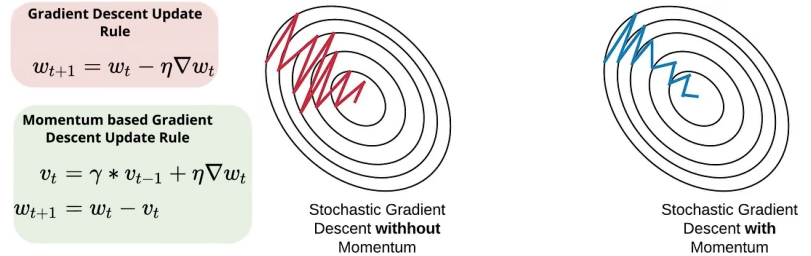


* + 1. Momentum

Momentum is a strategy employed in machine learning and deep learning to hasten neural network training. This approach involves integrating a portion of the prior weight update into the current update during the optimization process.

In momentum optimization, the gradient of the cost function is computed for each weight in the neural network. Instead of directly adjusting weights based on the gradient, a new variable, the momentum term, is introduced. This term, acting as a moving average of gradients, accumulates past gradients to guide the weight updates. It can be conceptualized as the optimizer's velocity, accumulating momentum during descent to minimize oscillations and facilitate faster convergence.

Momentum optimization proves valuable in scenarios with a noisy optimization landscape or rapidly changing gradients. Its smoothing effect on the optimization process prevents the optimizer from becoming ensnared in local minima. Overall, momentum serves as a potent optimization technique, expediting deep neural network training and improving performance.

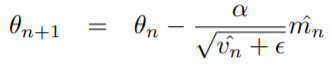


* + 1. Adam

Adam, short for Adaptive Moment Estimation, is a highly effective optimization algorithm utilized in machine learning and deep learning to optimize the training of neural networks.

Adam ingeniously merges the principles of both momentum and RMSProp. It keeps track of a moving average for the first and second moments of the gradient, representing the mean and variance, respectively. The first moment's moving average, akin to momentum, sustains the optimizer's direction even with diminishing gradients. Simultaneously, the second moment's moving average, resembling RMSProp, facilitates adaptive learning rate scaling based on gradient variance.

Adam incorporates a crucial bias correction step to rectify biases towards zero in the initial optimization stages. This correction enhances the algorithm's performance during early training. Notably, Adam stands out for its autonomy in hyperparameter tuning, eliminating the need for manual adjustments to parameters like learning rate decay or momentum coefficient. This feature contributes to its widespread popularity.

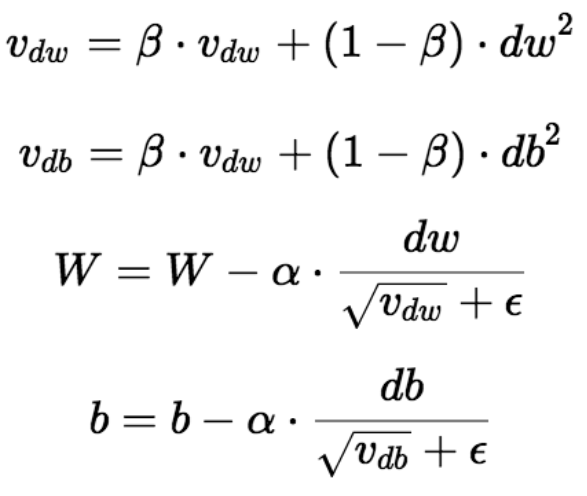


* + 1. RMSProp

RMSProp, short for Root Mean Square Propagation, is a pivotal optimization algorithm employed in machine learning and deep learning to enhance the training of neural networks.

Distinct from Adagrad and Adadelta, RMSProp dynamically adjusts the learning rate for each parameter during training. Instead of accumulating all past gradients like Adagrad, RMSProp calculates a moving average of squared gradients. This methodology ensures a more gradual adjustment of the learning rate, preventing rapid decreases.

A notable strength of RMSProp lies in its ability to handle non-stationary objectives, where the underlying function that the neural network seeks to approximate changes over time. Unlike Adagrad, which may converge too quickly in such scenarios, RMSProp adapts the learning rate to the evolving objective function. The inclusion of a decay factor further fine-tunes the influence of past gradients, assigning more weight to recent gradients and less to older ones.



