VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT

MACHINE LEARNING

(INDIVIDUAL PART)

Instructor: **GV LE ANH CUONG**

Student's name: **VO THANH DANH – 520H0525**

Class: **20H50205**

Course: **23-24**

HO CHI MINH CITY, 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT

MACHINE LEARNING

(INDIVIDUAL PART)

Instructor: **GV LE ANH CUONG**

Student's name: **VO THANH DANH – 520H0525**

Class: **20H50205**

Course: **23-24**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENTS

After half a semester of studying at Ton Duc Thang University, we want to give our sincere thanks to our teachers and Faculty of Information Technology for bringing a desired condition for students able to complete their course by giving many topics that can be applied in real-life.

We want to express our gratitude for our theory lecturer – Le Anh Cuong. He helped us a lot on this course, giving us information about Machine Learning so that we can improve our knowledge. Also, we want to give thanks to each other member in this group. All of us are busy because everyone has other classes, but we still reply to each other when we need to. Once again, we are truly grateful for everyone that helped us to do this project. With challenging work and effort, we have successfully completed this report. Maybe this report has some mistakes because this is our first report on this course, so we really want to take the comments of the teacher and lecturer for our improvement in future projects.

With sincere thanks.

THE REPORT WAS COMPLETED AT TON DUC THANG UNIVERSITY

We assure this is our own project with the instruction of Le Anh Cuong. All the researches, the results in this report are trustworthy and have never been announced in any appearance before. The data in tables for analysis, comments, evaluations were collected by the student in many different sources, which have clearly written in references.

Besides, we used some comments, evaluations, analysis and data of other writer, organizations in the project – which is also in the citations and source notes.

If there is any fraud in my project, we will take full responsibility for our report content. Ton Duc Thang University is not related to the copyright infringement that we made during the implementation process (if available).

Ho Chi Minh City, Oct 23rd, *2023*

Authors

(Full name and signature)

Vo Thanh Danh

EVALUATION OF INSTRUCTING LECTURER

Confirmation of the instructor

Ho Chi Minh City, 2023

(Sign and provide full name)

The assessment of the teacher marked

Ho Chi Minh City, 2023

(Sign and provide full name)

TABLE OF CONTENTS

CHAPTER 1 – OPTIMIZER METHODS IN MACHINE LEARNING

1. INTRODUCTION ABOUT OTIMIZATION

Before delving into the details, it is essential to understand what optimization algorithms are. Fundamentally, an optimization algorithm forms the foundation for constructing neural network models with the goal of "learning" features or patterns from input data. The objective is to find a suitable pair of weights and biases to optimize the model. However, the challenge lies in understanding how this "learning" process occurs. Specifically, how are weights and biases determined? It's not as simple as randomly initializing weights and biases a few times and hoping to find a solution. Clearly, that approach is not feasible and would be a waste of resources. We need an algorithm to iteratively improve weights and biases, and that's why optimization algorithms come into play.

To achieve an effective model for the given problem, model builders often need to focus on adjusting the learning rate during the training process. This tuning can be done manually (as in the case of the SGD optimization algorithm) or automatically (as in optimization algorithms with adaptive learning rates, such as Adam, Adadelta, Adagrad, RMSprop). Therefore, choosing the appropriate optimization algorithm to achieve the best training efficiency becomes a crucial requirement.

2. GRADIENT DESCENT (GD)

One of the most popular algorithms for optimizing neural networks. GD minimizes the loss function J(θ), where θ is the set of model weights to be optimized. The general update rule for GD is as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t)$$

Where $\nabla_\theta J(\theta_t)$ is the gradient of the loss function at θ in step t. η is a positive number known as the learning rate, determining the step size towards the minimum (or local minimum).

There are different variants of GD depending on the amount of data used to compute the gradient of the loss function. Batch Gradient Descent (Batch GD) calculates the gradient over the entire dataset, which can be computationally expensive for large datasets.

Advantages and Disadvantages:

- Advantages
  - Simple and easy to implement.
  - Effective on small datasets, and allows for customizable learning rates.
- Disadvantages
  - Slow on large datasets as it requires computing gradients over the entire dataset.

