**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**REPORT**

**MACHINE LEARNING**

*Instructor*: **PhD.LE ANH CUONG**

*Student*: **BUI HAI DUONG - 521H0220**

*Class*: **21H50302**

**HO CHI MINH CITY, 2023**

**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**REPORT**

**MACHINE LEARNING**

*Instructor*: **PhD.LE ANH CUONG**

*Student*: **BUI HAI DUONG - 521H0220**

*Class*: **21H50302**

**HO CHI MINH CITY, 2023**

ACKNOWLEDGEMENT

To complete this essay, besides our own efforts, we have received a lot of help in terms of knowledge, experience, and skills from the school and teachers. First and foremost, we would like to express my special gratitude to Assoc.Prof.PhD Le Anh Cuong - the lecturer of Introduction to Machine Learning course who has taught us valuable knowledge of the subject. That knowledge is the foundation for us to continue learning and effectively apply it to this essay. Additionally, We would like to thank the teacher for allowing us to complete this essay, which has helped us to further develop my understanding of the subject. Thank you for your guidance and support in helping us to complete this essay to the best of my ability. Moreover, we would also like to express our gratitude to the school and the teachers who have compiled the Introduction to Machine Learning materials, providing me with useful resources for research and essay writing. Thank you sincerely!

**COMPLETION OF THESIS**

**AT TON DUC THANG UNIVERSITY**

We here by certify that this thesis is my/our own work and was conducted under the guidance of PhD.Le Anh Cuong. The research and results presented in this thesis are truthful and have not been published previously in any form. The data presented in tables and figures used for analysis, comments, and evaluations were collected by the author from various sources and are clearly cited in the reference section.

Moreover, this thesis includes some comments, evaluations, and data from other authors and organizations, which are properly cited and referenced.

If any misconduct is detected, I fully take responsibility for the content of my thesis. Ton Duc Thang University is not liable for any copyright infringement that may occur during the thesis completion process.

*Ho Chi Minh City, October 22, 2023*

*Author*

*(signature and full name)*

ACKNOWLEDGEMENT AND EVALUATION SECTION BY INSTRUCTOR

**Instructor's Acknowledgement Section**

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

Ho Chi Minh City, 2023

(signature and full name)

**Instructor's Evaluation Section**

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

Ho Chi Minh City, 2023

(signature and full name)

SUMMARY

This document showing the theory based on the research document of Optimizer and experiment the continual learning with Test Production.

Contents

[ACKNOWLEDGEMENT i](#_Toc154352148)

[ACKNOWLEDGEMENT AND EVALUATION SECTION BY INSTRUCTOR iii](#_Toc154352149)

[SUMMARY iv](#_Toc154352150)

[CHAPTER 1: PERSONAL TASK 1](#_Toc154352151)

[**1.1.** **What is optimizer in machine learning** 1](#_Toc154352152)

[**1.2.** **Types of optimizer** 1](#_Toc154352153)

[**1.2.1.** **Gradient Descent** 1](#_Toc154352154)

[**1.2.1.1.** **How does Gradient Descent work** 1](#_Toc154352155)

[**1.2.1.2.** **Advantages and disadvantages** 1](#_Toc154352156)

[**1.2.2.** **Stochastic Gradient Descent** 1](#_Toc154352157)

[**1.2.2.1.** **How does Gradient Descent work** 1](#_Toc154352158)

[**1.2.2.2.** **Advantages and disadvantages** 1](#_Toc154352159)

[**1.2.3.** **Stochastic with Momentum** 1](#_Toc154352160)

[**1.2.3.1.** **How does Gradient Descent work** 1](#_Toc154352161)

[**1.2.3.2.** **Advantages and disadvantages** 1](#_Toc154352162)

[**1.2.4.** **Mini Batch** 1](#_Toc154352163)

[**1.2.4.1.** **How does Gradient Descent work** 1](#_Toc154352164)

[**1.2.4.2.** **Advantages and disadvantages** 1](#_Toc154352165)

[**1.2.5.** **Adagrad** 1](#_Toc154352166)

[**1.2.5.1.** **How does Gradient Descent work** 1](#_Toc154352167)

[**1.2.5.2.** **Advantages and disadvantages** 1](#_Toc154352168)

[**1.2.6.** **RMS Prop** 1](#_Toc154352169)

[**1.2.6.1.** **How does Gradient Descent work** 1](#_Toc154352170)

[**1.2.6.2.** **Advantages and disadvantages** 1](#_Toc154352171)

