VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

LE PHUOC NHAN – 520H0559

FINAL REPORT
INTRODUCTION TO MACHINE LEARNING

HO CHI MINH CITY, 2023

VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

LE PHUOC NHAN – 520H0559

FINAL REPORT
INTRODUCTION TO MACHINE LEARNING

Instructor
PGS.TS. Le Anh Cuong

HO CHI MINH CITY, 2023

THANK YOU

I would like to express my sincere thanks to the instructor PGS.TS Le Anh Cuong and the Faculty of Information Technology for supporting and helping me a lot in the process of completing the report. In the process of doing the assignment, there were certain advantages and difficulties. Fortunately, those difficulties were promptly supported by the knowledge the Teacher provided during the class lessons for me to recognize and overcome. The report during the writing process will have shortcomings. I hope to receive the contributions of the teachers so that I can identify and make changes to complete it more completely.

I would like to send my sincere thanks to you and Faculty of Information Technology!

Ho Chi Minh City, day 22 month 12 year 2023
Author
(Sign and write your full name)

Nhan

Le Phuoc Nhan

WORK IS COMPLETED

AT TON ĐUC THANG UNIVERSITY

I hereby declare that this is my own research project and is under the scientific guidance of instructor PGS.TS Le Anh Cuong. The research content and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluation were collected by the author from different sources and clearly stated in the reference section.

In addition, the Project also uses a number of comments, assessments as well as data from other authors and other organizations, all with citations and source notes.

If any fraud is detected, I will take full responsibility for the content of my Project. Ton Duc Thang University is not involved in copyright violations caused by me during the implementation process (if any).

Ho Chi Minh City, day 22 month 12 year 2023

Author

(Sign and write your full name)

Nhan

Le Phuoc Nhan

TABLE OF CONTENTS

LIST OF DRAWINGS

LIST OF ABBREVIATIONS

GD	Gradient Descent
SGD	Stochastic Gradient Descent
RMSprop	Root Mean Square Propagation
Adam	Adaptive Moment Estimation
Adagrad	Adaptive Gradient Algorithm

CHAPTER 1. Explore and compare various optimization methods in training machine learning models

The term “optimizer” is commonly used to refer to an algorithm or method employed to optimize a loss function. The goal of the optimization process is to adjust the parameters of the model to minimize the value of the loss function, thereby improving the quality of the model.

1.1 Gradient Descent (GD)

The Gradient Descent (GD) algorithm is a fundamental optimization method widely used in machine learning to adjust model parameters and minimize the loss function. The goal of GD is to move from a randomly chosen starting point in the parameter space to an optimum point where the value of the loss function reaches its minimum.

In optimization problems, we often seek the minimum value of a certain function, where the function reaches its minimum value when the derivative is equal to 0. However, not all functions, especially for multivariable functions, have easily computable derivatives, and in some cases, derivatives may be infeasible. Instead, people look for a point close to the minimum and consider it a solution. Gradient Descent involves gradually reducing the slope, so the approach here is to choose a random solution and progressively move it towards the desired point after each iteration (or epoch).

The rule to remember: always move in the opposite direction of the derivative.

Gradient Descent for a single-variable function:

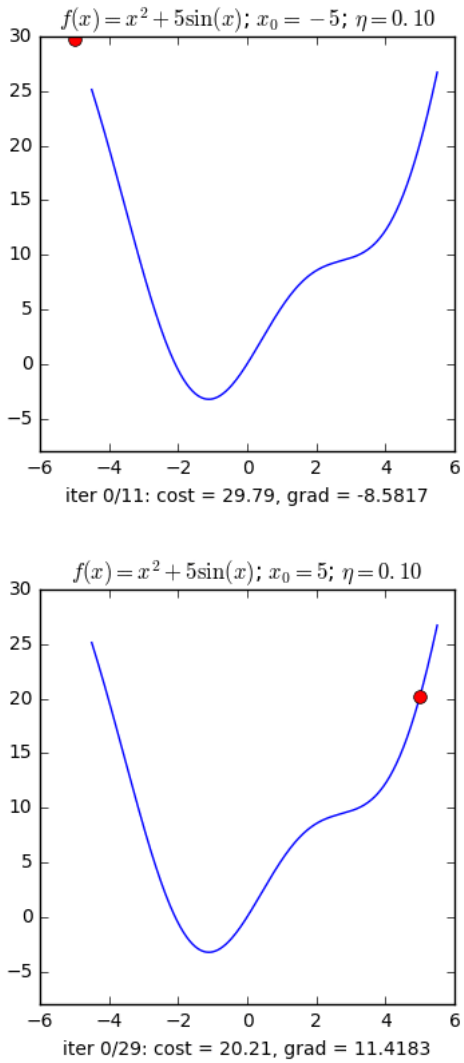

Figure 1.1.1: Example of Gradient Descent with x=5 and x=-5
(*link: https://machinelearningcoban.com/2017/01/12/gradientdescent*)

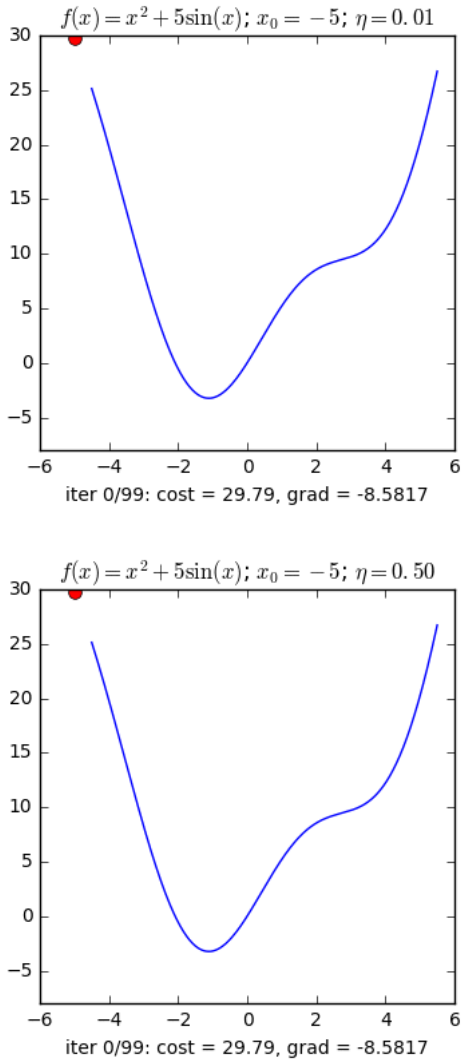

Figure 2.1.2: Example of Gradient Descent with learning rate of 0.01 and 0.5 (*link: https://machinelearningcoban.com/2017/01/12/gradientdescent*)

From the above figures, we can observe that Gradient Descent depends on several factors. For instance, choosing different initial points can impact the convergence process. Additionally, the learning rate plays a crucial role; if the learning rate is too large or too small, it can affect the convergence. If the learning rate is too small, the convergence speed becomes very slow, impacting the training process. On the other hand, if the learning rate is too large, the algorithm may quickly approach the target within a few iterations, but it may fail to converge, oscillating around the target due to excessively large steps.

Gradient Descent for a mutiple-variable function:

The gradient descent algorithm for multiple variables is an extension of the basic gradient descent used for optimizing a cost function with respect to a single variable. In the context of machine learning, this is often applied to minimize the cost function associated with a multivariate linear regression model.

Let's denote the cost function as $J(\theta_0, \theta_1, \dots, \theta_n)$, where $\theta_0, \theta_1, \dots, \theta_n$ are the parameters of the model. The goal is to find the values of these parameters that minimize the cost function.