* + 1. Adagrad

Adagrad, or Adaptive Gradient, stands out as an optimization algorithm in machine learning and deep learning. Its primary function lies in optimizing the training process of neural networks through the dynamic adjustment of learning rates for each parameter.

The Adagrad algorithm achieves adaptability by scaling the learning rate for each parameter based on historical gradients. Parameters with large gradients receive a reduced learning rate, whereas those with smaller gradients are assigned a higher learning rate. This nuanced approach prevents a rapid decline in the learning rate for frequently encountered parameters, fostering accelerated convergence during training.

Adagrad excels in scenarios involving sparse data, where certain input features are infrequent or missing. Its ability to flexibly adapt the learning rate for each parameter proves beneficial in effectively handling sparse data, contributing to improved overall performance in neural network training.

IMG_256

IMG_256

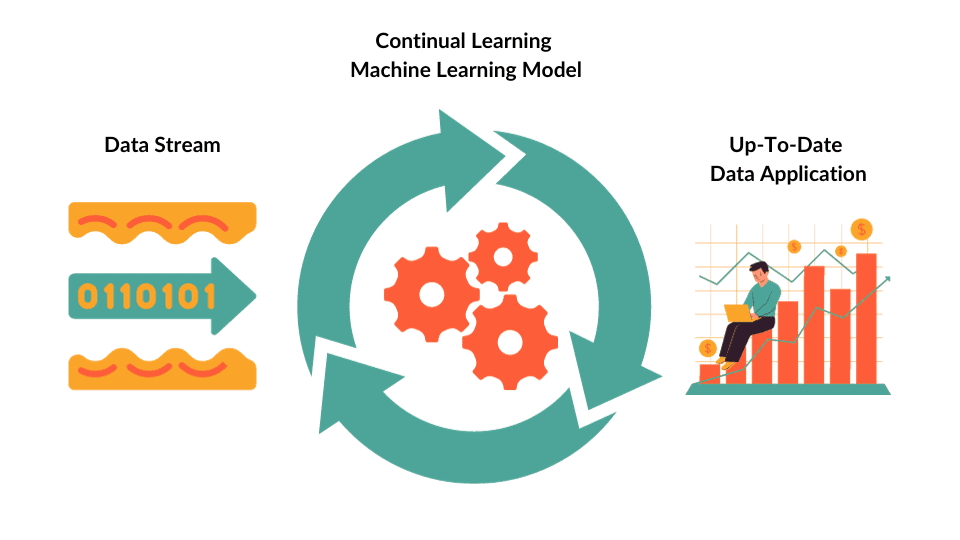
* + 1. Pros and Cons

|  |  |  |
| --- | --- | --- |
| ****Optimizer**** | ****Pros**** | ****Cons**** |
| Stochastic Gradient Descent (SGD) | - Simple to implement and computationally efficient. - Effective for large datasets with high dimensional feature space.  - Memory requirement is less compared to Gradient Descent algorithm. | - SGD can get stuck in local minima. - High sensitivity to initial learning rate.  - Time taken by 1 epoch is large compared to Gradient Descent |
| Mini-batch Gradient Descent (MBGD) | - Less time taken to converge the model  - Requires medium amount of memory  - Frequently updates the model parameters and also has less variance. | - If the learning rate is too small then convergence rate will be slow.  - It doesn't guarantee good convergence |
| Momentum | - Reduces oscillations and high variance of the parameters - Faster convergence for ill-conditioned problems. | - Increases the complexity of the algorithm. |
| Adam | - Efficient and straightforward to implement. - Applicable to large datasets and high-dimensional models. - Good generalization ability. | - Requires careful tuning of hyperparameters. |
| RMSProp | - Adaptive learning rate per parameter that limits the accumulation of gradients. - Effective for non-stationary objectives. | - Can have a slow convergence rate in some situations. |
| Adagrad | - Adaptive learning rate per parameter. - Effective for sparse data. | - Accumulation of squared gradients in the denominator can cause learning rates to shrink too quickly. - Can stop learning too early. |

* 1. Continual Learning
     1. Introduction

Continuous Machine Learning (CML) is a model in machine learning that fosters perpetual adaptation and improvement. It addresses the challenge of catastrophic forgetting, allowing models to accumulate new data and skills without erasing past experiences, similar to how humans learn. In practical applications, CML enhances language models, vision systems, and robotics by adapting to evolving trends, environments, and tasks. This adaptability ensures more accurate and contextually relevant responses in natural language understanding, robust recognition systems in vision, and versatile autonomous robotics.