[**1.2.7.** **AdaDelta** 1](#_Toc154352172)

[**1.2.7.1.** **How does Gradient Descent work** 1](#_Toc154352173)

[**1.2.7.2.** **Advantages and disadvantages** 1](#_Toc154352174)

[**1.2.8.** **Adam** 1](#_Toc154352175)

[**1.2.8.1.** **How does Gradient Descent work** 1](#_Toc154352176)

[**1.2.8.2.** **Advantages and disadvantages** 1](#_Toc154352177)

[CHAPTER 2: CONTINUAL LEARNING AND TEST IN PRODUCTION 1](#_Toc154352178)

[**2.1.** **What is continual learning** 1](#_Toc154352179)

[**2.2.** **Comparison of Stateless retraining and Stateful training** 1](#_Toc154352180)

[**2.2.1.** **Stateless retraining** 1](#_Toc154352181)

[**2.2.2.** **Stateful training** 1](#_Toc154352182)

[**2.3.** **Challenge of continual learning** 1](#_Toc154352183)

[**2.3.1.** **Fresh data access** 1](#_Toc154352184)

[**2.3.2.** **Evaluation** 1](#_Toc154352185)

[**2.3.3.** **Data scaling** 1](#_Toc154352186)

[**2.3.4.** **Algorithm** 1](#_Toc154352187)

[**2.4.** **How continual learning works** 1](#_Toc154352188)

[**2.4.1.** **Manual, stateless retraining** 1](#_Toc154352189)

[**2.4.2.** **Fix schedule automated stateless retraining** 1](#_Toc154352190)

[**2.4.3.** **Fix schedule automated stateful retraining** 1](#_Toc154352191)

[**2.4.4.** **Continual training** 1](#_Toc154352192)

[**2.5.** **Test in production** 1](#_Toc154352193)

[**REFERENCES** 2](#_Toc154352194)

# CHAPTER 1: PERSONAL TASK

* 1. **What is optimizer in machine learning**

Optimizers in machine learning are the algorithms used to adjust parameters in training section with the aim is to minimize the loss function and maximize the accuracy of the training model. Each optimizer has specific update rules, learning rates, and momentum to find optimal model parameters for improved performance.

* 1. **Types of optimizer**
     1. **Gradient Descent**
        1. **How does Gradient Descent work**

Many machine learning algorithms, including linear regression, logistic regression, neural networks, and support vector machines, utilize gradient descent to optimize their cost functions during training. This iterative algorithm adjusts model parameters to minimize the cost, aiming for improved predictive accuracy or classification performance on the training data. Gradient descent is a fundamental optimization technique widely employed across various machine learning models for convergence and enhanced performance.

Here is the formula of Gradient Descent:

With:

represents the model parameters (weights and biases) that we want to optimize.

is the learning rate, which controls how big of a step we take in the direction of the negative gradient.

is the gradient of the loss function with respect to the parameters evaluated at the current parameter values.

* + - 1. **Advantages and disadvantages**

**Advantages:** Gradient Descent is easy to understand. It can solve the problem of optimizing the model by updating the weight value in each loop of the iteration.

**Disadvantages:** Gradient Descent is a simple algorithm. Therefore, it is dependent on the initial parameters.

* + 1. **Stochastic Gradient Descent**
       1. **How does Gradient Descent work**

Stochastic Gradient Descent (SGD) is a variation of the Gradient Descent algorithm designed to enhance the efficiency of optimizing machine learning models, particularly when working with large datasets. Instead of utilizing the entire dataset in each iteration, SGD randomly selects a single training example or a small batch to calculate the gradient and update model parameters. This introduces randomness into the optimization process, hence the term "stochastic."

Here is the formular of Stochastic Gradient Descent (SGD):

With:

represents the model parameters (weights and biases) that we want to optimize.

is the learning rate, which controls the step size taken during each update.

is the gradient of the loss function J with respect to the parameters θ, evaluated at the current parameter values and using a single randomly selected data point from the training set.

* + - 1. **Advantages and disadvantages**

**Advantages:**

* Fast on Big Data: Processes one data point per update, speeding up optimization for large datasets.
* Escapes Local Minima: Random updates prevent getting stuck in suboptimal solutions compared to traditional methods.
* Lower Memory Footprint: Requires less memory, making it resource-efficient.
* Improved Generalization: Can enhance performance on unseen data compared to other optimizers.