In vectorized form, the update rule can be expressed as:

$$\theta = \theta - \alpha \nabla J(\theta)$$

Here, θ is the vector of parameters, α is the learning rate, and ∇J(θ) is the gradient vector of the cost function with respect to the parameters.

This approach is applicable not only to multivariate linear regression but also to other machine learning models where the goal is to optimize a cost function with respect to multiple parameters.

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- The basic gradient descent algorithm is easy to understand.
- The algorithm successfully addresses the optimization problem in neural network models by updating weights after each iteration.
- It exhibits the ability to optimize over large parameter spaces, reducing loss functions and updating model parameters.
- It is versatile and applicable to various types of models.

Disadvantages:

- The Gradient Descent algorithm has several limitations, such as dependence on the initial guess and learning rate.
- For a function with two global minima, the final solution will vary depending on the two initial points, leading to different outcomes.

- A too-large learning rate can prevent the algorithm from converging, causing it to oscillate around the target due to excessively large steps, while a too-small learning rate can impact training speed.
- Data needs to be preprocessed or smoothed for optimal performance.

1.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent (GD) algorithm widely used in machine learning to optimize model parameters. The main difference between SGD and GD lies in how gradients are computed and parameters are updated.

Instead of updating weights once per epoch, in each epoch with N data points, we update weights N times. On one hand, SGD reduces the speed of one epoch. However, from another perspective, SGD converges very quickly, often in just a few epochs. The formula for SGD is similar to GD but applied to each data point. Therefore, SGD is suitable for problems with large databases and tasks that require continuous model adjustments, i.e., online learning.

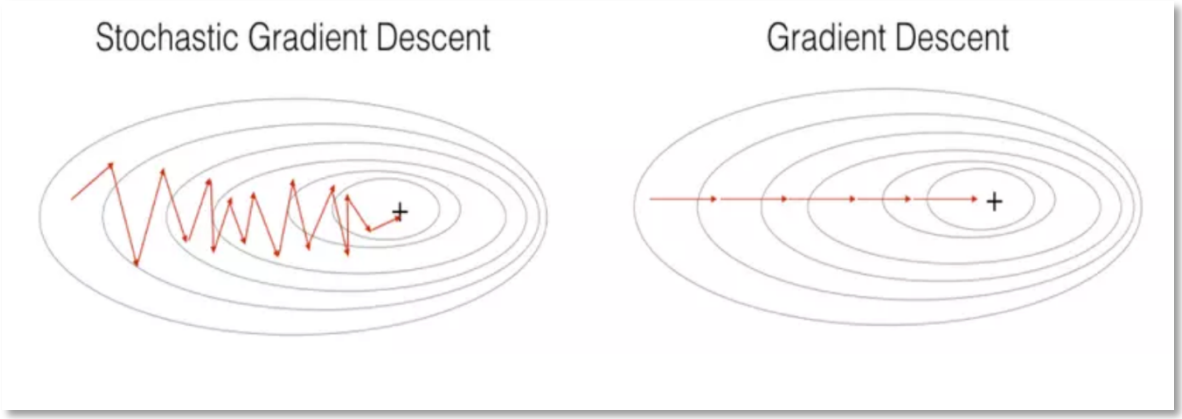

Figure 3.2.1: Images of Stochastic Gradient Descent and Gradient Descent

Online learning is when the database is continuously updated (an example is adding new user registrations daily), adding a few new data points each time. If we were to recompute the gradient of the loss function at all data points, the computation time would be very long, and our algorithm would no longer be considered online due to excessive computational time.

Therefore, there is a simpler and more efficient algorithm called Stochastic Gradient Descent (SGD).

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- Optimization algorithm.
- Addresses problems on large databases.

Disadvantages:

- Still doesn't solve the issues of learning rate and initial data point as in Gradient Descent.
- Significant oscillations if a fixed learning rate is used.