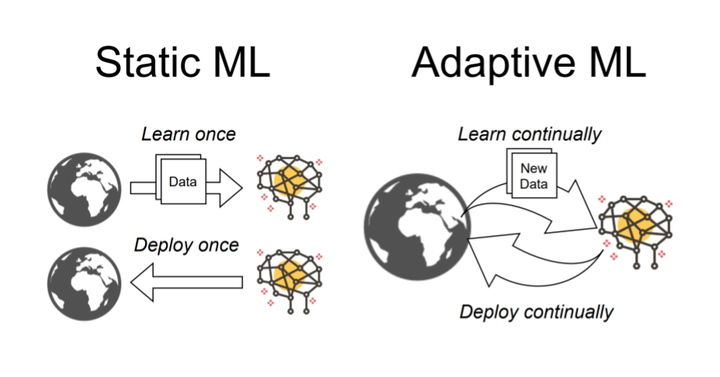
CML is implemented through an open-source machine learning library, facilitating Continuous Integration (CI) and Continuous Delivery (CD) processes. CI automates code integration from multiple contributors, while CD creates deployable applications based on code, data, and models. CML, akin to guiding the growth of children, ensures models autonomously advance with updates from data streams continuously, boosting accuracy and efficiency in AI and ML operations.



* + 1. The role of Continual Learning

Continual learning tackles these issues by enabling machine learning models to adjust and develop in tandem with evolving data and tasks. Instead of commencing anew for each new data stream or task, continual learning models leverage and preserve knowledge from prior experiences. This allows them to amass expertise, confront new challenges, and sustain high performance levels across diverse tasks.

Fundamentally, continual learning establishes the structure for machine learning models to emulate a natural human ability—learning from experience, adapting to novel situations, and retaining accumulated knowledge over their lifetimes.



* + 1. The catastrophic forgetting phenomenon

Catastrophic forgetting is a prevalent issue in machine learning, particularly in lifelong learning scenarios, where neural networks and other learning algorithms tend to lose previously acquired knowledge when exposed to new and unrelated data or tasks.

Key points about catastrophic forgetting include:

1. Task Interference: This phenomenon is most evident when learning multiple tasks sequentially, as updates for current tasks inadvertently disrupt knowledge from previous tasks.
2. Overwriting of Weights: Model updates during training can overwrite previously learned representations, making features for earlier tasks irrelevant.
3. Fixed Capacity: Machine learning models have limited storage capacity, leading to the replacement or competition of new information with existing knowledge.
4. Generalization vs. Specialization Trade-off: Catastrophic forgetting results from a trade-off between adaptability to new tasks and specialization for earlier ones.
5. Importance of Regularization: Techniques like Elastic Weight Consolidation (EWC) and Synaptic Intelligence (SI) are used to counteract forgetting, introducing terms into the loss function that penalize significant changes to crucial model parameters.
6. Memory and Replay: Approaches involving memory buffers or replay mechanisms mitigate forgetting by revisiting past data during training.
7. Incremental Learning: In continual learning scenarios, strategies like fine-tuning and transfer learning help models adapt to new tasks while retaining knowledge from previous ones.

Addressing catastrophic forgetting is crucial for real-world machine learning applications where the learning environment evolves. Researchers actively explore methods to enable models to adapt to new information while retaining past knowledge, a critical step in building intelligent systems for dynamic environments.

* + 1. Techniques help prevent catastrophic forgetting

Replay buffers and regularization techniques are crucial strategies to counteract catastrophic forgetting in machine learning models, especially in continual learning scenarios. These approaches enable models to retain and consolidate knowledge from previous tasks while adapting to new ones.

1. Replay Buffers:

A replay buffer is a memory mechanism storing a subset of past experiences encountered during training. These experiences are periodically replayed or sampled with new data.

Replay buffers offer several benefits:

* Experience Replay: Replaying past experiences helps the model revisit and consolidate knowledge from previous tasks, maintaining a balance between old and new tasks.
* Decorrelation of Data: Replaying past experiences decorrelates data, reducing the risk of overfitting the most recent data and encouraging better generalization across tasks.
* Stabilizing Learning: Replay buffers contribute to stable learning, providing a consistent learning signal, particularly useful with non-stationary data or tasks.
* Priority Sampling: Some implementations prioritize experiences based on significance, aiding in giving more attention to challenging or important tasks.
* Data Augmentation: The replay buffer enhances data diversity, making the model more robust to variations in the data.