**Disadvantages:**

* Noisy Updates: Fluctuating gradients may hinder convergence, yielding suboptimal solutions.
* Hyperparameter Sensitivity: Requires precise tuning of learning rate and parameters for optimal performance.
* Feature Scaling Impact: Sensitivity to feature scale differences can affect updates, potentially reducing accuracy.
* Local Minima Risk: While less probable, there's a chance of getting stuck in unfavorable solutions.
* Complex Convergence: The stochastic nature makes proving theoretical convergence challenging.
  + 1. **Stochastic with Momentum**
       1. **How does Gradient Descent work**

Here is the of Stochastic with Momentum algorithm step by step:

**Calculate the gradient:** Find the gradient of the loss function with respect to the parameters, using the current mini-batch.

**Update momentum variable:** Combine the current gradient with a fraction of the previous update, creating an "accumulation" of past gradients. This momentum variable acts like a ball rolling downhill, building up speed and smoothing out bumps along the way.

**Update parameters:** Adjust the parameters using the momentum-adjusted gradient, moving towards a minimum of the loss function.

* + - 1. **Advantages and disadvantages**

**Advantages:**

* Faster convergence: Momentum helps accelerate convergence, especially in situations with long, shallow valleys in the loss landscape.
* Reduced oscillations: Momentum smooths out gradient fluctuations, minimizing oscillations and overshooting, and leading to more stable convergence.
* Less sensitive to hyperparameters: Compared to vanilla SGD, SGDM is less sensitive to hyperparameter tuning for the learning rate.
* Easy to implement: The addition of momentum to SGD requires minimal code changes, making it readily accessible.

**Disadvantages:**

* Can be slower than other optimizers: While faster than vanilla SGD, SGDM may still be slower than some advanced adaptive learning rate optimizers like Adam or RMSProp.
* Hyperparameter tuning still necessary: While less sensitive than SGD, choosing optimal values for learning rate and momentum hyperparameters is still crucial for good performance.
* May overshoot in narrow valleys: Although momentum helps reduce oscillations, it can still lead to overshooting the minimum in narrow valleys or noisy situations.
* No adaptive learning rate: Unlike some advanced optimizers, SGDM doesn't automatically adjust learning rates for individual parameters, potentially hindering performance for sparse gradients.
  + 1. **Mini Batch**
       1. **How does Gradient Descent work**

Mini-batch gradient descent updates a model's parameters using the gradient of a small subset of the training set, known as a mini-batch. It calculates the average gradient of the cost function for the mini-batch and updates the parameters in the opposite direction. This approach combines the advantages of both batch and stochastic gradient descent, offering computational efficiency and less noise than stochastic gradient descent. Mini-batch gradient descent is the most used method in practice, striking a balance between efficiency and convergence to a good solution.

Here is the formula of Mini-batch Gradient Descent:

With:

**:** represents the model parameters (weights and biases)

**:** is the learning rate

is the gradient of the loss function J with respect to θ, evaluated using the current mini-batch

: is the t-th mini-batch, a randomly selected subset of data points from the training set

* + - 1. **Advantages and disadvantages**

**Advantages:**

* Works well on large datasets;
* Produce better generalization performance than batch-based gradient descent;
* Its cost function converges faster and more accurately;
* Has a low memory requirement which makes storage requirements easier to manage;
* Stochastic nature allows for fast prototyping and testing of different models and hyperparameters with less computational resources required for training;
* Its ability to escape local minima makes it a powerful tool for solving highly nonlinear problems.

**Disadvantages:**

* Noisy gradients can cause inaccurate parameter updates leading to suboptimal results if not carefully monitored during training.
* If the batch size is too small, then we may suffer from high bias problems due to insufficient exploration of the search space and poor generalization performance.
* If systematic correlations exist within data points, such as in time series prediction tasks, traditional SGD does not take into account this information, leading to worse convergence rates than normal gradient descent algorithms.
  + 1. **Adagrad**
       1. **How does Gradient Descent work**

Stochastic Gradient Descent (SGD) is a variation of the Gradient Descent algorithm designed to enhance the efficiency of optimizing machine learning models, particularly when working with large datasets. Instead of utilizing the entire dataset in each iteration, SGD randomly selects a single training example or a small batch to calculate the gradient and update model parameters. This introduces randomness into the optimization process, hence the term "stochastic."

Here is the formular of Stochastic Gradient Descent (SGD):

With:

represents the model parameters (weights and biases) that we want to optimize.

is the learning rate, which controls the step size taken during each update.

is the gradient of the loss function J with respect to the parameters θ, evaluated at the current parameter values and using a single randomly selected data point from the training set.

