**521H0287 – Văn Công Nguyên Phong**

**Câu 1: Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy:**

**What is Optimizer?**

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses.

There are some Optimal’s methods in training machine learning model:

* Gradient Descent (GD).
* Stochastic Gradient Descent (SGD) .
* Adam (Adaptive Moment Estimation).
* RMSprop (Root Mean Square Propagation).
* AdaGrad (Adaptive Gradient Algorithm).

**GRADIENT DESCENT**

1. **Concept**

A diagram of a weight loss

Description automatically generated

Gradient Descent(GD) is one of the most commonly used interative optimization algorithms, especially in the field of Machine Learning. This algorithms based on the finding the maximum or minimum local of a function by calculating the moving and derivative in the decreasing direction of gradient. It is extrememly useful in traning models for regression task or classification tasks. In additional, the aim purpose of gradient descent is to reduce a cost function in minimum by reset the adjust of paramenters from a model.

The cost function preresents the distinct between the prediction ouput and reality output and the goal of gradient descent is to find the set of parameters, which help to reduce the different and improve the model’s performance.

The algorithm works by the calculation the gradient of the cost function, which gives the direction and magnitude of steepest ascent. However, cause the objective is the minimize of the cost function so that the gradient descent will move in the opposite diẻction of gradient, which was called is negative direction.

Looking from an overall perspective, gradient descent can be applied in multiples machine learning algorithms, including linear regression, logistic regression, nerual networks, and support vector machine

1. **How does Gradient Descent work?**
2. Initialization: Starting by radomly creating the model’s parameters.
3. Forward Propagation: The model computers the predicted output by using the valid parameters.
4. Loss Calculation: Comparing the different between the predicted and actual output which is calculated by using a loss function.
5. Back Propagation: The model calculates the gradient of loss function with each parameter.
6. Parameter Update: The model will update each value of parameter by subracting a small part of gradient.
7. Interation: step 2 and 5 will be repeated until the algorithms converges or get the time of iteration was planed before.

**Note:** By iteratively updating the model's parameters using the gradients, gradient descent aims to find the optimal set of parameters that minimizes the cost function and improves the model's performance.

1. A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

   Description automatically generated**Learning Rate of Gradient Descent**

This is hyperparameter that define the size of each iterations while updateing the parameters of machine learning model. It will controls how fast of learning from the traning data. A high learning rate can make the model converge faster, but may also gets over the optim.

1. **How to solve gradient descent challenges?**

There are some steps to solve the gradient descent:

1. Understand the problem and define the variables: To ensure that you have a clear understanding the problem you are trying to solve by using gradient descent, define the function and variables will be fixed in the optimization process. These parameters will be updates by gradient descent algorithms.
2. Calculate the gradient: to define the object function from each parameters. The gradient the direction of steepest of descent in the function.
3. Initialze variables: Starting by initializing the variables to some initial values, these values will be opted randomly or based on the prior data.
4. Update the variables: the formular is

**variable = variable – learning rate \* gradient**

1. Check convergence: If the changes in the objective function or gradient become negligible, the algorithm has likely converged.
2. Adjust learning rate: if the algorithms are converging or not, adjust the learning rate.
3. Repeat: Repeat steps 3 to 7 until the algorithm converges or a predefined stopping criterion is met. This could be a maximum number of iterations or a desired threshold for the objective function.
4. Evaluate the solution: evaluating by calculating the value of objective function by using optimal parameters.
5. Iteration and refine: if the solution is not satisfied, you can iterate and refine the process by using problem formula, change the objective function or exploring different optimization algorithms.
6. **Advantages and Disadvantages of Descent Algorithm**

|  |  |
| --- | --- |
| **Advantages** | Basic gradient descent algorithm, easy to understand. The algorithm solved the problem of optimizing the neural network model by updating the weights after each loop. |
| **Disadvantages** | - Because of its simplicity, the Gradient Descent algorithm has many limitations such as depending on the initial initial solution and learning rate.  - For example, a function with 2 global minimums, depending on the 2 initial initial points, will produce 2 different final solutions.  - A learning rate that is too large will cause the algorithm to fail to converge and hang around the target because the jump is too large; or the small learning rate affects the training speed |

