**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**MACHINE LEARNING**

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

*Student*: **NGUYEN SONG HUNG – 521H0399**

*Class* **: 503044**

*Year* **:** **25**

**HO CHI MINH CITY, 2023**

**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**MACHINE LEARNING**

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

*Student*: **NGUYEN SONG HUNG – 521H0399**

*Class* **: 503044**

*Year* **:** **25**

**HO CHI MINH CITY, 2023**

**ACKNOWLEDGEMENT**

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

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

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

*Author*

*(Sign and write full name)*

Page **3** of **24**

**THIS PROJECT WAS COMPLETED AT**

**TON DUC THANG UNIVERSITY**

We fully declare that this is our 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, 11th October, 2023*

*Author*

*(Sign and write full name)*

Page **4** of **24**

**CONFIRMATION AND ASSESSMENT SECTION**

**Instructor confirmation section**

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

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

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

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

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

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

| *Ho Chi Minh* | *, 2023* |
| --- | --- |

*(Sign and write full name)*

**Evaluation section for grading instructor**

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

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

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

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

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

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

| *Ho Chi Minh* | *, 2023* |
| --- | --- |

*(Sign and write full name)*

Page **5** of **24**

**SUMMARY**

1 Understanding and Comparing Optimization Methods in Machine Learning:

During the training of machine learning models, selecting an appropriate optimization method (Optimizer) is crucial. There are various optimization methods, including Gradient Descent, Stochastic Gradient Descent (SGD), Momentum, RMSprop, Adam, Adagrad, etc. Each method has its own advantages and disadvantages and may perform well on specific types of models. Comparing these methods requires a deep understanding of how they work and their impact on the model's learning process.

2 Continual Learning and Test Production in Machine Learning Solutions:

Continual Learning is a crucial field in machine learning where models are trained to continue learning from new data without forgetting what they have learned from previous data. This is particularly useful when building machine learning models for real-world problems where data evolves over time.

On the other hand, Test Production involves testing and ensuring that the model operates as expected when deployed in a production environment. This includes performance testing, accuracy verification, and assessing the model's ability to meet specific problem requirements.

Page **6** of **24**

**TABLE OF CONTENT**

[**I. Introduce 8**](#_845u5h8di9zj)

[1 – Presenting a research paper and my assessment of Optimizer issues 8](#_9hxovvjucde2)

[1.1. What is Optimizer 8](#_9mz139scuny2)

[1.2. Why use it? 9](#_aj7vtkhijrjv)

[1.3. Optimization algorithms? 10](#_37ust058aueq)

[1. Gradient Descent (GD) 10](#_g62su384fmqf)

[2. Momentum 11](#_quzxpq23seao)

[3. Stochastic Gradient Descent 13](#_43sh77fiv7t1)

[4. Mini-batch Gradient Descent: 15](#_8ihtcu9ta1u3)

[5. Adam (Adaptive Moment Estimation): 16](#_ihs30upl7qx5)

[6. RMSprop (Root Mean Square Propagation): 18](#_sw4x9t3pv3ui)

[7. RMSprop (Root Mean Square Propagation): 19](#_ku4mumwumdok)

[2 – Continual Learning and Test Production 21](#_xcp11a5xiizm)

[2.1. Continual learning? 21](#_y3aogp5zwezx)

[2.2. Test production? 22](#_rfaixtgck4aj)

[**REFERENCES 25**](#_nwudb3rbnrct)