* + - 1. **Advantages and disadvantages**

**Advantages:**

* No manual tuning of the learning rate required.
* Faster convergence
* More reliable

**Disadvantages:**

can become large as the number of iterations will increase and due to this will decrease at the larger rate. This will make the old weight almost equal to the new weight which may lead to slow convergence.

* + 1. **RMS Prop**
       1. **How does Gradient Descent work**

RMS Prop (Root Mean Square Propagation) is an adaptive learning rate optimization algorithm commonly used in machine learning, particularly for training deep neural networks.

RMS Prop algorithm:

Calculate the gradient:

Update accumulator variable:

Normalize the gradient:

Update parameters:

With:

(learning rate): Controls the overall step size.

(decay rate): Determines how quickly the influence of past gradients decays (typically between 0.9 and 0.99).

(small constant): Prevents division by zero

* + - 1. **Advantages and disadvantages**

**Advantages:**

* Faster convergence: Compared to vanilla SGD and Adagrad, it often reaches the minimum loss value quicker, especially for deep networks.
* Smoother updates: Normalizing gradients reduces oscillations and overshooting, leading to more stable convergence.
* Adaptive learning rates: Adjusts learning rates for each parameter based on recent gradient history, making it effective for sparse gradients.
* Easy to implement: Requires minimal changes compared to SGD, making it accessible for beginners.

**Disadvantages:**

* Hyperparameter sensitive: Choosing the optimal learning rate and decay rate can be crucial and require experimentation.
* High memory footprint: Accumulating squared gradients adds to memory requirements, potentially impacting efficiency on large datasets.
* Not always the best: While effective in many cases, other optimizers like Adam may sometimes achieve better performance.
* Limited theoretical guarantees: Convergence proofs for RMSProp are less well-developed compared to simpler algorithms like SGD.
  + 1. **AdaDelta**
       1. **How does Gradient Descent work**

Adadelta is an adaptive learning rate optimization algorithm used in machine learning, particularly for training deep neural networks.

Here is the step by step for adam algorithm:

Calculate the gradient:

Update

Compute RMS of previous gradients:

Compute update step:

Update parameters:

Update

Compute RMS of previous updates:

**With:**

Represents the model parameters (weights and biases) that you're optimizing.

: The decay rate, typically between 0.9 and 0.99. It controls how quickly the influence of past gradients and updates decays. Higher values of ρ place more emphasis on recent information, while lower values give more weight to past history.

: A small constant, usually around 1e-8, to prevent division by zero in the calculations.

A vector of the same size as θ, storing a decaying average of the squared gradients for each parameter. It's used to normalize the gradients and stabilize updates.

A vector of the same size as θ, storing a decaying average of the squared updates for each parameter. It contributes to momentum-like behavior and helps smooth out oscillations.

The root mean square of the previous gradients, calculated as It's used to normalize the gradient in the update step, preventing excessively large or small updates.

The root mean square of the previous updates, calculated as It introduces a momentum-like effect, helping to accelerate convergence and avoid overshooting.

: The update step, which determines how much the parameters will be adjusted in each iteration. It's calculated based on the normalized gradient and .

* + - 1. **Advantages and disadvantages**

**Advantages:**

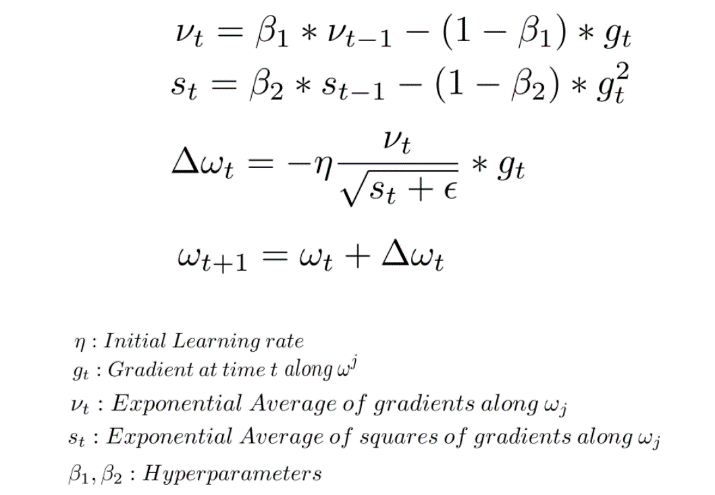
* Less hyperparameter tuning: Eliminates the need for setting a learning rate, potentially simplifying hyperparameter tuning.
* Stable updates: Fixed window size can be beneficial for noisy or sparse gradients.
* Lower memory footprint: Efficient for scenarios with limited memory resources.