**STOCHASTIC GRADIENT DESCENT(SGD)**

1. **Concept**

SGD is a popular machine learning algorithm for interactive optimization. This variation of the gradient descent approach updates the parameter (weight) depending on the gradient of the loss function, which is calculated on a carefully selected subset of the training data rather than the entire set. The basic idea behind SGD is to take a small random subset of the model from the training data and use it just to compute the gradient of the loss function for each model parameter.

The gradient will then be utilized to update the parameters, and until the algorithm converges or meets a predetermined condition, this procedure is repeated with a fresh Radome mini-batch.The simple idea of SGD is getting the small random subset model from traning data and calculate the gradient of the loss function with each parameter of model, only used the subset.

The gradient then will be used tp update the paramenters and this process is repeated with a new radom mini-batch until the algorithm converges or reaches a define criterion before. Stochastic Gradient Descent is often more suitable when applied to large data because it can handle large datasets without having to store all the data in memory and can converge faster in some cases.

Example:

A diagram of a solar system

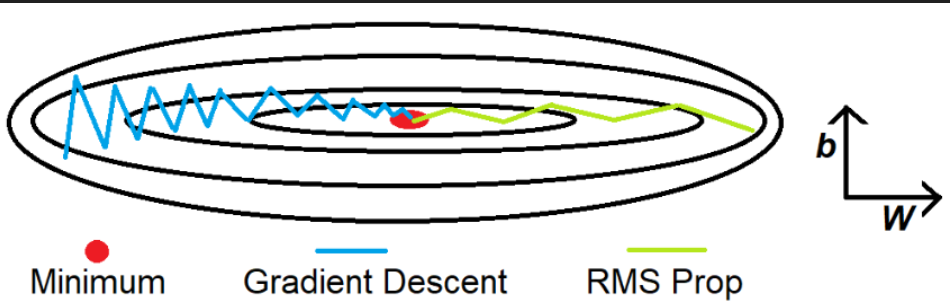
Description automatically generated with medium confidence

Looking from the illustration mentioned above, we can see the direction of SGD is flactuate, it is not smooth like GD, the reason why the point of data can nỏ represent for all data.

1. **Advantages and Disadvantages**

|  |  |
| --- | --- |
| **Advantages** | The algorithm can handle large databases that GD cannot. This optimization algorithm is still commonly used today.  SGD can converge faster than gradient descent, especially when the training data is large. SGD has the ability to escape local minima, which can help it achieve better results than standard gradient descent . |
| **Disadvantages** | The algorithm has not yet solved the two major disadvantages of gradient descent (learning rate, initial data points). Therefore, we have to combine SGD with some other algorithms such as Momentum, AdaGrad, etc. These algorithms will be presented in the following section. |

**ROOT MEAN SQUARE PROPAGATION(RMSprop)**

****

An optimization approach called RMSProp is used to maximize neural network training in deep learning and machine learning. RMSProp computes a moving average of the squares of the gradients, in contrast to Adagrad, which accumulates all gradients. This keeps the learning rate from falling too quickly and enables more gradual learning rate adjustments.

In order to give current gradients more weight, RMSProp also employs a decay factor to reduce the impact of earlier gradients. RMSProp's ability to effectively handle time-varying goal functions is one of its advantages. Meanwhile, Adagrad could converge too quickly in this situation. The learning rate can be modified by RMSProp in response to changes in the goal function.