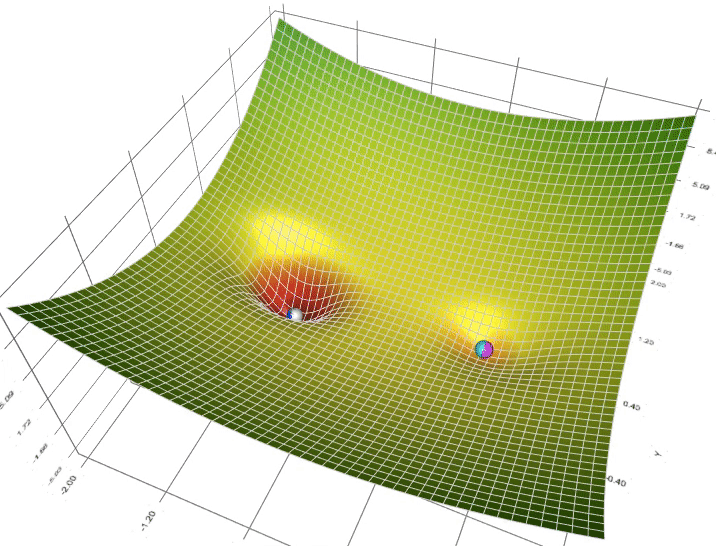
Page **7** of **24**

QUESTION 1

# I. Introduce

## 1 – Presenting a research paper and my assessment of Optimizer issues

### 1.1. What is Optimizer



In machine learning, an optimizer is a crucial component of the model training process. It is used to optimize the loss function by adjusting the model parameters so that the model can learn optimally from the training data.

The goal of using an optimizer is to find the minimum or near-minimum value of the loss function. This is particularly important in training machine learning models because it helps the model learn more effectively from the data, leading to more accurate predictions.

The optimizer plays a vital role in iteratively updating the model parameters during training to minimize the discrepancy between the predicted output and the actual target values. By adjusting the parameters in the right direction, the optimizer guides the model towards a configuration that yields the lowest possible loss.

Different optimization algorithms exist, each with its own strengths and weaknesses. These algorithms aim to strike a balance between the speed of convergence and avoiding issues such as getting stuck in local minima. Common optimization algorithms include Gradient Descent, Stochastic Gradient Descent, Mini-batch Gradient Descent, Adam, RMSprop, and others.

### 

### 1.2. Why use it?

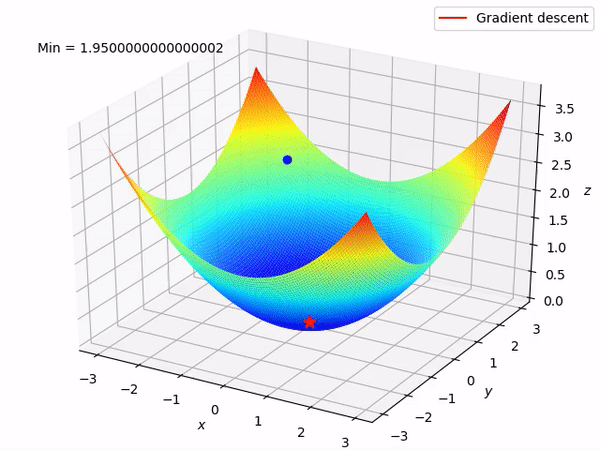
The reasons for using an optimizer in machine learning are as follows:

1. **Loss Function Optimization:**
   * *Objective:* It helps find optimal parameters for the model by adjusting them to minimize the loss function.
   * *Explanation:* The optimizer plays a crucial role in iteratively updating the model parameters to reduce the discrepancy between predicted and actual values, thus optimizing the model for better performance.
2. **Training Acceleration:**
   * *Objective:* The optimizer facilitates faster model learning by intelligently updating parameters, thereby reducing training time.
   * *Explanation:* Through smart parameter updates, optimizers enable the model to converge more quickly during the training process, leading to faster learning and more efficient use of computational resources.
3. **Avoiding Local Minima:**
   * *Objective:* Some optimization methods have the ability to escape local minima and approach the global minimum of the loss function.
   * *Explanation:* By preventing the model from getting stuck in local minima, optimizers help explore the parameter space more effectively, increasing the likelihood of reaching a globally optimal solution.

### 1.3. Optimization algorithms?

### 

#### Gradient Descent (GD)



Gradient Descent is a widely used optimization algorithm in machine learning and related fields. The goal of Gradient Descent is to find the minimum value of a function by adjusting its parameters through iterative steps.

The algorithm works by computing the gradient of the function (partial derivatives with respect to each parameter), indicating the direction and rate of increase of the function. It then moves in the opposite direction of this gradient to reduce the value of the function.