Replay buffers find common use in deep reinforcement learning and continual learning, effectively preventing catastrophic forgetting by preserving and reusing past experiences.

1. Regularization Techniques:

Regularization introduces additional terms or constraints in the model’s loss function, penalizing significant changes to important parameters.

Popular techniques include:

* Elastic Weight Consolidation (EWC): Introduces a regularization term to keep model parameters associated with important tasks close to their initial values, protecting critical knowledge.
* Synaptic Intelligence (SI): Similar to EWC but adapts regularization strengths based on parameter importance, proving more effective in continual learning.
* Variational Methods: Treat model parameters as probability distributions, incorporating uncertainty to prevent catastrophic forgetting.
* Dropout: Randomly deactivates neurons during training, introducing noise to prevent overconfidence in predictions and improve adaptability.

Regularization techniques encourage a conservative model update, preserving knowledge from previous tasks and preventing catastrophic forgetting.

* + 1. Types of Continual Learning

Task-based Continual Learning involves a model learning a series of distinct tasks over time, with the objective of adapting to each new task while retaining knowledge from previously learned tasks. Techniques like Elastic Weight Consolidation (EWC) and Progressive Neural Networks (PNN) fall within this category.

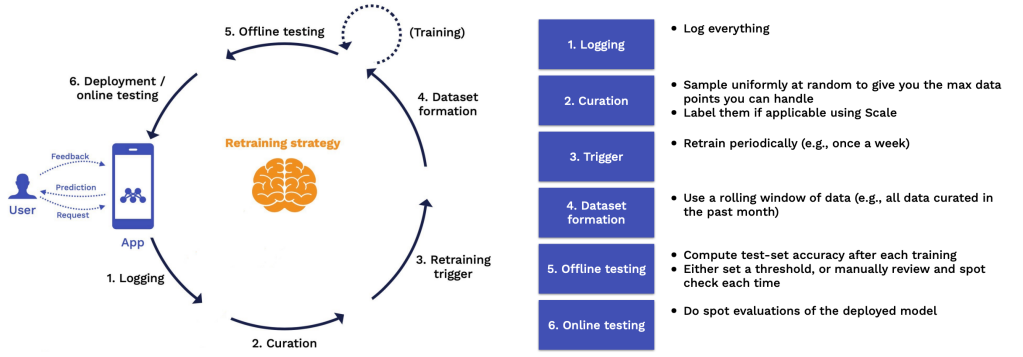
Class-incremental Learning focuses on managing new classes or categories of information over time while retaining knowledge of previously encountered classes. This is common in applications like image recognition, where new object categories are periodically introduced. Methods like iCaRL (Incremental Classifier and Representation Learning) are employed for class-incremental learning.

Domain-incremental Learning deals with adapting to new data distributions or domains. For instance, in autonomous robotics, a robot may need to adapt to different environments. Techniques for domain adaptation and domain-incremental learning are utilized to address this scenario.

