**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**Introduction to**

**machine learning**

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

*Student:* **Le Tran Nhat Quang - 521H0413**

*Course* : **503044**

*Year* : **25**

**HO CHI MINH CITY, 2023**

Vietnam General Confederation of Labor

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**Introduction to**

**machine learning**

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

*Student:* **Le Tran Nhat Quang - 521H0413**

*Course* : **503044**

*Year* : **25**

**HO CHI MINH CITY, 2023**

ACKNOWLEDGEMENT

The first sincere thanks i want to give to Mr Le Anh Cuong, who enthusiastically taught and worked tirelessly to give me enough skills to complete this report. He played an important role in improving our mathematical logic and knowledge. The second thanks i would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving me the opportunity to do this report. Nevertheless, my report will have some errors, i am open to receive any feedbacks from teachers so that i can improve my report.

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

*Ho Chi Minh city, 23rd December, 2023*

*Author*

*(Sign and write full name)*

**THIS PROJECT WAS COMPLETED AT**

**TON DUC THANG UNIVERSIY**

I fully declare that this is my own project and is guided by Mr. Le Anh Cuong. The research contents and results in this topic are honest and have not been published in any form before.

**Should any frauds be found, i will take full responsibility for the content of my report.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh city, 23rd December, 2023*

*Author*

*(Sign and write full name)*

CONFIRMATION AND ASSESSMENT SECTION

**Instructor confirmation section**

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

*Ho Chi Minh December, 2023*

*(Sign and write full name)*

**Evaluation section for grading instructor**

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

*Ho Chi Minh December 2023*

*(Sign and write full name)*

SUMMARY

This report aims to understand basic optimizer methods in machine learning training, Continuous Learning and Test Production when building a machine learning solution to solve a certain problem.

INDEX

[ACKNOWLEDGEMENT i](#_Toc154242103)

[CONFIRMATION AND ASSESSMENT SECTION iii](#_Toc154242104)

[SUMMARY iv](#_Toc154242105)

[INDEX 1](#_Toc154242106)

[1. Optimizer methods 2](#_Toc154242107)

[1.1. Overview: 2](#_Toc154242108)

[**1.2. Gradient Descent:** 2](#_Toc154242109)

[**1.2.1. Gradient Descent of a single variable:** 3](#_Toc154242110)

[**1.2.2. Gradient Descent of two or more variables:** 5](#_Toc154242111)

[**1.3. Stochastic Gradient Descent:** 7](#_Toc154242112)

[**1.4. Comparision between optimier methods:** 8](#_Toc154242113)

[2. Continual Learning and Test Production 12](#_Toc154242114)

[2.1. Continual Learning: 12](#_Toc154242115)

[**2.2. Test Production:** 16](#_Toc154242116)

[3. Complete percentages for each task. 20](#_Toc154242117)

[4. REFERENCES 20](#_Toc154242118)

1. Optimizer methods

1.1. Overview:

Introduction: In machine learning, the training process of the model requires adjusting the input parameters or weights to minimize the difference between the predicted and the actual outputs. This process is achieved through the use of optimizer methods.

Approach:

* We often have to find the largest or smallest value of a number. Solving the derivative equation equal to zero is very complicated or can take infinite solutions. Instead, people often try to find local minimum points, and to some extent, consider it a necessary root.
* The most common approach to solving optimization problems is to start from a point considered close to the root, then use a mathematical operation to gradually approach the desired point, that is, until the derivative close to zero.

**1.2. Gradient Descent:**

Gradient Descent is the most fundamental method, which updates the parameters by moving in the direction opposite to the gradient of the loss function

*x*

*f*

0

(

*x*

)

*f*

(

*x*

)

−

∞

1

+

∞

0

−

+

+

∞

+

∞

-2

*f*

(

*x*

)=

1

2

(

*x*

−

1)

2

−

2

*x*

∗

**Picture 1.2.1.** Describes the variation of a function

**1.2.1. Gradient Descent of a single variable:**

Consider functions of one variable f : R → R. Return to **Picture 1.2.1.** Suppose is the point found after the **t** iteration. We need to find an algorithm to bring as close to as possible. There are two observations as follows:

* To make the next point closer to , we need to move the opposite sign of the derivative:

: a quantity with opposite sign to the **derivative**

* is far from to the right, is greater than 0 (and vice versa). So, the amount of movement is proportional to − .

a learning rate, **gradient step**

: Partial derivatives of Loss function to

* The minus sign (-) shows that we have to go against the derivative so that is the reason this method called gradient descent – descent means going backwards.