There are two main types of Gradient Descent:

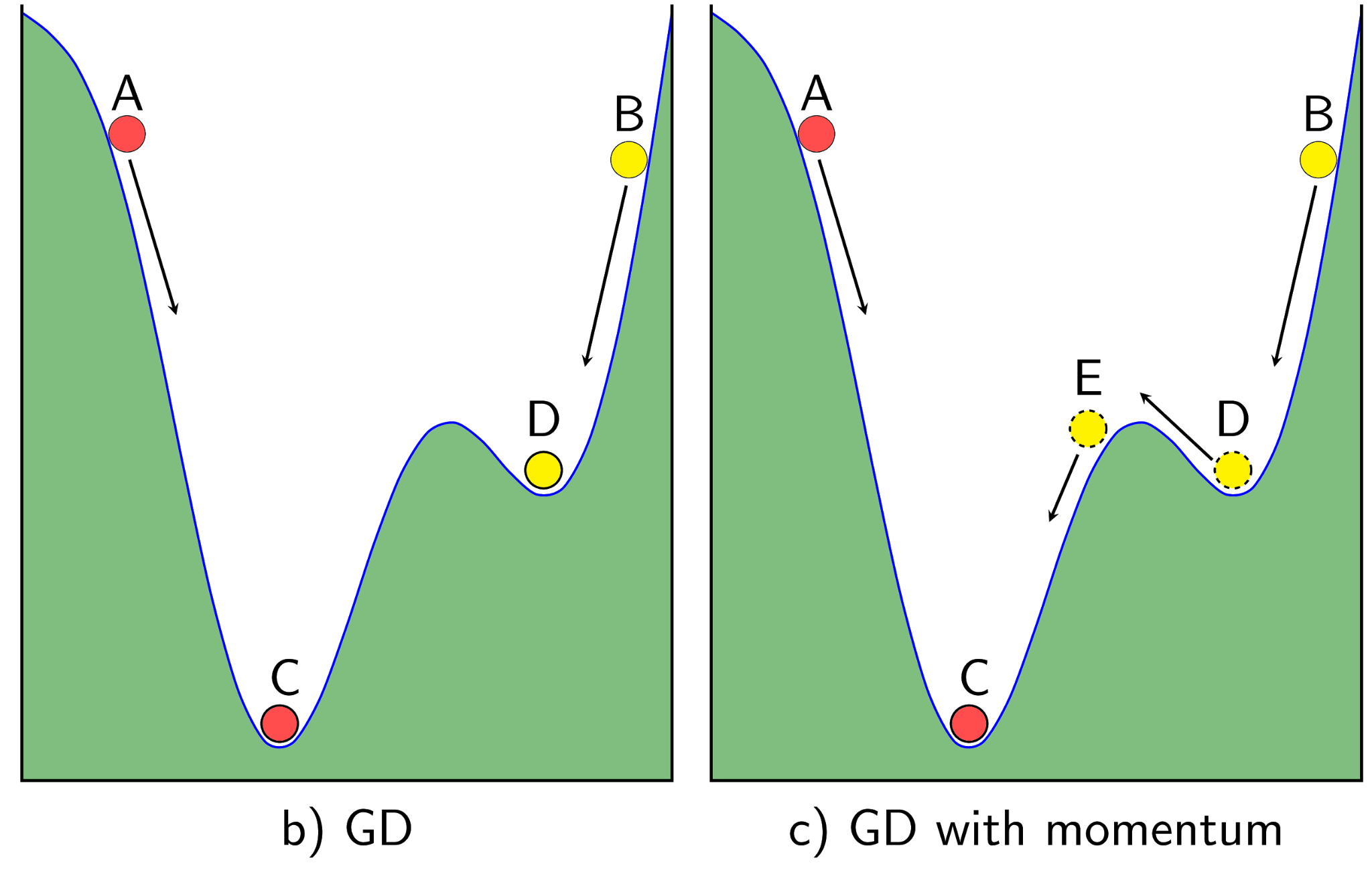
**Advantages of Gradient Descent (GD):**

* The basic gradient descent algorithm is easy to understand. The algorithm effectively addresses the optimization problem in neural network models by updating weights after each iteration.

**Disadvantages of Gradient Descent (GD):**

* Due to its simplicity, the Gradient Descent algorithm has limitations, such as dependency on the initial solution and learning rate.
* For example, for a function with two global minima, depending on the two initial points, the algorithm may converge to different final solutions.
* A too-large learning rate can cause the algorithm not to converge, wandering around the target because the step size is too large. Conversely, a small learning rate affects the training speed.

#### Momentum



To overcome the limitations of the Gradient Descent algorithm, one can use the Gradient Descent with Momentum. So, what is Gradient Descent with Momentum?

Gradient Descent with Momentum is an optimization technique in the process of training machine learning models, especially in updating the weights of neural networks. This technique is designed to accelerate convergence and avoid undesired oscillations during optimization.