**Disadvantages:**

* May be slower to converge: Can lag behind RMSProp in terms of convergence speed, especially for initial learning phases.
* Less flexible control: Fixed window size provides less control over the impact of past gradients compared to RMSProp's decaying average.
  + 1. **Adam**
       1. **How does Gradient Descent work**

Adam (Adaptive Moment Estimation) is a powerful and popular adaptive learning rate optimization algorithm used in machine learning, particularly for training deep neural networks.

Here is the equation of adam:



* + - 1. **Advantages and disadvantages**

**Advantages:**

* Fast convergence properties and its ability to adapt to the local smoothness condition, making it more effective than non-adaptive methods like SGD.
* Performs well with large batch sizes, taking advantage of the reduction in noise.

**Disadvantages:**

* The need for larger and deeper models with more parameters to achieve similar model fits as other optimizers like the LM algorithm.
* Adam may struggle to fit lower amplitude components of a function compared to other optimizers.

# CHAPTER 2: CONTINUAL LEARNING AND TEST IN PRODUCTION

1. **Continual Learning And Test In Production**
   1. **What is continual learning**

Continual learning is a concept to learn a model for many tasks sequentially without forgetting knowledge obtained from the preceding tasks, where the data in the old tasks are not available any more during training new ones.

* 1. **Comparison of Stateless retraining and Stateful training**
     1. **Stateless retraining**

Retrain your model from scratch each time, using randomly initialised weights and fresher data:

* There might be some overlap with data that had be used for training previous model version.
* Most companies start doing continual learning using stateless retraining.
  + 1. **Stateful training**
* Allows your model to update with significantly less data.
* Allows your model to converge faster and use less compute power.
* Some companies have reported 45 reduction in compute power.
* It theoretically makes it possible to avoid storing data altogether once the data has been used for training (and leaving some margin of safety time). This theoretically eliminates data privacy concerns.
* In practice, most companies have a practice of let's-keep-track-of-everything and are reluctant about throwing away data even if it is not needed anymore.
* Every now and then you will need to run stateless retraining with a large amount of data to re-calibrate the model.
* Once your infrastructure is setup correctly, changing from stateless retraining to stateful training becomes a push of a button.
* Model iteration vs data iteration: Stateful training is mostly used to incorporate new data into an existing and fixed model architecture (i.e. data iteration). If you want to change your model's features or architecture, you will need to do a first-pass of stateless retraining.
* There has been some research on how to port weights from one model architecture to a new one (Net2Net knowledge transfer, model surgery). There is little to no adoption on these techniques in industry yet.
  1. **Challenge of continual learning**
     1. **Fresh data access**

Numerous companies extract training data from data warehouses such as Snowflake or BigQuery. Nevertheless, data originating from diverse sources enters the warehouse through various mechanisms and at varying speeds.

To illustrate, segments of warehouse data might be directly sourced from real-time transport (events), while other portions arrive through periodic ETL processes on a daily or weekly basis, transferring data from alternative sources.

A prevalent solution to address this challenge involves extracting data directly from real-time transport for training purposes before it is officially stored in the warehouse. This becomes particularly advantageous when the real-time transport is integrated with a feature store.

The bottleneck in many cases is the speed at which one can assign labels to new data. Tasks that are most suitable for continual learning are those with inherent labels and quick feedback loops – the shorter the loop, the faster the labeling process.

In situations where obtaining natural labels within the required timeframe is challenging, alternative approaches like weak supervision or semi-supervision can be explored, even though they may introduce some level of label noise. As a final option, resorting to recurrent and swift crowdsourcing for label annotation can be considered.

* + 1. **Evaluation**

Embracing continual learning as a practice introduces the potential for severe model failures. The increased frequency of model updates correlates with a higher likelihood of encountering failures. Furthermore, continual learning creates vulnerabilities to coordinated adversarial attacks aimed at corrupting the models. Therefore, thorough testing of models before their broader deployment becomes imperative.

* + 1. **Data scaling**

Calculating features usually involves scaling, a process that demands access to global data statistics such as minimum, maximum, average, and variance.

In the case of employing stateful training, the global statistics need to account for both the historical data used for initial model training and the new data incorporated for updates. Managing global statistics under these circumstances can be challenging.

A widely used approach is to incrementally compute or estimate these statistics as new data is observed, as opposed to loading the entire dataset during training and calculating from that static set.