**Advantages and Disadvantages**

|  |  |
| --- | --- |
| **Advantages** | * Adaptive Learning Rate: RMSprop adapts the learning rate for each parameter by dividing the learning rate by the root mean square of recent gradients for that parameter. This adaptive nature helps in addressing the challenges associated with learning rates in traditional stochastic gradient descent algorithms. * Improved Convergence: RMSprop tends to converge faster and more reliably compared to standard stochastic gradient descent, especially in scenarios where the input features have different scales or when dealing with non-stationary objectives. * Mitigation of Vanishing and Exploding Gradients: By keeping the moving average of squared gradients, RMSprop mitigates the risk of vanishing or exploding gradients, which can significantly impact the training of deep neural networks. * Robustness to Noisy Gradients: RMSprop exhibits robustness to noisy gradients, making it suitable for a wide range of real-world optimization problems. * Simplicity and Ease of Use: The algorithm is relatively simple to implement and use, making it accessible for practitioners to apply in various machine learning tasks. * Compatible with Sparse Data: RMSprop is well-suited for optimizing models trained on sparse datasets due to its ability to handle different learning rates for different parameters. |
| **Disadvantages** | * Sensitivity to Learning Rate: RMSprop can be sensitive to the choice of learning rate, and inappropriate learning rates may lead to suboptimal convergence or performance. Tuning the learning rate for different datasets or models can be a challenge. * Lack of Momentum: Unlike some other optimizers such as Adam, RMSprop does not incorporate momentum by default. Momentum can help accelerate convergence and escape shallow local minima, which may limit RMSprop's performance in certain cases. * Memory Requirements: RMSprop requires additional memory to store and maintain the moving average of squared gradients for each parameter, which may contribute to higher memory usage compared to simpler optimization algorithms. * Sensitivity to Hyperparameters: While all optimization algorithms require tuning of hyperparameters, RMSprop's performance can be particularly sensitive to specific hyperparameters such as the decay rate. Finding the right hyperparameters for optimal performance may require additional effort. * Lack of Bias Correction: RMSprop does not include bias correction for the moving averages of gradients, which may lead to suboptimal performance when working with smaller training datasets. |

**How does it work?**

1. Adaptive Learning Rate: RMSprop utilizes an adaptive learning rate for each parameter in the neural network. It achieves this by dividing the learning rate by the root mean square (RMS) of recent gradients for that parameter. This adaptive nature helps address the challenge of choosing an appropriate global learning rate, making it suitable for different types of data and architectures.
2. Squared Gradients for Weight Updates: RMSprop maintains a moving average of the squared gradients for each parameter. By doing so, RMSprop scales the learning rates individually for each parameter based on the magnitudes of the recent gradients, effectively giving larger updates to infrequently occurring parameters and smaller updates to frequently occurring parameters.
3. Weight Update Calculation: When updating the weights, RMSprop uses the scaled gradients to adjust the learning rates for each parameter, allowing for more rapid progress in the less frequently updated parameters while maintaining stability for the frequently updated ones.
4. Mitigation of Vanishing and Exploding Gradients: One of the crucial aspects of RMSprop is that it helps mitigate the issue of vanishing and exploding gradients, which can hinder the convergence of deep neural networks. By using the moving average of squared gradients, RMSprop normalizes the gradient updates, leading to more stable and effective training.

**ADAPTIVE MOMENT ESTIMATION(ADAM)**

A couple of fundamental ideas are combined in the method: momentum and RMSProp. A moving average is used by Adam to monitor the gradients' mean and variation. Since the gradient moving average takes into account small variations in the gradient, Adam can continue going in the same direction. By utilizing the variance section, Adam may set a different learning rate for each parameter.

To improve performance even further, Adam uses the technique of correcting the moving average's initial error. Because Adam requires fewer hyperparameters than some other methods, it converges rapidly and performs well with noisy and sparse gradients.