3. STOCHASTIC GRADIENT DESCENT (SGD)

To overcome the limitation of Batch GD, Stochastic Gradient Descent (SGD) updates weights for each data sample $x^{(i)}$ with corresponding label $y^{(i)}$ as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J\left(\theta_t; x^{(i)}; y^{(i)}\right)$$

This updating scheme makes SGD faster than Batch GD and suitable for online learning when the training dataset is continuously updated.

Advantages and Disadvantages:

- Advantages
  - Faster than Gradient Descent, as it uses only one data sample per update.
  - Suitable for online learning, well-suited for large datasets.
- Disadvantages
  - Unstable due to noise from random data samples.
  - Requires choosing a batch size.

4. ADAGRAD

The Adagrad algorithm was proposed by J.Duchi and colleagues in 2011. In contrast to SGD, the learning rate in Adagrad varies depending on the weights: a low learning rate is applied to weights associated with common features, while a high learning rate is applied to weights associated with rare features. The symbol g_t represents the gradient of the loss function at step t, and $g_{t,i}$ is the partial derivative of the loss function with respect to θ_i at step t.

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

The update rule of Adagrad:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i}$$

Advantages and Disadvantages:

- Advantages
- Adapts to both common and rare features.
- Effective for sparse data with infrequent features.
- Disadvantages
- Learning rate diminishes over time and may become too small, potentially leading to convergence issues.

5. ADADELTA

The Adadelta algorithm was introduced by Zeiler and collaborators in 2012. Adadelta is a variant of Adagrad designed to overcome the issue of diminishing learning rates in Adagrad. Instead of storing all gradients, as in Adagrad, Adadelta restricts the accumulation of gradients within a window of size w that is specified. This way, Adadelta continues learning over multiple update steps. During its execution, rather than storing the squared sum of gradients in the conventional manner, Adadelta accumulates it in the form of the second-order moment:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

Here, the term $E[g^2]_t$ epresents the average of squared gradients at step t, depending on the previous average $E[g^2]_{t-1}$ at step t −1 and the current gradient g_t at step t . The coefficient γ t is typically set to 0.9, indicating that the current gradient at time t heavily depends on the gradients from previous steps.

Advantages and Disadvantages:

- Advantages
- Addresses the diminishing learning rate issue in Adagrad.
- No need to manually set a learning rate.
- Disadvantages
- Requires tuning different parameters (e.g., window size).

6. ADAM (ADAPTIVE MOMENT ESTIMATION)

As an algorithm enabling adaptive learning rates for each weight, Adam not only stores the squared average of past gradients like Adadelta but also maintains the average moment m_t. The values m_t and v_t are calculated by the formulas:

$$m_t = \beta_1 m_{t-1} + (1- \beta_1)g_t$$

$$v_t = \beta_2 v_{t-1} + (1- \beta_2)g_t^2$$

where β_1 and β_2 are non-negative coefficients typically chosen as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. If initialized with vectors of zeros, these values tend to be biased towards zero, especially when β_1 and β_2 are close to. Therefore, to address this, the values are estimated as:

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Subsequently, the weights are updated using the formula:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \widehat{m}_t \text{ } \in \text{ thường bằng } 10^{-8}$$

where η is the learning rate, ϵ is typically set to 10^{-8} and t denotes the time step.

Advantages and Disadvantages:

- Advantages
- Integrates adaptive learning rates, effective across various problem types and datasets.
- Easy to use with default parameter values.
- Disadvantages
- Requires careful consideration of hyperparameters (e.g. β1, β2).

- May be prone to overfitting.

7. RMSPROP

RMSprop addresses the diminishing learning rate issue of Adagrad by scaling the learning rate with the average of the square of the gradient.

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

Advantages and Disadvantages:

- Advantages
  - Effective in reducing learning rates for commonly occurring features.
  - Requires fewer hyperparameter adjustments.
- Disadvantages
  - Not well-adapted to rare features.