1.3 Momentum

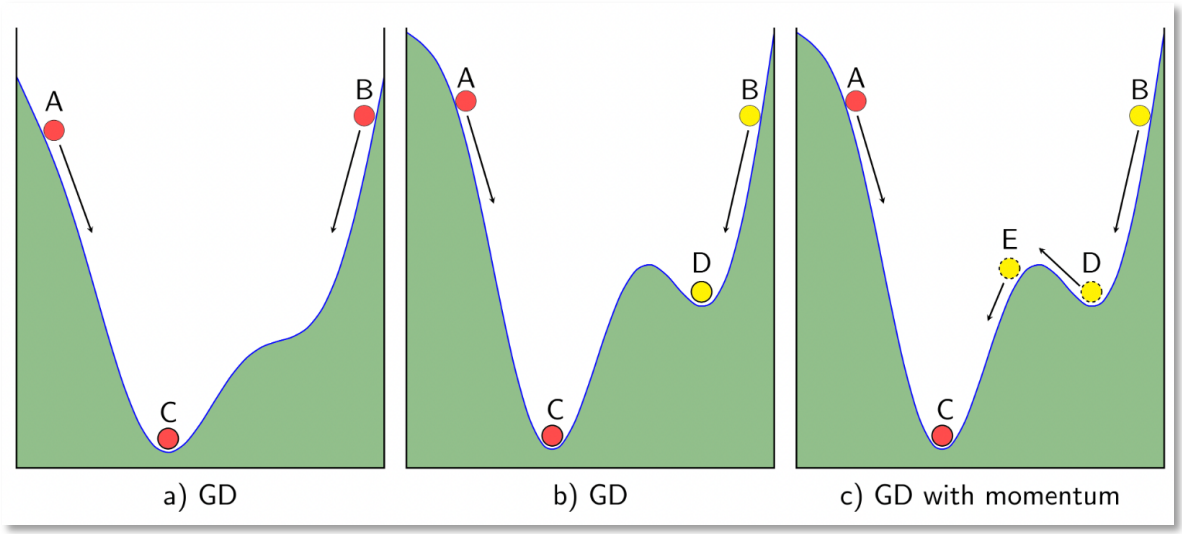

Figure 4.3.1: Exemple of Momentum

To explain Gradient with Momentum, let's first consider it from a physics perspective: As shown in diagram b above, if we release two balls from two different points A and B, the ball from A will slide down to point C, and the ball from B will slide down to point D. However, we do not want the ball from B to stop at point D (local minimum) but to continue rolling to point C (global minimum). To achieve this, we need to provide the ball from B with an initial velocity large enough for it to

overcome point E and reach point C. Based on this concept, the Momentum algorithm is developed (meaning, following the momentum).

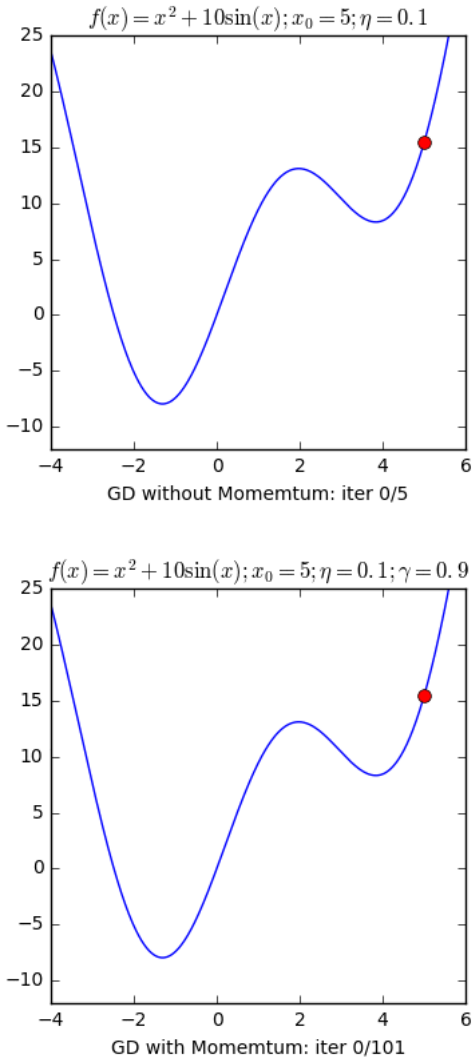

Figure 5.3.2: Exemple of Momentum
(*link: https://machinelearningcoban.com/2017/01/16/gradientdescent2*)