**1.2.1.1. Steps to do Gradient Descent Algorithm:**

1. Initialize the parameters of the model randomly ()
2. Compute the gradient of the Loss function () with respect to each parameter.
3. Update the value of parameters () by Gradient Descent.
4. Repeat steps 2 and 3 iteratively to get the best parameter for the defined model.

**1.2.1.2.** **Loss function linear regression (MSE):**

: predicted value

: true value

=> Aim: Find min loss function

**1.2.1.3.** **Gradient descent works:**

* By moving downward toward the pits or valleys in the graph to find the minimum value. This is achieved by taking the derivative of the cost function.
* During each iteration, gradient descent step-downs the loss function in the direction of the **steepest** **descent**. By adjusting the parameters in this direction, it seeks to reach the **minimum of the loss function** and find the best-fit values for the parameters.
* The size of each step is determined by parameter called **Learning Rate**.

**1.2.1.4.** **Choosing learning rate:**

The choice of learning rate is very important. This depends a lot on each problem and some experiments must be done to choose the best value.

Additionally, depending on some problems, GD can work more effectively by choosing an appropriate learning rate or choosing a different learning rate in each round, usually decreasing.

|  |  |
| --- | --- |
| Choose  **Learning Rate** to be very large. It fails to converge: |  |
| Choose  **Learning Rate** to be very small, it will take a longer time to reach minima. |  |

**1.2.2. Gradient Descent of two or more variables:**

Suppose we need to find the global minimum for the function

: the set of parameters that need to be optimized.

: The derivative of that function at any point θ

Similar to a **Gradient Descent of a single variable**, this also starts with a prediction point θ. After t iteration:

**1.2.2.1.** **Loss function linear regression (MSE):**

: predicted value

: true value

=> Aim: Find min loss function

**1.2.2.2.** **Compared to Gradient Descent of a single variable:**

|  |  |  |
| --- | --- | --- |
|  | **GD of single variable** | **GD of multiple variables** |
| **Algorithm** | - calculate the **derivative of the loss function** with respect to the **single variable**.  - update the variable accordingly. | - computing the **gradient vector**, which consists of **partial derivatives of the loss function.**  - updating all variables simultaneously. |
| **Convergence Properties** | relatively straightforward since there is **only one variable to optimize**:  - converges to a local minimum or, in convex cases, the global minimum | can be **more** **complex** due to the **increased number of variables**:  - converges to a local minimum, saddle point, or even fail to converge in some cases |
| **Computational Complexity** | **less demanding:**  - computing the derivative  - updating a single variable | **computationally more intensive**:  - computing the gradient vector  - updating multiple variables |
| **Applications** | used in **simple regression models** or optimization **problems with only one variable** to optimize. | used in more **complex machine learning models** like **neural networks.** |
| **Challenges** | avoiding local minima and ensuring convergence to the global minimum | vanishing or exploding gradients, and the curse of dimensionality |

**1.3. Stochastic Gradient Descent:**

**Introduction:**

* Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent. It addresses the computational inefficiency of Gradient Descent when dealing with large datasets in machine learning.
* Instead of using the entire dataset for each iteration, it tries to lower the computation per iteration by selecting only a single random training example (small batch) to calculate the gradient and update the model parameters.
* This random selection introduces randomness - stochastic” in stochastic Gradient Descent.

**1.3.1.** **Gradient descent works:**

1. Randomly assigning values to the parameters of the model.
2. Set the iterations and the learning rate.
3. Repeat the following steps until we reach the desired results or complete the maximum number of iterations:
4. Shuffle the training data to introduce randomness.
5. Go through each training example (or a small group of examples) in the shuffled order.
6. Calculate the slope (gradient) of the cost function with respect to the model parameters using the current example (or group of examples).
7. Adjust the model parameters by taking a step in the opposite direction of the gradient, scaled by the learning rate.
8. Check if the model has converged by comparing the cost function values between iterations.
9. Once we have reached the convergence criteria or completed the maximum number of iterations, provide the optimized values for the model parameters.