The mechanism of Gradient Descent with Momentum relies on accumulating momentum from previous gradients to update weights. Instead of using only the current gradient to adjust weights, Gradient Descent with Momentum computes the weight change based on the current gradient and accumulated momentum from previous steps.

The formula for updating weights with Gradient Descent with Momentum is typically represented as follows:

Δ*w*(*t*)=*α*⋅Δ*w*(*t*−1)−*η*⋅∇*J*(*w*)

Where:

* Δ*w*(*t*) is the weight update at time t.
* Δw(t−1) is the accumulated momentum from the previous step.
* *α* is the momentum coefficient, usually a value between 0 and 1 to adjust the influence of accumulated momentum.
* *η* is the learning rate.
* ∇*J*(*w*) is the gradient of the loss function *J* with respect to the weights *w*.

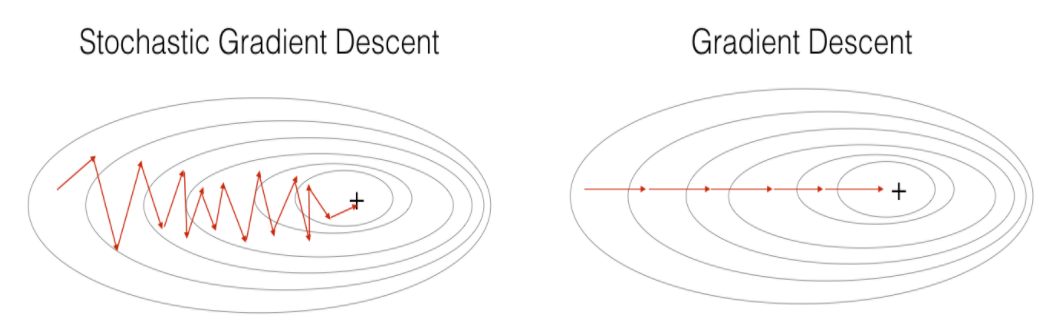
**Advantages of Momentum:**

* One noticeable benefit of Adagrad is the automatic adjustment of the learning rate without manual tuning. By setting the default learning rate to, for example, 0.01, the algorithm will autonomously make adjustments as needed.

**Disadvantages of Momentum**

* A significant drawback of Adagrad is the accumulation of squared gradients over time, leading to an increasing denominator in the learning rate formula. This can cause the learning rate to become exceedingly small, effectively freezing the training process. As a result, Adagrad may struggle with adapting to changing data distributions or maintaining effective learning rates throughout the entire training duration.

#### Stochastic Gradient Descent



The passage you provided describes the Stochastic Gradient Descent (SGD) optimization algorithm. In this algorithm, at each iteration, the derivative of the loss function is calculated based on only one data point *xi*​, and then the parameters *θ* are updated based on this derivative. This process is repeated for each data point in the entire dataset, and this constitutes one epoch. In standard Gradient Descent (GD), each epoch corresponds to one update of *θ*, while in SGD, each epoch corresponds to *N* updates of *θ*, where *N* is the number of data points.

Although updating parameters point by point may seem to reduce the speed of each epoch, SGD has proven to be very effective in practice. While it might slow down the process of completing one epoch, SGD typically requires a small number of epochs (often around 10 for the first run, and even fewer for subsequent runs with new data). Therefore, SGD is well-suited for problems with large datasets (commonly encountered in deep learning, as discussed in later sections) and for tasks that require continuous model adaptation, such as online learning.

**Advantages of Stochastic Gradient Descent (SGD):**

1. **Computational Efficiency:**
   * SGD is computationally more efficient compared to batch gradient descent, especially for large datasets. It processes one data point at a time, requiring less memory and allowing for online learning.
2. **Quick Convergence:**
   * Due to its frequent updates, SGD can converge faster than batch gradient descent. This is particularly advantageous when dealing with large datasets where updating the model parameters for the entire dataset can be computationally expensive.
3. **Online Learning:**
   * SGD is well-suited for online learning scenarios, where the model needs to adapt continuously as new data becomes available. It can efficiently update the model with each new data point.
4. **Escape from Local Minima:**
   * The noisy updates introduced by processing individual data points can help the optimization algorithm escape from local minima. This property can be beneficial in complex, non-convex optimization landscapes.