**Advantages and Disadvantages**

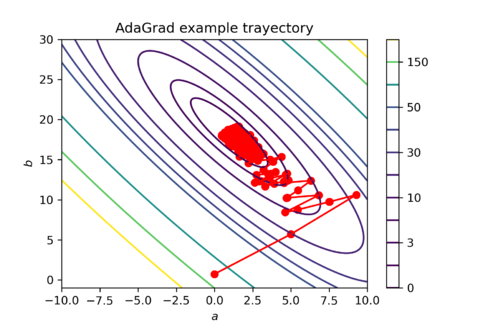
|  |  |
| --- | --- |
| **Advantages** | 1. Adaptive Learning Rate: Adam uses individually adaptive learning rates for each parameter, leading to improved convergence and faster training for deep learning models. 2. Effective with Sparse Data: Adam performs well with sparse data due to its ability to handle varying learning rates for different parameters. 3. Diversity and Flexibility: Adam combines two popular optimization methods, namely momentum and RMSprop, which enhances its adaptability and effectiveness for a wide range of machine learning tasks. |
| **Disadvantages** | 1. High Memory Requirements: Adam requires storing and maintaining information about gradients for each parameter, which can lead to significant memory requirements, especially for large and complex models. 2. Potential for Overfitting: Due to the adaptive learning rate nature of Adam, there is a risk of overfitting in certain scenarios, requiring careful monitoring and adjustment of hyperparameters to avoid this. |

**How does it work?**

To begin with, Adam maintains a separate learning rate for each parameter in the model, which allows it to adapt the learning rate on a per-parameter basis. It computes these learning rates based on the first and second moments of the gradients. Next, it calculates the first moment (meaning) of the gradients by exponentially decaying the previous gradients and incorporating the new ones. This helps Adam keep track of the direction of the gradients. Then, Adam also calculates the second moment (variance) of the gradients using the same exponential decay mechanism. This estimate provides information about the scale of the gradients, aiding in adjusting the learning rate.

After that, the algorithm then combines the first and second moment estimates to calculate the adaptive learning rates for each parameter. Additionally, Adam incorporates bias correction to overcome the initial bias towards zero that occurs in the first few iterations. And finally, Adam updates the model's parameters using the computed learning rates.

**ADAPTIVE GRADIENT ALGORITHM(AdaGrad)**



Adagrad (Adaptive Gradient) is an optimization approach that improves neural network training. The Adagrad method adaptively adjusts each neural network parameter's learning rate during training. To modify the learning rate of each parameter, it makes use of the historical gradients that have been computed for that parameter. Parameters with large gradients require more learning, while those with small gradients require less learning. Adagrad (Adaptive Gradient) is an optimization approach that improves neural network training.

The Adagrad method adaptively adjusts each neural network parameter's learning rate during training. To modify the learning rate of each parameter, it makes use of the historical gradients that have been computed for that parameter. Smaller learning rates are used to parameters with significant gradients, whereasƯu điểm và Nhược điểm:

|  |  |
| --- | --- |
| **Advantages** | * It eliminates the need to manually tune the learning rate. * Convergence is faster and more reliable – than simple SGD when the scaling of the weights is unequal. * It is not very sensitive to the size of the master step. |
| **Disadvantages** | * Learning Rate Decay: AdaGrad's accumulation of squared gradients over time can lead to a monotonically decreasing learning rate. This can cause the learning rate to become too small, resulting in very slow convergence or premature stopping in training. * Inflexibility with Sparse Data: In some cases, AdaGrad may not perform as effectively with sparse data, as the accumulation of squared gradients may overly diminish the learning rate for infrequently occurring features, thus impacting convergence. * Accumulation of Gradients: AdaGrad's accumulation of the squared gradients for each parameter can result in significant memory requirements, potentially limiting its applicability to very large models. * Lack of Momentum: AdaGrad does not incorporate momentum by default, which can impact its ability to accelerate convergence, particularly when dealing with complex and noisy loss landscapes. |

**How to work?**

Gradient descent methods have historically used a single learning rate for each parameter. Applying this to high-dimensional optimization situations, where certain dimensions demand bigger updates than others, may provide issues. Adagrad solves this problem by customizing the learning rate for every parameter.