Through the example, we can observe the figure on the left (without using Momentum), where the algorithm converges after only 5 iterations, but the obtained solution is a local minimum. On the right side (using Momentum), the ball is able to overcome the slope to the region near the global minimum, then oscillates around this point, gradually slows down, and finally reaches the destination. It may take more iterations, but GD with Momentum provides us with a more accurate result.

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- Gradient Descent may not reach the global minimum and instead stops at a local minimum.

Disadvantages:

- RMSprop tends to find a solution that is only a local minimum rather than reaching the global minimum like Momentum.
- RMSprop also has the drawback of potentially leading to "vanishing gradients" in certain situations.

1.4 RMSprop (Root Mean Square Propagation)

RMSprop (short for Root Mean Square Propagation) is an optimization algorithm designed to enhance the optimization model of Adagrad, particularly to alleviate issues related to a rapidly decreasing learning rate. This algorithm preserves the dynamic learning rate characteristic of Adagrad but employs a technique to reduce the rapid change in the learning rate.

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- Addresses the issue of the decreasing learning rate in Adagrad. This problem, where the learning rate decreases over time, can slow down the training process and potentially lead to stagnation.

Disadvantages:

- RMSprop tends to find a solution that is only a local minimum rather than reaching the global minimum like Momentum.
- RMSprop also has the drawback of potentially leading to "vanishing gradients" in certain situations.

1.5 Adam

Adam (short for Adaptive Moment Estimation) is a popular optimization algorithm in machine learning and deep learning. It combines the advantages of the

Momentum and RMSprop algorithms to enhance convergence and model optimization.

It uses a momentum component to track the momentum of the gradient and an adaptive learning rate component (RMSprop) to reduce the rapid changes in the learning rate.

Momentum is like a ball rolling down a hill, while Adam is like a very heavy ball with friction. Therefore, it easily overcomes local minima to reach the global minimum, and when it reaches the global minimum, it does not oscillate much around the target because of its friction, making it easier to come to a stop.

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- Adam combines both components of the Momentum and RMSprop algorithms, balancing the ability to track momentum and reduce rapid changes in the learning rate.
- Adam automatically adjusts the learning rate for each parameter, reducing the need for manual selection of learning rates.

Disadvantages:

- Requires significant computational resources, especially when dealing with large datasets and complex models.
- In some cases, Adam may encounter issues with the rapid changes in the learning rate.

1.6 Adagrad

Adagrad (short for Adaptive Gradient Algorithm) is an optimization algorithm designed to automatically adjust the learning rate of each parameter in the model based on the history of the gradient. It is designed to address challenges associated with fixed learning rates in various optimization algorithms.

With this algorithm we also have the following advantages and disadvantages.

Advantages:

- No need to manually adjust the learning rate.

- Converges faster.
- More reliable.

Disadvantages:

- There is a risk that Adagrad may "overlearn" for parameters with large gradients, leading to overfitting.
- Due to the need to store the gradient history for each parameter, Adagrad requires significant memory, especially when working with large models.
- The accumulation of squared gradients can lead to a too rapid decrease in the learning rate for parameters with large gradients, causing the issue of "gradient vanishing."

CHAPTER 2. Study Continual Learning and Test Production when constructing a machine learning solution to address a specific problem

2.1 Continual Learning

Continual learning, alternatively termed lifelong learning or incremental learning, pertains to a branch of machine learning where models are crafted to continually acquire and integrate knowledge from new data without erasing previously acquired knowledge. The objective is to guarantee that the model sustains effective performance on fresh data over an extended duration, even when the model has undergone prior training on distinct datasets.