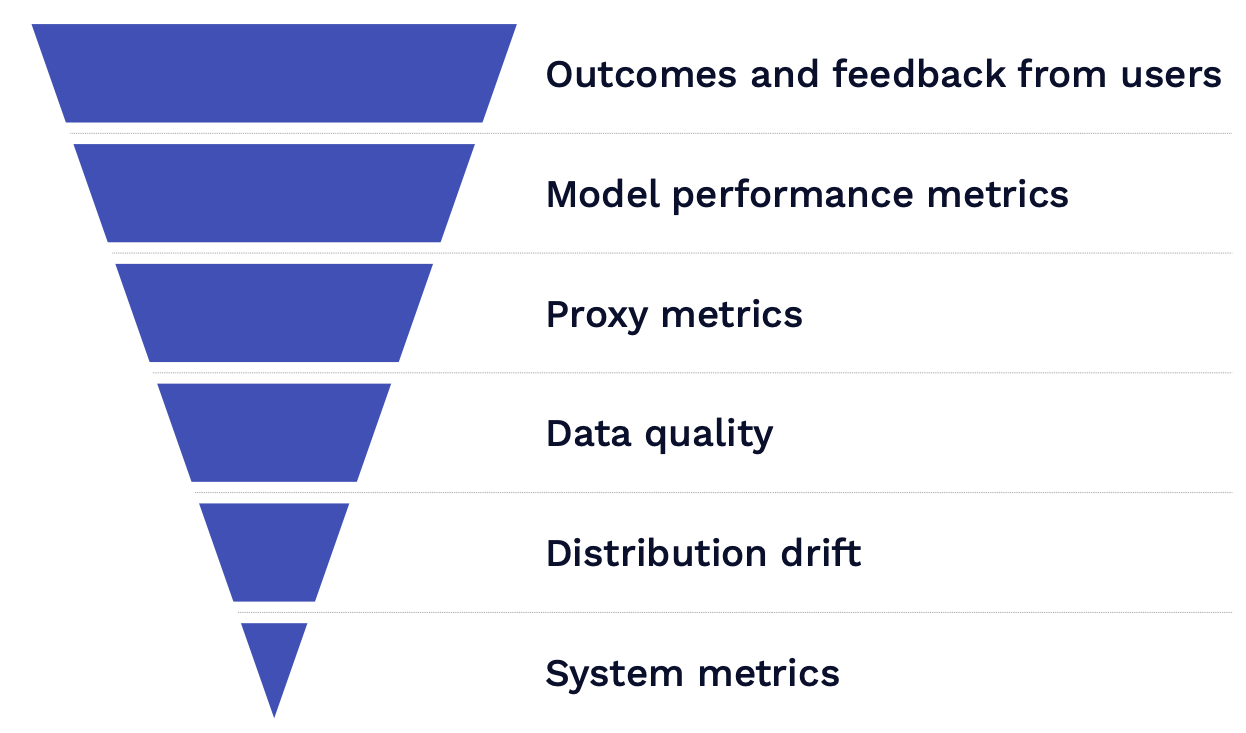
* + 1. Process of Continual Learning

Each stage in this process forwards its output to the subsequent step, with the output determined by a set of rules that collectively constitute our retraining strategy.

1. During the logging stage, the retraining strategy addresses the fundamental question of which data should be stored. By the end of this stage, an "infinite stream" of potentially unlabeled data is generated from production, ready for downstream analysis.
2. Moving to the curation stage, the critical rules to define involve prioritizing data from the infinite stream for labeling and potential retraining. At this stage's conclusion, a reservoir of candidate training points with labels is established, fully prepared for integration into a training process.
3. The retraining trigger stage raises the question of when to initiate retraining, providing a signal to commence a retraining job.
4. Proceeding to the dataset formation stage, rules are set to determine the specific subset of data from the reservoir used for this particular training job. The output is a view into the reservoir or training data, specifying the exact data points for the training process.
5. At the offline testing stage, the crucial rule defines what constitutes "good enough" for all stakeholders, resulting in a report card akin to a "pull request" for the model, signaling approval for the new model to go into production.
6. Finally, during the deployment and online testing stage, the key rule defines the criteria for a successful deployment. The output of this stage is a signal to fully roll out the model to all users.



Different types of metrics ranked in order of how valuable they are



* + 1. Why should ML models be retrained?

Some key reasons for retraining ML models:

1. Regular retraining ensures ML models stay current with the latest data.

2. ML models should undergo periodic retraining; however, it may be costly if there's no concept drift or significant reason, as seen in situations like the mentioned pandemic.

3. Occasionally, ML models may fall below an acceptable threshold, and determining accurate ground truth or data takes time.

4. Data can evolve to be dissimilar from the original training data, emphasizing the importance of involving the team or individual familiar with the initial data input to prevent such discrepancies.

* + 1. Advantages of Continual Learning

Continuous learning proves beneficial for various types of data projects, including descriptive, diagnostic, predictive, and prescriptive scenarios, with particular relevance in situations involving rapidly changing data. Advantages over traditional machine learning approaches encompass:

* Generalization: Continuous learning enhances model robustness and accuracy when confronted with new data.
* Adaptability: Models can evolve over time, making them well-suited for dynamic environments, crucial in fields like autonomous robotics and natural language understanding, ultimately improving long-term predictive capabilities.
* Efficiency: Instead of retraining models entirely with each emergence of new data or tasks, continuous learning allows incremental updates, conserving computational resources and time.
* Knowledge Retention: It addresses the issue of catastrophic forgetting, enabling models to retain knowledge from past tasks or experiences, valuable for long-term memory retention in AI systems.
* Reduced Data Storage: Techniques like generative replay decrease the need to store and manage large historical datasets, making the deployment of continuous learning more feasible in resource-constrained settings.
* Versatility: Continuous learning is applicable across various domains, including natural language processing, computer vision, and recommendation systems, showcasing its versatility in the field of AI.
  + 1. Limitations of Continual Learning

Challenges related to modeling can be alleviated by having a proper methodology in place and through human intervention. Practices such as model versioning, monitoring, and evaluation are key to tracking model performance.