**1.3.2.** **Advantages and Disadvantages of SGD:**

|  |  |
| --- | --- |
| **Advantage** | **Disadvantage** |
| SGD is faster than other variants of Gradient Descent since it uses only one example to update the parameters. | - The updates in SGD are noisy and have a high variance.  - Make the optimization process less stable and lead to oscillations around the minimum. |
| Memory-efficient and can handle large datasets that cannot fit into memory. | - Require more iterations to converge to the minimum. |
| Due to the noisy updates in SGD, it has the ability to escape from local minima and converges to a global minimum. | - Due to the noisy updates, SGD may not converge to the exact global minimum and can result in a suboptimal solution.  - This can be mitigated by using techniques such as learning rate scheduling and momentum-based updates. |

**1.4. Comparision between optimier methods:**

**1.4.1.** **Similarity:**

* Objective: Both GD and SGD aim to optimize the parameters of a model by minimizing a cost function.
* Parameter Updates: Both algorithms update the model parameters iteratively based on the calculated gradients.
* Learning Rate: Both GD and SGD incorporate a learning rate that determines the step size in each parameter update.
* Convergence Criteria: Both algorithms can use similar convergence criteria, such as monitoring the change in the cost function or the gradient magnitude, to determine when to stop the optimization process.

**1.4.2.** **Difference:**

|  |  |  |
| --- | --- | --- |
|  | **Gradien Descent** | **Stochastic Gradient Descent** |
| **Calculation of Gradients** | - Calculates the gradients using the entire training dataset.  - It computes the average gradient across all examples, resulting in a more accurate estimation but at a higher computational cost. | - Approximates the gradients using a single example or a small subset (mini-batch) of examples.  - This enables faster computations but introduces more noise due to the limited sample size. |
| **Update Frequency** | - Updates the model parameters once per iteration, considering the gradients of all training examples.  - This means fewer parameter updates but each update is based on more comprehensive information. | - Updates the parameters more frequently, as it processes individual examples or mini-batches.  This leads to more frequent updates but with less comprehensive information. |
| **Convergence Speed** | - Takes longer to converge, especially when dealing with large datasets or complex models.  - It considers the entire dataset for each parameter update. | - Converges faster in terms of the number of iterations, as it benefits from more frequent updates.  - However, each update only considers a fraction of the training examples, which may introduce more noise and lead to slower convergence in terms of actual computational time. |
| **Noise** | - Less sensitive to noisy data as it smooths out the noise by considering the entire dataset for gradient computation. | - More sensitive to noisy data due to its reliance on individual examples or mini-batches, which may introduce more variance. |
| **Memory Usage** | - Requires storing the entire dataset in memory to calculate gradients, making it memory-intensive for large datasets. | - Requires memory for storing only a small subset of examples or mini-batches, making it more memory-efficient for large datasets. |

**1.4.3.** **Improvements:**

**1.4.3.1.** **Computational Efficiency:**

* Batch gradient descent can be used as a compromise between GD and SGD.
* Batch gradient descent computes gradients using a fixed-size subset of the training data, striking a balance between accuracy and computational cost.

**1.4.3.2.** **Convergence Speed:**

* Adaptive learning rate techniques such as learning rate schedules or momentum can be employed.
* These techniques adjust the learning rate or introduce momentum to accelerate convergence.

**1.4.3.3.** **Robustness to Noise:**

* Techniques such as mini-batch SGD or stochastic gradient with momentum can be utilized.
* These methods introduce additional stability by considering a small subset of examples or incorporating momentum to smooth out the noise.

**1.4.3.4.** **Memory Usage:**

* Techniques such as mini-batch GD or parallel computing can be employed.
* Mini-batch GD computes gradients using smaller subsets of the data, reducing memory requirements.
* Parallel computing distributes the computation across multiple machines or processors, further reducing memory usage

**1.4.3.5.** **Finding the Optimal Learning Rate:**

* Techniques such as learning rate decay, adaptive learning rates (e.g., AdaGrad, RMSprop, Adam), or line search methods can be used to automatically adjust the learning rate during the optimization process, improving convergence speed and stability for both GD and SGD.
* By employing optimization techniques and strategies specific to each algorithm, their performance can be improved in terms of convergence speed, accuracy, stability, and memory efficiency.