**Disadvantages of Stochastic Gradient Descent (SGD):**

1. **Noisy Updates:**
   * The use of individual data points makes the updates noisy, which can lead to fluctuations in the convergence path. This noise can make it harder to converge to the global minimum, especially in the presence of outliers.
2. **Irregular Convergence:**
   * The stochastic nature of the updates can result in irregular convergence. The loss function may not decrease smoothly, making it challenging to determine when the optimization process has reached a satisfactory solution.
3. **Sensitive to Learning Rate:**
   * The choice of learning rate is crucial in SGD. If the learning rate is too high, the algorithm may oscillate or fail to converge. If it's too low, the convergence may be slow.
4. **Difficulty in Choosing Hyperparameters:**
   * Selecting appropriate hyperparameters, such as the learning rate and the number of epochs, can be challenging. These parameters may need to be fine-tuned for different datasets.

#### Mini-batch Gradient Descent:

Mini-batch Gradient Descent is an optimization algorithm used in machine learning and deep learning. It is a compromise between two other popular optimization algorithms: Gradient Descent (GD), which processes the entire dataset in each iteration, and Stochastic Gradient Descent (SGD), which processes only one data point at a time. In Mini-batch GD, the dataset is divided into smaller batches, and each batch is used to update the model parameters.

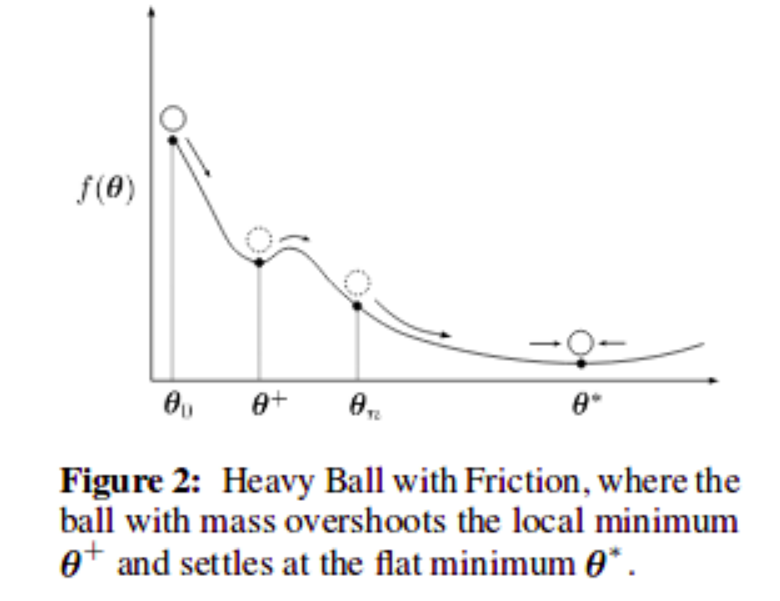
**Advantages:**

1. **Efficiency:**
   * Mini-batch GD strikes a balance between the computational efficiency of GD and the reduced noise of SGD. It processes a subset of the data, making it computationally more efficient than GD.
2. **Better Generalization:**
   * Mini-batches introduce a level of noise that can help the model generalize better. This can prevent the algorithm from getting stuck in local minima and improve convergence.
3. **Parallelization:**
   * Mini-batch processing allows for parallelization, especially in the case of large datasets. Different batches can be processed concurrently, leading to faster training times.
4. **Memory Efficiency:**
   * Mini-batches require less memory compared to processing the entire dataset at once, making it feasible to train models on hardware with limited memory.

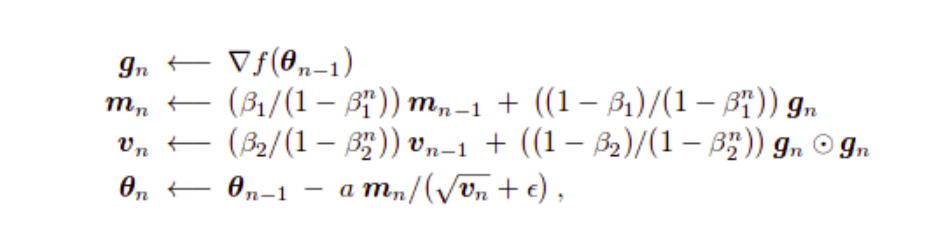
**Disadvantages:**

1. **Hyperparameter Tuning:**
   * Selecting an appropriate mini-batch size requires careful consideration. The choice of mini-batch size can impact the convergence speed and the quality of the final model.
2. **Noisy Updates:**
   * The noise introduced by mini-batch updates may result in a less stable convergence, especially in the early stages of training. This noise can cause the algorithm to oscillate around the minimum rather than converging smoothly.
3. **Sensitive to Learning Rate:**
   * The choice of learning rate becomes crucial in Mini-batch GD. An inappropriate learning rate can lead to slow convergence or divergence.