Some challenges and limitations of Continual Learning:

1. Data Collection Challenges: Collecting precise and relevant data for training AI models poses a significant challenge. Public datasets are often unsuitable for applied machine learning applications like continuous machine learning, necessitating in-house collection or third-party purchases.
2. ML Model Maintenance: Ensuring the accuracy of machine learning models requires regular retraining with the most recent and relevant data. Long-term success depends on having the right infrastructure and consistently updated processes.
3. CML Process Monitoring: Establishing a real-time workflow to monitor the machine learning pipeline is essential. Detection mechanisms for issues like malicious or corrupted data and monitoring for anomalies and concept drifts are crucial components.
4. Catastrophic Forgetting: Despite attempts to mitigate it, continual learning models can suffer from catastrophic forgetting, gradually losing performance on past tasks as new ones are learned. Techniques like regularization, replay buffers, and architectural modifications address this challenge.
5. Overfitting to Old Data: Some continual learning methods may overfit to old data, making it challenging for the model to generalize to new tasks or domains.
6. Complexity: Implementing continual learning techniques can be intricate, requiring careful tuning and design, potentially limiting their adoption in certain applications.
7. Scalability: As the model accumulates more knowledge, scalability becomes a challenge, with the model's size and computational requirements potentially growing significantly over time.
8. Data Distribution Shifts: Continual learning models may struggle to adapt effectively when faced with significantly different data distributions in new tasks or domains.
9. Balancing Old and New Knowledge: Striking the right balance between old and new knowledge poses a challenge, as models need to decide how to effectively incorporate both types of information.
   * 1. Applications of Continuous Learning

Autonomous Systems:

* Self-Driving Cars: Autonomous vehicles continuously learn from real-world driving experiences, adjusting to varying road conditions, traffic patterns, and regulatory changes.
* Drones: Drones utilize continual learning to enhance their navigation, obstacle avoidance, and surveillance capabilities, adapting to new environments and challenges.

Natural Language Processing (NLP):

* Chatbots and Virtual Assistants: NLP models continually adjust to evolving language patterns, slang, and user preferences for more accurate responses.
* Translation Services: Continual learning ensures translation models stay current with language changes and idiomatic expressions.

Recommendation Systems:

* Streaming Platforms: Recommendation engines adapt to users’ evolving preferences over time for personalized content suggestions.
* E-commerce: Recommendation systems refine product suggestions based on users’ browsing and purchasing behavior.

Healthcare and Medical Imaging:

* Diagnosis and Disease Detection: Medical imaging models learn to detect new diseases while retaining the ability to identify previously known ailments.
* Drug Discovery: Continual learning aids in predicting the effectiveness and safety of new drugs based on evolving research and data.

Anomaly Detection and Security:

* Network Intrusion Detection: Security systems adapt to emerging attack vectors and new threats, safeguarding against known vulnerabilities.
* Fraud Detection: Continual learning models evolve to recognize novel fraud patterns and tactics used by malicious actors.

Education and Personalization:

* Personalized Learning Platforms: Educational technology platforms adapt content and recommendations based on students’ progress and learning styles.
* Adaptive Testing: Continual learning enables adaptive testing systems to dynamically adjust question difficulty based on students’ performance.

Entertainment and Gaming:

* Gaming: Game environments evolve to keep players engaged, adapting to player preferences and introducing new challenges.
* Content Recommendations: Streaming services adjust recommendations based on user interactions, viewing habits, and emerging trends.

Financial Services:

* Algorithmic Trading: Continual learning models adapt to changing market conditions and evolving trading strategies to optimize portfolio performance.
* Fraud Prevention: Financial institutions use continual learning to detect new types of financial fraud while maintaining accuracy in identifying known fraud patterns.

Environmental Monitoring:

* Climate and Environmental Studies: Environmental monitoring systems adapt to changing conditions, incorporating new data for more accurate predictions and analyses.
* Agriculture: Continual learning aids precision agriculture by adapting to variations in soil, weather, and crop conditions.