Main characteristics of Continuous Learning:

- Retention of Past Knowledge: Continual learning models are constructed to preserve and utilize knowledge gained from past data when confronted with new data.
- Adaptability: These models possess the capability to learn from fresh data and adapt to shifts in data distribution over time.

- Task Diversity Management: Continual learning models excel in handling diverse tasks and continually expanding their knowledge without experiencing significant performance deterioration on prior tasks.
- Parameter Adjustment: Effectively managing the adjustment of model parameters is a critical challenge to ensure that old knowledge is retained while acquiring new knowledge.
- Handling Extensive Data: Continual learning must address the challenge of processing extensive and potentially intricate data that may exhibit variations over time.

Advantages:

- Adaptability to Dynamic Environments: Continual learning empowers models to adjust to evolving data distributions and dynamic environments, rendering them suitable for applications where data evolves over time.
- Efficient Resource Utilization: Models can make use of previously acquired knowledge, diminishing the necessity for extensive retraining from the ground up and optimizing the utilization of computational resources.
- Avoidance of Catastrophic Forgetting: Continual learning approaches seek to alleviate catastrophic forgetting, ensuring that models preserve knowledge gained from previous tasks when encountering new ones.
- Incremental Learning: The capacity for incremental learning enables models to handle novel tasks without retraining on the entire dataset, facilitating swift adaptation to emerging patterns.
- Improved Generalization: Continual learning has the potential to enhance a model's generalization performance by consistently exposing it to new and diverse data, allowing it to capture a broader spectrum of patterns.
- Memory and Experience Retention: Models can integrate memory or replay mechanisms, enabling them to retain crucial experiences and patterns from the past, contributing to superior long-term performance.

Disadvantages:

- Catastrophic Forgetting Risk: Despite attempts to address it, the risk of catastrophic forgetting persists as a challenge. There is a potential for new learning to disrupt previously acquired knowledge, resulting in a substantial decline in performance on older tasks.
- Task Interference: Confronted with a diverse stream of tasks, continual learning models may encounter difficulty managing interference between tasks, potentially affecting their ability to specialize in specific domains.
- Complexity of Model Design: Crafting models for continual learning can be intricate, necessitating additional mechanisms like memory, regularization, or replay. This complexity may introduce challenges in the training and deployment of these models.
- Task Boundary Identification: The accurate identification of task boundaries is pivotal, and misidentifying them may lead to erroneous learning behavior. This requires careful consideration, and the definition of tasks may not always be straightforward.
- Overfitting to Old Data: Continual learning models may grapple with avoiding overfitting to old data, especially if the distribution of tasks is highly diverse or if there is insufficient new data for adaptation.
- Computational Resource Requirements: Implementing continual learning may impose substantial computational demands, particularly when memory mechanisms or replay strategies are employed. This can impact scalability and efficiency.
- Limited Data for New Tasks: In situations where new tasks have limited labeled data, continual learning models may encounter challenges in effective generalization. Striking a balance between adapting to new information and retaining old knowledge becomes crucial.

2.2 Test Production

Testing machine learning models in production is a crucial step to ensure their performance, reliability, and accuracy when deployed in real-world scenarios.

Types of tests:

- Unit tests: examinations of individual components, each assigned a single responsibility (e.g., a function filtering a list).
- Integration tests: evaluations of the collective functionality of individual components (e.g., data processing).
- System tests: assessments of the overall system design to validate expected outputs for given inputs (e.g., training, inference, etc.).
- Acceptance tests: verifications ensuring that specified requirements have been fulfilled, commonly known as User Acceptance Testing (UAT).
- Regression tests: examinations based on past errors to prevent the reintroduction of known issues with new changes.