ALGORITHM	PROS	CONS
Gradient Descent	Simple, effective on small datasets	Slow on large datasets, requires choosing learning rate
SGD	Fast, suitable for online learning	Unstable, requires choosing batch size
Adagrad	Adapts to common and rare features	Diminishing learning rate, may lead to convergence issues
Adadelta	Addresses diminishing learning rate issue	Requires tuning various parameters
Adam	Integrates adaptive learning rates	Requires careful consideration of

		hyperparameters, may overfit
RMSprop	Effective in reducing learning rates	Not well-adapted to rare features

CHAPTER 2 - CONTINUAL LEARNING AND TEST PRODUCTION

1. CONTINUAL LEARNING

Continuous learning, also referred to as continual machine learning (CML), involves the ongoing process of a model adapting and learning from incoming data streams without requiring a full retraining.

Unlike traditional methodologies where models are trained on a fixed dataset, deployed, and periodically retrained, continuous learning models continuously adjust their parameters to accommodate shifts in data distributions.

In this iterative approach, the model enhances its performance by assimilating insights from the latest data, thereby updating its knowledge base as new information becomes accessible. The life cycle of continuous learning models allows them to sustain relevance over time due to their inherently dynamic nature.

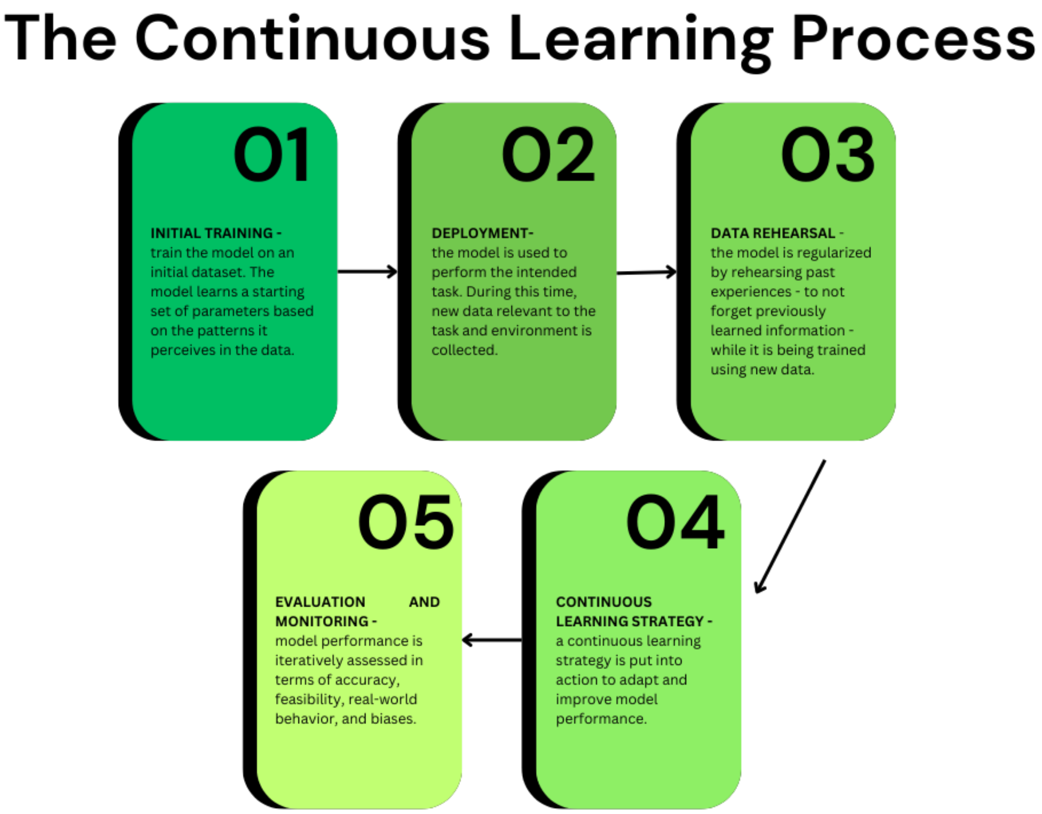
Several methods exist for continuous machine learning modeling, with commonly used strategies encompassing incremental learning, transfer learning, and lifelong learning. Continual Learning strategies include:

1. Regularization: Utilizing techniques such as Elastic Weight Consolidation (EWC) to ensure that the crucial weights of the model remain relatively unchanged when learning from new data.
2. Memory-Augmented Networks: Employing models with the ability to retain important information from previous tasks and utilizing it when addressing new tasks.
3. Dynamic Architectures: Constructing models that can adjust their structure based on the current data and tasks.