2. Continual Learning and Test Production

2.1. Continual Learning:

Introduction:

Continal learning is Learning continuous sequences of tasks.

3 popular scripts:

* Task-based incremental learning (TIL):
  + TIL focuses on sequentially learning multiple tasks over time. Each task is treated as a separate learning problem, and the model is trained on one task at a time.
  + After training on a task, the model's parameters are typically frozen or partially preserved to prevent catastrophic forgetting of previously learned tasks.
  + TIL aims to leverage knowledge transfer from previously learned tasks to accelerate learning on new tasks.
* Domain incremental learning (DIL):
  + DIL focuses on learning from new data that belongs to different domains or distributions over time.
  + The model is incrementally updated with new data from the new domain, while trying to retain the knowledge learned from previous domains.
  + DIL aims to adapt the model to new domains without losing performance on previously learned domains.
* Grade-based incremental learning (CIL):
  + CIL refers to learning incrementally with increasing difficulty or complexity.
  + The model is trained on a series of tasks or examples with varying levels of difficulty, gradually increasing the complexity as the learning progresses.
  + CIL aims to build upon previously learned concepts and gradually improve the model's performance on more challenging examples or tasks.

Online Continuous Learning: Continual Learning Research in online setup (data arrives continuously, distributed by data changes frequently)

2.1.1. Evaluation Metrics:

* Use the hold-out test set for each task in T tasks.
* Train the tasks one by one and evaluate on the hold-out test set of the tasks:
* Build a matrix where is the model's accuracy on task when completing task .
* 3 Popular Measurements:
  + Average Accuracy (ACC):
    - Average Accuracy is a commonly used metric to evaluate incremental learning algorithms.
    - It measures the average performance of the model on all tasks encountered during the incremental learning process.
    - ACC provides an overall assessment of the model's ability to retain knowledge from previous tasks while learning new tasks.
    - Higher ACC values indicate better performance, indicating that the model successfully retains previously learned knowledge while adapting to new information.
  + Backward Transfer (BWT):
    - Backward Transfer evaluates the influence of learning new tasks on the performance of previously learned tasks.
    - It measures the change in performance on earlier tasks after training on new tasks.
    - BWT can be calculated by comparing the performance of the model on the previously learned tasks before and after training on the new tasks.
    - Positive BWT values indicate that learning new tasks has positively impacted the performance on earlier tasks, indicating knowledge transfer and retention.
  + Forward Transfer (FWT):
    - Forward Transfer measures the influence of previously learned tasks on the performance of new tasks.
    - It evaluates whether knowledge gained from earlier tasks improves the learning of new tasks.
    - FWT can be calculated by comparing the performance of the model on new tasks before and after training on previous tasks.
    - Positive FWT values indicate that knowledge from earlier tasks has positively impacted the performance on new tasks, indicating transfer and generalization of knowledge.

2.1.1. Common Approaches:

* + Prior/Regularization-based:
    - This approach incorporates prior knowledge or regularization techniques to mitigate catastrophic forgetting and retain previously learned information.
    - Regularization methods, such as Elastic Weight Consolidation (EWC) or Synaptic Intelligence (SI), assign importance or penalty to certain parameters to preserve their significance during training on new tasks.
    - By constraining the model's parameters based on the previously learned knowledge, this approach aims to prevent drastic changes that could lead to forgetting.
  + Memory-based:
    - The memory-based approach focuses on storing and replaying past experiences to facilitate the learning of new information.
    - Experiences or data samples from previous tasks are stored in a memory buffer or replay buffer.
    - During the training of new tasks, a combination of current and past experiences is used, either by sampling from the memory buffer or by generating synthetic examples based on the stored experiences.
    - The idea is to use the stored memories to provide a diverse and balanced training set, aiding in retaining knowledge from previous tasks.
  + Architecture-based:
    - The architecture-based approach involves adapting the model's architecture or structure to accommodate new tasks or complexities over time.
    - This approach can include methods such as network expansion, where the model adds new neurons, layers, or modules to accommodate new tasks or information.
    - Other techniques involve using dynamic architectures, such as Progressive Neural Networks (PNNs) or Dynamic Neural Networks (DNNs), which allow the model to expand or modify its structure based on the encountered tasks or data

**2.2. Test Production:**

**Introduction**:"test production" is not a commonly used term. However, in the machine learning workflow, there is a step known as "testing" or "evaluation" that involves assessing the performance and generalization capabilities of a trained model on unseen data.