Robotics:

* Industrial Robots: Manufacturing robots adapt to new processes and tasks while maintaining efficiency and accuracy.
* Search and Rescue Robots: Continual learning enhances search and rescue robots’ adaptability and problem-solving abilities in complex environments.
  1. Test Production
     1. Introduction

1. Machine Learning in Production - Testing

Traditional software testing and machine learning (ML) tests can be separated by Machine Learning; software tests examine the written logic, whereas ML tests check the obtained logic.

ML tests are further classified as testing and evaluation. We're all familiar with ML assessment, which involves training a model and evaluating its performance on an unknown validation set using metrics (accuracy).

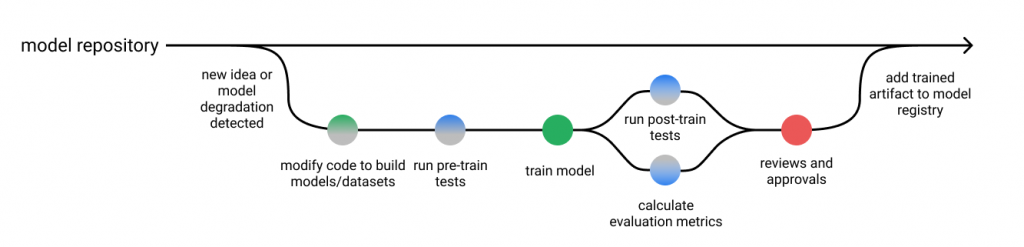
ML testing, moreover, entails checking model behavior. Pre-train tests, which can be performed without learned parameters, determine whether our written logic is correct. Post-train tests determine whether the learnt logic is correct.

How to Test Machine Learning Models

- Unit test. Check the correctness of individual model components.

- Regression test. Check whether your model breaks and test for previously encountered bugs.

- Integration test. Check whether the different components work with each other within your machine learning pipeline.



Machine Learning model monitoring framework.

1. Machine Learning for Predictive Analysis

Machine learning algorithms have transformed the way we approach software testing by enabling predictive analysis, which is a proactive method of identifying possible issues before they escalate in the real environment. Here's how machine learning does it:

* Identifying prospective Issues - Machine learning algorithms evaluate historical data and patterns to identify prospective codebase issues. These algorithms can forecast the possibility of similar issues emerging in the future by finding links between particular code changes and past problems.
* Anomaly Detection - An important part of predictive analysis is recognizing patterns that differ from the norm. Machine learning algorithms excel in detecting anomalies in large datasets, allowing teams to identify oddities and potential problems in real time.

Take, for example, an e-commerce platform. Machine learning algorithms can examine data on user behavior. If a sudden increase in traffic or a rise in specific user behaviors is identified, the system can forecast possible server load, allowing proactive measures to avert crashes or slowdowns.

To summarize, the marriage of AI with software testing is a game changer in the industry. It's not just about writing lines of code; it's about creating solutions that are intelligent, adaptive, and ethical. It is the role of developers and testers to harness the potential of AI and produce software that not only satisfies today's needs but anticipates tomorrow's issues. By investigating AI-powered testing tools and processes, we pave the path for a time when software development is no longer a necessity.

* + 1. Step of Test Production

1. Monitoring for Data Drift:

Regularly inspect incoming data to verify that it adheres to the same distribution as the training data. Abrupt alterations in data distribution can significantly impact the model's effectiveness.

1. Continuous Model Evaluation:

Persistently assess the model's performance using pertinent metrics tailored to your specific problem, such as accuracy, precision, recall, and F1 score. Establish automated testing protocols to systematically evaluate the model's accuracy and overall effectiveness.

1. A/B Testing for Model Comparison:

Deploy multiple iterations of your model through A/B testing in a real-world environment. This methodology facilitates the examination of changes and enhancements, offering insights into their effects on crucial metrics.

1. In-Depth Error Analysis:

Conduct a comprehensive analysis of the errors generated by the model in a production setting. Gain insights into the nature of these errors, identify patterns, and ascertain whether specific conditions contribute to their occurrence.

1. Logging, Monitoring, and Anomaly Detection:

Implement a robust logging mechanism to capture significant events and activities within the production system. This aids in debugging and provides a historical perspective on system behavior. Additionally, set up monitoring tools to promptly detect anomalies, such as sudden fluctuations in model accuracy.

1. Testing Scalability and Resource Handling:

Verify that your system can effectively manage anticipated loads by assessing its scalability. Evaluate how well it scales as the volume of requests or data points increases, ensuring optimal resource allocation.