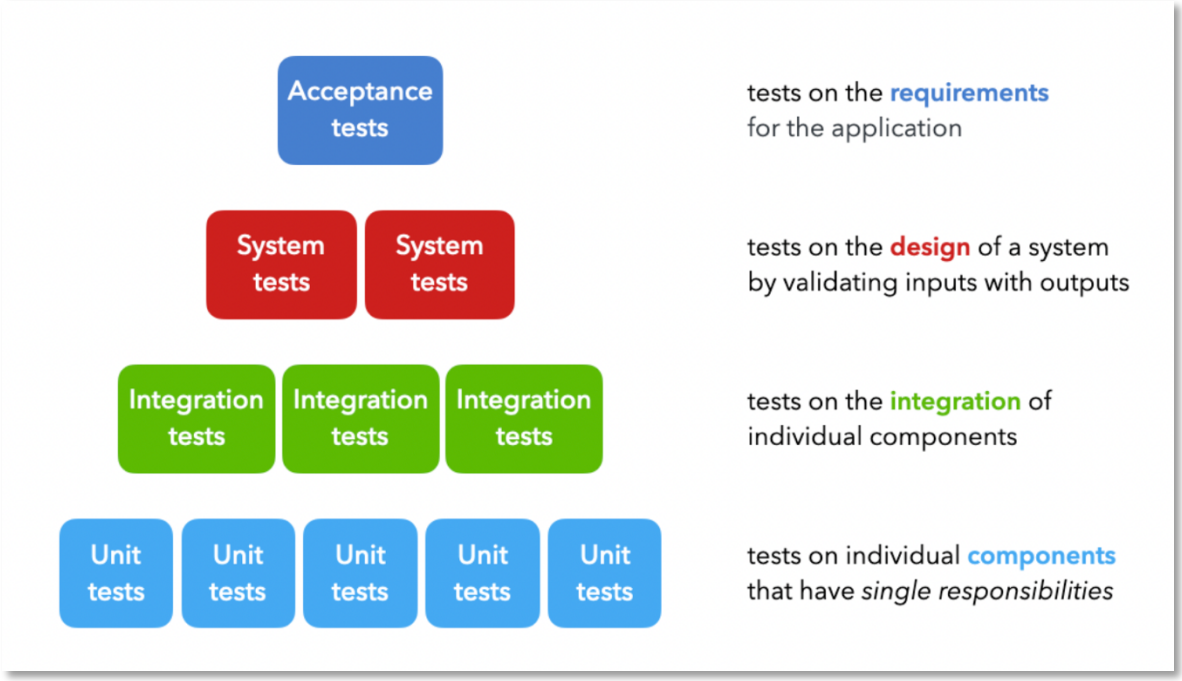

Figure 2.2.1: Information about types of test

When testing a machine learning model, it is important to consider the following:

- Data Quality: Ensure that the data used for testing is of the same quality as the data that will be used in production. This includes checking for missing values, outliers, and other anomalies.
- Scalability: Test the model's ability to handle large amounts of data and high traffic volumes. This will help ensure that the model can handle the demands of a production environment.
- Performance: Test the model's performance under different conditions, such as varying levels of traffic or changes in the data. This will help identify any potential issues that may arise in a production environment.
- Security: Ensure that the model is secure and that sensitive data is protected. This includes testing for vulnerabilities and implementing appropriate security measures.

One method for testing machine learning models in a production environment is to use a staging environment. This is a separate environment that is identical to the production environment but is used for testing and development purposes. This allows developers to test the model in a controlled environment before deploying it to production.

Overall, testing machine learning models in a production environment is a critical part of Machine Learning Operations. By ensuring that the model is tested thoroughly and can handle the demands of a production environment, businesses and organizations can ensure that their machine learning models are effective and reliable

REFERENCES

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

https://machinelearningcoban.com/2017/01/12/gradientdescent/

https://madewithml.com/courses/mlops/testing/

https://deepchecks.com/how-to-test-machine-learning-models/

https://www.datacamp.com/blog/what-is-continuous-learning