* + 1. **Algorithm**

This issue arises when utilizing specific algorithms that require rapid updates, such as every hour. These algorithms, inherently dependent on access to the entire dataset for training, include matrix-based, dimensionality reduction-based, and tree-based models. Unlike neural networks or other weight-based models that can be incrementally trained, these models cannot adapt to new data in an incremental fashion.

The challenge becomes prominent when the need for swift updates is critical, and waiting for the algorithm to process the complete dataset is not feasible. While there are some variations of these models designed for incremental training, their adoption remains limited. An example is Hoeffding Trees and their sub-variants.

* 1. **How continual learning works**
     1. **Manual, stateless retraining**

Models are retrained solely when two criteria are fulfilled:

* The model's effectiveness has declined to a point where its outcomes are detrimental rather than beneficial.
* There is available time for the team to carry out the update.
  + 1. **Fix schedule automated stateless retraining**

This phase commonly occurs when the primary models within a domain have been established, shifting the focus from creating new models to maintaining and enhancing existing ones. The challenges of persisting in stage 1 become too significant to overlook.

During this stage, the frequency of retraining is often guided by an intuitive sense rather than a defined schedule.

The transition from stage 1 to stage 2 typically involves the creation of a script that facilitates periodic stateless retraining. The complexity of writing this script varies, ranging from straightforward to intricate, contingent on the coordination of dependencies required for the model's retraining.

* + 1. **Fix schedule automated stateful retraining**

V1 and V2 represent distinct model architectures designed to address the same problem. The notation V1.2 vs V2.3 indicates that the model architecture V1 has undergone its second full stateless retraining, while V2 has undergone its third.

The notation V1.2.12 vs V2.3.43 further details the training history, specifying that there have been 12 stateful trainings for V1.2 and 43 for V2.3.

To comprehensively track model evolution, it is recommended to combine these versioning techniques with others, such as data versioning. Despite the author noting a lack of model stores with such lineage capabilities, companies typically develop their in-house solutions.

Throughout the operational phase, multiple models coexist in production simultaneously, managed through strategies outlined in Testing Models in Production.

* + 1. **Continual training**

**Time-based:**

Retraining is scheduled at predefined time intervals.

**Performance-based:**

Retraining is initiated when the model's performance falls below a specified threshold (e.g., dropping below x%).

Note: Directly measuring accuracy in a production setting may be challenging, necessitating the use of weaker proxy metrics.

**Volume-based:**

Retraining is triggered when the total amount of labeled data increases by a certain percentage (e.g., 5%).

**Drift-based:**

Retraining is prompted by detecting a "major" shift in data distribution.

Note: The challenge with a drift-based trigger is determining when a data distribution shift becomes problematic, as highlighted in the preceding chapter, given that it only poses an issue if it degrades the model's performance. This can be challenging to discern.

* 1. **Test in production**

**Shadow Deployment:**

Deploying the challenger model in parallel with the existing champion model, directing all requests to both models while serving only the champion's predictions. Pros include safety and gathering significant data quickly, but cons involve increased computational cost and limitations in certain cases.

**A/B Testing:**

Deploying the challenger alongside the champion, routing a percentage of traffic to the challenger, and using monitoring for performance comparison. While cost-effective, it requires strong offline evaluation guarantees and a careful traffic split to avoid biased conclusions.

**Canary Release:**

Launching the challenger alongside the champion, gradually transitioning traffic, and closely monitoring performance. This easy-to-implement strategy is cost-effective, but it introduces the risk of less rigor in determining performance differences.

**Interleaving Experiments:**

Incorporating interleaved predictions from both models for a single user to compare user preferences. Netflix found it efficient for identifying the best model with a smaller sample size, but it demands more computational power and is task-specific.

**Bandits:**

Utilizing algorithms that dynamically decide, based on performance, whether to exploit the current best model or explore others. Bandits are data-efficient but challenging to implement, with safety considerations and limitations on use cases.

**REFERENCES**

1. Ayush Gupta, C. (2023). *A Comprehensive Guide on Optimizers in Deep Learning.*

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

1. Trần Trung Trực, C. (2020). *Optimizer - Hiểu sâu về các thuật toán tối ưu ( GD,SGD,Adam,..).*

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

1. amananandrai, C. (2023). *10 famous Machine Learning Optimizers.*

<https://dev.to/amananandrai/10-famous-machine-learning-optimizers-1e22>

1. Huyen, C. (2022, Chapter 9). *Designing machine learning systems: An Iterative Process for Production-Ready Applications*. O’Reilly Media.

<https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md>