1. Security Assessments:

Conduct thorough security assessments to pinpoint and mitigate potential vulnerabilities in your machine learning system. Ensure the secure handling of sensitive data by implementing and maintaining essential security measures.

1. Robustness Testing under Diverse Conditions:

Subject your model to robustness testing by introducing noisy or unexpected inputs. Evaluate its performance under various conditions, including edge cases and outliers.

1. Effective Model Versioning:

Establish a systematic versioning system for your models, enabling seamless rollbacks to prior versions in case issues arise with a new deployment. This ensures a reliable and controlled deployment process.

1. Comprehensive Documentation Practices:

Maintain detailed documentation for your models, encompassing crucial details such as training data, model architecture, hyperparameters, and preprocessing steps. This documentation serves as a valuable resource for understanding and replicating model behavior.

1. Iterative Feedback Loop:

Create a dynamic feedback loop with end-users and stakeholders to collect insights and continuously enhance the model based on real-world feedback. This iterative process contributes to ongoing improvements and adaptations to evolving requirements.

1. Adherence to Compliance and Ethical Standards:

Ensure strict adherence to legal and ethical standards in your machine learning system. Address any potential biases or fairness issues that may arise during deployment, prioritizing compliance with regulatory frameworks. Regularly reassess and update practices to align with evolving standards and ethical considerations.

CHAPTER 2. GROUP PROBLEMS

<https://github.com/neit1803/ml_final.git>

REFERENCE

1. [Gradient Descent, Quý Nguyễn](file://C:\\Users\\User\\Downloads\\520H0583_Final_Report.docx" \l ":~:text=Gradient Descent là một thuật,cho việc tối ưu models.).
2. [Multivariable Regression and Gradient Descent. Rajkeshav](https://www.codingninjas.com/studio/library/multivariable-regression-and-gradient-descent).
3. [Gradient Descent, Anirudh Pai.](https://www.machinelearningworks.com/tutorials/gradient-descent)
4. [[Gradient Descent] – Phần 1: “Gradient Descent là gì?”, Khoa Nguyen Van.](http://tutorials.aiclub.cs.uit.edu.vn/index.php/2020/06/07/machine-learning-gradient-descent-la-gi-phan-1-5/)
5. [A Comprehensive Guide on Optimizers in Deep Learning, Ayush Gupta.](file://C:\\Users\\User\\Downloads\\520H0583_Final_Report.docx" \l "Gradient_Descent_Deep_Learning_Optimizer)
6. [Optimizers in Deep Learning, Cathrine Jeeva.](https://www.scaler.com/topics/deep-learning/optimizers-in-deep-learning/)
7. [Various Optimization Algorithms For Training Neural Network, Sanket Doshi.](https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6)
8. [Optimizer- Hiểu sâu về các thuật toán tối ưu ( GD,SGD,Adam,..), Trần Trung Trực.](https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8)
9. [Types of Optimizers in Deep Learning Every AI Engineer Should Know, Pavan Vadapalli.](https://www.upgrad.com/blog/types-of-optimizers-in-deep-learning/)
10. [ĐÁNH GIÁ HIỆU NĂNG CỦA CÁC THUẬT TOÁN TỐI ƯUTRONG MÔ HÌNH HỌC SÂU ĐỐI VỚI BÀI TOÁN PHÂN LỚP HÌNH ẢNH, Khoa Tin học – Trường ĐHSP Huế (Vĩnh Anh Nghiêm Quân – Nguyễn Lê Trung Thành – Nguyễn Thị Lan Anh)](https://csdlkhoahoc.hueuni.edu.vn/data/2021/5/BaiDangHoiThao.pdf)
11. [ĐÁNH GIÁ CÁC THUẬT TOÁN TỐI ƯU ĐỐI VỚI MÔ HÌNH MẠNG NƠ-RONTÍCH CHẬP TRONG TÁC VỤ NHẬN DIỆN HÌNH ẢNH, Trường Đại học Khoa học, Đại học Huế (Vương Quang Phước, Nguyễn Đức Nhật Quang).](https://jos.husc.edu.vn/backup/upload/vol_18/no_1/668_fulltext_4.ĐTVT - Phuoc - Vuong Quang Phuoc.pdf)
12. [Chương 2. Mô hình hoc máy liên quan tới Lifelong Machine Learning, Tram Ho.](https://itzone.com.vn/vi/article/chuong-2-mo-hinh-hoc-may-lien-quan-toi-lifelong-machine-learning/)