The Continuous Learning Process:

Continuous learning represents a progression from conventional machine learning modeling, and consequently, it incorporates many of the identical modeling principles: ***pre-processing, model selection, hyperparameter tuning, training, deployment, and monitoring.***

In the continuous learning framework, two supplementary stages become essential: data rehearsal and the incorporation of a continuous learning strategy. These stages are crucial to guarantee that the model effectively learns from ongoing streams of new data, taking into account the specific application and context of the data task.


The Continuous Learning Process

01

INITIAL TRAINING - train the model on an initial dataset. The model learns a starting set of parameters based on the patterns it perceives in the data.

02

DEPLOYMENT- the model is used to perform the intended task. During this time, new data relevant to the task and environment is collected.

03

DATA REHEARSAL - the model is regularized by rehearsing past experiences - to not forget previously learned information - while it is being trained using new data.

04

CONTINUOUS LEARNING STRATEGY - a continuous learning strategy is put into action to adapt and improve model performance.

05

EVALUATION AND MONITORING - model performance is iteratively assessed in terms of accuracy, feasibility, real-world behavior, and biases.

Advantages of Continuous Learning:

1. **Generalization:** Continuous learning empowers the model to be more robust and accurate in the face of new data, contributing to improved generalization.

2. **Retention of Information:** By employing a continuous learning strategy, the model considers previous knowledge gained in past iterations, enabling it to accumulate information over time.
3. **Adaptability:** A model employing continuous learning adapts to new knowledge – such as concept drift and new trends – thereby having greater predictive capabilities in the long run.

Limitations of Continuous Learning:

1. **Cost:** Continuous learning approaches, while effective, also tend to be more computationally complex than traditional ones as the model needs to consistently adapt to new data. Said complexity often translates into higher economic costs because it necessitates more data, human, and computing resources.
2. **Model Management:** Every time a model's parameters update based on new data, a new model is formed. Therefore, a continuous learning approach may generate a large number of models, complicating the identification of best-performing ones.
3. **Data Drift:** For a continuous learning approach to be worthwhile, we must process a large volume of new data. However, such a model risks the chance of losing predictive capabilities if the feature distribution changes abruptly. Learn more about data drift in a separate article.

2. TEST PRODUCTION

1. Explanation of Test Production

Test Production is a process of testing a machine learning model in a production environment without affecting actual operations. This process aims to ensure that the model functions correctly and performs well when deployed for end users. In Test Production, the model runs in parallel with the real system, but the input data and results are not applied in reality.

2. Problems with Testing Machine Learning Models

Software developers create code to achieve deterministic behavior, allowing for clear identification of failures and a relatively coherent coverage measure, such as lines of code covered. Testing serves two main purposes:

- Ensuring software functions in accordance with specified requirements.
- Identifying defects during development and in production.

On the flip side, data scientists and ML engineers grapple with testing challenges unique to ML models:

- **Lack of Transparency:** Many ML models operate as black boxes, lacking clear internal workings.
- **Indeterminate Modeling Outcomes:** Stochastic algorithms lead to unpredictable model outcomes, making replication challenging.
- **Generalizability:** Models must consistently perform beyond their training environment.
- **Unclear Testing Coverage:** No established method for expressing testing coverage in ML; it differs from traditional code-based metrics and may involve aspects like input data and model output distribution.

- **Resource Intensiveness:** Continuous testing of ML models demands significant resources and time.