**During the testing phase in machine learning, the following steps are typically followed**:

**2.2.1. Test Dataset**: A separate dataset, distinct from the training dataset, is used for testing. This dataset should represent real-world data that the model is expected to perform well on.

* + **Cross-Validation**: In addition to a separate test dataset, cross-validation is often used to assess the model's performance.
    - Cross-validation involves dividing the training dataset into multiple subsets or "folds" and performing multiple rounds of training and testing.
    - This helps in obtaining a more reliable estimate of the model's performance by reducing the impact of data variability.
  + **Test Set Size**: The size of the test dataset is important for reliable evaluation.
    - Generally, a larger test set provides a more accurate estimate of a model's performance.
    - However, it is essential to strike a balance between having enough data for testing and ensuring that the test set remains representative of the real-world distribution of data.
  + **Test Set Bias**:
    - Care must be taken to ensure that the test dataset is unbiased and represents the same distribution as the real-world data.
    - Biased test sets can lead to misleading performance estimates and inaccurate conclusions about the model's effectiveness.
  + **Model Prediction**:
    - The trained model is used to make predictions on the test dataset.
    - The inputs from the test dataset are fed into the model, and the model produces output predictions.

**2.2.2. Evaluation Metrics**: Various evaluation metrics are used to measure and quantify the performance of the model. Common evaluation metrics include accuracy, precision, recall, F1 score, mean squared error, etc. The choice of evaluation metrics depends on the specific problem and the nature of the data.

* **Accuracy**: especially used for classification tasks.
  + It measures the proportion of correct predictions made by the model out of the total predictions.
  + It is calculated by dividing the number of correct predictions by the total number of predictions.
* **Precision**: Precision is a metric that focuses on the correctness of positive predictions made by the model.
  + It measures the proportion of true positive predictions out of all positive predictions (both true positives and false positives).
  + It is useful when the cost of false positives is high.
  + It is calculated by dividing the number of true positives by the sum of true positives and false positives.
* **Recall**: also known as sensitivity or true positive rate:
  + Itmeasures the proportion of correctly predicted positive instances out of the total actual positive instances.
  + It is particularly useful when the cost of false negatives is high. Recall is calculated by dividing the number of true positives by the sum of true positives and false negatives.
* **F1 Score**: The F1 score is a harmonic mean of precision and recall.
  + It provides a balanced assessment of a model's performance, especially when dealing with imbalanced datasets.
  + It is calculated as .
* **Mean Squared Error (MSE):** MSE is primarily used for regression tasks, where the predicted values are continuous.
  + It measures the average squared difference between the predicted and actual values.
  + MSE gives a higher weight to larger errors, making it sensitive to outliers.
* **Performance Assessment**:
  + - The model's performance is assessed based on the evaluation metrics.
    - This assessment helps determine how well the model generalizes to unseen data and whether it meets the desired level of accuracy or performance.

3. Complete percentages for each task.

|  |  |  |
| --- | --- | --- |
| Task 1 | Optimizer methods | 100% |
| Task 2 | Continual Learning and Test Production | 100% |

4. REFERENCES

1. GeeksforGeeks, “Gradient descent in linear regression,” *GeeksforGeeks*, May 29, 2023. <https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/>
2. L. Giba, “Gradient descent for linear regression explained, step by step,” *Machine Learning Compass*, Apr. 03, 2021. <https://machinelearningcompass.com/machine_learning_math/gradient_descent_for_linear_regression/>
3. T. Vu, “Bài 7: Gradient Descent (phần 1/2),” *Tiep Vu’s Blog*, Jan. 12, 2017. <https://machinelearningcoban.com/2017/01/12/gradientdescent/#3-gradient-descent-cho-h%c3%a0m-nhi%e1%bb%81u-bi%e1%ba%bfn>
4. GeeksforGeeks, “ML Stochastic Gradient Descent SGD,” *GeeksforGeeks*, May 29, 2023. <https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/>
5. <https://edu.soict.ai/wp-content/uploads/2023/05/LinhNV_MML2023.pdf>