#### Adam (Adaptive Moment Estimation):



**Formula:**



Adam is an optimization algorithm used in machine learning and deep learning for training artificial neural networks. It combines the ideas of both momentum optimization and RMSprop. Adam computes adaptive learning rates for each parameter by considering both the first-order moment (mean) and the second-order moment (uncentered variance or standard deviation) of the gradients.

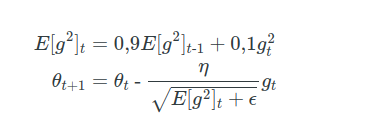
**Advantages:**

1. **Adaptive Learning Rate:**
   * Adam adjusts the learning rate for each parameter individually based on the historical gradients. This adaptability can be advantageous in situations where different parameters may require different learning rates.
2. **Efficiency:**
   * Adam is computationally efficient and requires less memory compared to methods that store historical gradients for each parameter separately.
3. **Suitability for Sparse Gradients:**
   * Adam is well-suited for scenarios with sparse gradients. It can handle situations where some parameters have frequent updates while others remain relatively static.
4. **No Manual Tuning of Learning Rate:**
   * Adam reduces the need for manual tuning of the learning rate, as it dynamically adjusts the learning rates based on the past gradients.
5. **Stability:**
   * Adam is known for providing stable and consistent convergence in a wide range of scenarios.

**Disadvantages:**

1. **Sensitivity to Hyperparameters:**
   * Adam involves hyperparameters such as β₁ (decay rate for the first moment estimate) and β₂ (decay rate for the second moment estimate). The performance of Adam can be sensitive to the choice of these hyperparameters.
2. **Memory Requirements:**
   * Although Adam is more memory-efficient than methods that store historical gradients separately for each parameter, it still requires additional memory to maintain the moving averages.
3. **Limited Theoretical Guarantees:**
   * Unlike some simpler optimization algorithms, Adam lacks strong theoretical guarantees regarding convergence. The choice of hyperparameters may affect the algorithm's behavior.

#### RMSprop (Root Mean Square Propagation):



RMSprop is an optimization algorithm used in machine learning and deep learning for training neural networks. It is designed to address some of the limitations of the standard Stochastic Gradient Descent (SGD) optimization algorithm, particularly those related to the choice of learning rates.

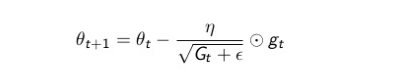
**Advantages:**

1. **Adaptive Learning Rates:**
   * RMSprop adapts the learning rates based on the historical information about the gradients, making it well-suited for scenarios where the gradients of different parameters vary widely.
2. **Stability:**
   * RMSprop helps stabilize the learning process by preventing the learning rates from fluctuating too much during training.
3. **Suitability for Non-Convex Optimization:**
   * RMSprop has been found to be effective in non-convex optimization problems, such as training deep neural networks.

**Disadvantages:**

1. **Hyperparameter Sensitivity:**
   * The performance of RMSprop can be sensitive to the choice of hyperparameters, including the decay rate (α) and the small constant (ε).
2. **Lack of Theoretical Guarantees:**
   * While RMSprop has shown empirical success, it lacks strong theoretical guarantees regarding convergence.

#### 7. RMSprop (Root Mean Square Propagation):



Unlike previous algorithms where the learning rate remains nearly constant during training (the learning rate is a constant), Adagrad treats the learning rate as a parameter that can vary over time. In other words, Adagrad adjusts the learning rate dynamically after each time step.

**In the formula:**

* *n*: a constant
* *gt*​: the gradient at time step *t*
* *ϵ*: a small constant to avoid division by zero
* *G*: a diagonal matrix where each element on the diagonal (i, i) is the squared derivative of the parameter vector at time step *t*.