3. Evaluation and Testing

- **ML Model Evaluation:**

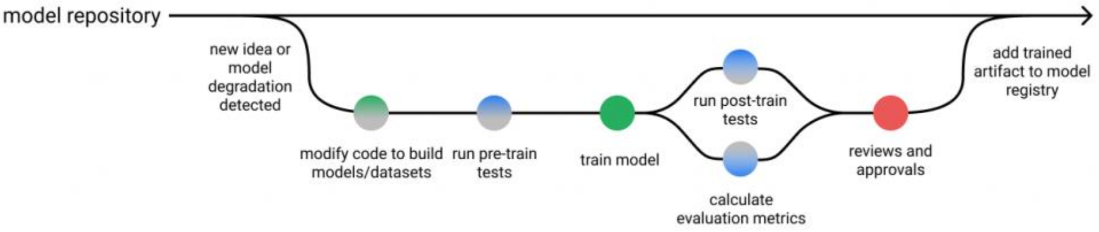
Evaluating machine learning models focuses on their overall performance. This assessment includes performance metrics, curves, and instances of incorrect predictions. It serves as an excellent method for tracking a model's outcomes across different versions. However, it's important to note that model evaluation doesn't provide extensive insights into the reasons behind failures and specific model behaviors.

- **Machine Learning Testing:**

On the contrary, machine learning testing goes beyond assessing model performance on data subsets. It ensures that the composite components of the ML system effectively work together to achieve the desired level of quality results. One could say that it helps teams pinpoint flaws in the code, data, and model so that necessary corrections can be made.

4. Test Machine Learning Models in Practice

Testing tasks can fall into various categories, such as model evaluation, monitoring, and validation, based on the specific problem, circumstances, and organizational structure. This article specifically addresses tests related to post-training in the context of Machine Learning modeling, excluding coverage of other types of tests.


model repository
new idea or model degradation detected
modify code to build models/datasets
run pre-train tests
train model
run post-train tests
calculate evaluation metrics
reviews and approvals
add trained artifact to model registry

- **Testing Trained Models**

While manual test cases work well for code, they are not optimal for Machine Learning models due to the challenge of addressing all edge cases in a multi-dimensional input space.

Instead, assess model performance through monitoring, data slicing, or property-based testing tailored to real-world issues. Augment these approaches with post-training tests that specifically scrutinize the internal behavior of your trained models:

- ***Invariance tests:*** method used to identify input changes that are expected to have no impact on the outputs of a model. This test is often applied through data augmentation, where modified and unmodified input examples are paired to observe the model's response.

- ***Directional expectation tests:*** a method used to assess how changes in the input distribution are expected to influence the output of a model. In practical terms, it involves testing assumptions about specific input features and their anticipated impact on the model's predictions.

- ***Minimum functionality tests:*** is designed to assess whether individual components of a model behave as expected. The rationale behind these tests is that relying solely on overall output-based performance may mask critical underlying issues in the model.

- **Test Model Skills**

Traditional software development tests primarily center on the project's code, which may not seamlessly apply to machine learning (ML) workflows. This is due to ML involving elements beyond code, and model behavior not aligning straightforwardly with pieces of code.

A more effective and 'behavioral' approach to ML testing involves focusing on the expected "skills" of the model, as proposed in a paper on testing NLP models. For instance, evaluating whether a natural language model captures information about vocabulary, names, and arguments, or ensuring that a time series model recognizes trends, seasonal patterns, and change points.

This approach emphasizes testing these skills programmatically by examining key model properties such as invariance, directional expectation, and minimum functionality.

- **Test Performance**

Testing a complete model, especially with integration tests, can be a time-consuming process. To enhance efficiency and conserve resources, it is advisable to focus on testing small components of the model, such as assessing if a single iteration of gradient descent leads to a decrease in loss.

Additionally, utilizing a limited amount of data for testing purposes can expedite the process. Simpler models can be employed to detect shifts in feature importance and proactively identify concept drift and data drift.

This approach aims to streamline the testing workflow while effectively managing the associated time and resource

5. Conclusion

In conclusion, the construction of a robust machine learning solution for a specific problem demands a harmonious blend of Test Production. Test Production plays a pivotal role in guaranteeing the stability and effectiveness of the model in a real-world setting, despite the challenges posed by the model's black-box nature.

REFERENCE

English

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer

[2] RosenblumCS142 Lecture Notes - Mendel Rosenblum, accessed on 10th May 2022 at [www.stanford.edu]

- **[3]** https://www.linkedin.com/pulse/best-practices-ml-model-testing-kolena-ai/
- **[4]** https://paperswithcode.com/task/continual-learning
- **[5]** https://deepchecks.com/how-to-test-machine-learning-models/
- **[6]** https://www.datacamp.com/blog/what-is-continuous-learning