**Advantages**:

* Adaptive Learning Rates:
  + Adagrad adapts the learning rates based on the historical information about the gradients, making it suitable for scenarios where some parameters may require larger or smaller updates.
* Sparse Data Handling:
  + Adagrad performs well on sparse data since it individually adjusts learning rates for each parameter, allowing it to handle features that occur infrequently.

**Disadvantages**

* Accumulation of Squared Gradients:
  + Over time, the sum of squared gradients can become large, causing the learning rates to become very small. This can lead to slow or stalled learning.
* Hyperparameter Sensitivity:
  + Adagrad's performance can be sensitive to the choice of hyperparameters, such as the initial learning rate and the small constant *ϵ*.

## 2 – Continual Learning and Test Production

Ảnh có chứa văn bản, hình mẫu, thiết kế đồ họa, Đồ họa

Mô tả được tạo tự động

### 2.1. Continual learning?

Continual learning is the idea of updating your model as new data becomes available, allowing your model to keep up with the current data distributions. After your model is updated, it cannot be blindly released into production. It needs to be tested to ensure that it is safe and performs better than the current production model. Therefore, the Test Production phase is crucial.

Why Continual Learning is Necessary: In today's world, data is continually generated and updated from various sources, and older data is often no longer as reliable due to changes in the environment or new trends. In the real world, machine learning models need to be flexible, capable of learning and adapting to new data to maintain performance and accuracy.

Continual learning addresses the issue of "forgetting" when machine learning models are trained on specific datasets and do not perform well when faced with new data. It helps the model retain the ability to learn and remember important information from old data while also being able to learn new information without losing previously acquired knowledge.

Main Steps of Continual Learning: a. Evaluate and Select Suitable Models:

* Choose machine learning models capable of continual learning and compatible with integrating new data.

b. Build Architecture for Continual Learning:

* Develop methods and algorithms so that the model can learn and retain old information while learning new data.

c. Data Management:

* Create strategies to manage new data, old data, and how to integrate them into the learning process without affecting performance.

d. Optimize Continual Learning:

* Optimize algorithms to ensure that the model does not forget important information when learning new data.

Challenges of Continual Learning: a. Forgetting Important Information:

* Models may forget crucial information from old data when exposed to new data, leading to "catastrophic forgetting."

b. Computational Complexity:

* Continual learning requires complex algorithms to ensure that the model can learn continuously without consuming excessive resources.

c. Performance and Accuracy:

* Striking a balance between learning new data and maintaining performance on old data is a significant challenge.

Continual learning is becoming a crucial research area in machine learning, offering prospects for building intelligent, flexible models capable of adapting to ever-changing environments. Progress in continual learning not only provides powerful real-world applications but also lays the foundation for advancing artificial intelligence.

### 2.2. Test production?

Test Production is a crucial process in ensuring that a machine learning model operates correctly and effectively before deploying it into a real-world environment. It plays a vital role in evaluating the model's quality, confirming its functionality, and assessing its performance before putting it to practical use in production.

The Test Production process typically begins after the model has been trained and validated on training and test data. However, the key difference is deploying the model into the real production environment to test its stability, performance, and ability to work with new data, not necessarily identical to the training data.

Some important steps in the Test Production process include:

* **Prepare New Data:**
  + Real-world production data may differ from the data used during training. Therefore, preparing and cleaning new data to test the model is crucial.
* **Deploy the Model:**
  + The model is deployed into the real-world environment to test its operation. This involves configuring the necessary infrastructure and connecting to the system so that the model can process real-time data or meet the requirements of the production environment.
* **Performance Testing:**
  + The model is evaluated for performance in the real-world environment, with metrics such as accuracy, classification, or other performance indicators depending on the machine learning task.
* **Monitoring and Updating:**
  + The Test Production process does not stop after the model is deployed. Continuous monitoring of the model's performance in the real environment and updating the model as needed is a crucial step to maintain and improve performance over time.
* **Security and Compliance:**
  + Data security and compliance with legal regulations are essential factors in the Test Production process, especially when the model processes personal or sensitive data.

The Test Production process not only helps evaluate the model but also aids in improving and optimizing it to best reflect its capabilities in a real-world environment. The accuracy and performance of the model in production can significantly impact business outcomes and the end-user experience. Therefore, conducting Test Production carefully and comprehensively is crucial.

# REFERENCES

1. <https://machinelearningcoban.com/2017/01/16/gradientdescent2/>
2. https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

Page **24** of **24**