The key idea behind Adagrad is to accumulate the sum of squares of past gradients for each parameter and use this information to scale the learning rate for new parameters. Mathematically speaking, the update at each iteration is given by:

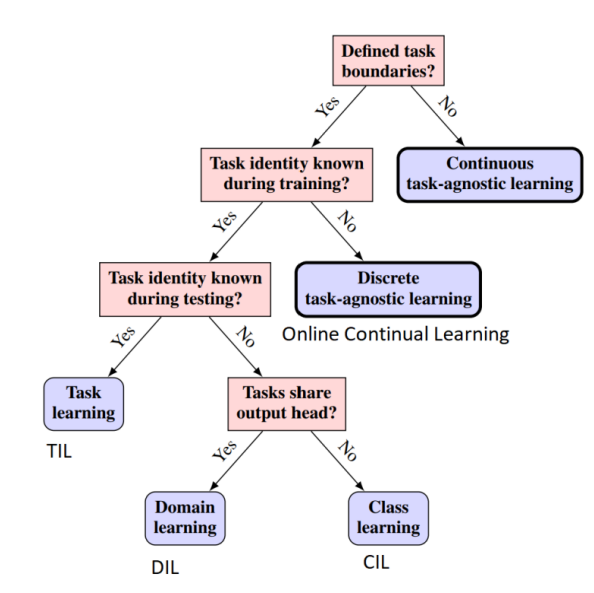
**θ = θ - (η / √G) \* g**

Here, G is the sum of squares of the previous gradients for that parameter, g is the current gradient, and θ is the parameter that is changed with each iteration. η is the learning rate.Large gradient parameters have lower learning rates due to this update algorithm, whereas small gradient parameters have higher learning rates. In addition to preventing oscillations that impede optimization, this enhances convergence.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms** | **Character** | **Advantages** | **Disadvantages** |
| **Gradient Descent** | * Update the weights based on the derivative of the entire training data set | - Easy to implement and understand. | - Can converge slowly. |
| **Stochastic Gradient Descent** | * Update the weights based on each training data point randomly. | - Fast calculation with large data. | - Unstable in updating weights. |
| **Adam** | * Combines the advantages of RMSprop and Momentum. Use momentum parameters and learning rate adjusted based on time gradient | - High performance with big data. | - Need to carefully observe the parameters |
| **RMSprop** | * Suitable for problems with sparse data. Divide the learning rate by the magnitude of the nearest gradient calculated. | - Effective for fast convergence problems. | . - Need to choose appropriate learning rate |
| **AdaGrad** | * Adjust learning rate based on gradient history for each parameter. Suitable for optimization algorithms for sparse data problems. | . - Effective with sparse data. | - Effective with sparse data. - Efficiency decreases when training is advanced. |

**Câu 2: Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.**

**\*Continual Learning**

**- Concept?**

Continual Learning, also known as Lifelong learning, is built on the idea of learning continuously about the external world in order to enable the autonomous, incremental development of ever more complex skills and knowledge.

A Continual learning system can be defined asan adaptive algorithm capable of learning from a continuous stream of information, with such information becoming progressively available over time and where the number of tasks to be learned (e.g. membership classes in a classification task) are not predefined.

• Continuous learning deals with the problem of catastrophic forgetting.

• When training on new tasks, neural networks tend to forget information learned from previous task.

**⇒ Reduces model performance on old tasks.**

• This phenomenon is closely related to the stability-plasticity dilemma

• If the model is too stable, it will be difficult to learn new information.

• If the model is too flexible, it will have the problem of forgetting learned information.

**A screenshot of a computer screen

Description automatically generated**

**There are 3 types of Continual Learning:**

**illustration of Continual Learning**

* **Advantages:**

Continual Learning allows the model to learn from the new continuous data without training from the beginning. It will help the model maintain and update the knowledge over time.

Secondly is saving resources, instead of retraining all part odd model with new data, CL helps to save computed resources and time.

Third is application, with the ability to receive and learn from new data, continuous learning is very useful in real applications such as processing user data continuously or sequentially. track changes in the environment.

And finally, is solving the problem of forgetting, reducing the forgetting old-data situation when gaining new knowledge.

* **Limitations:**

Continuous learning in machine learning faces some notable limitations. First, the continuous learning model can easily forget old knowledge, leading to the phenomenon of "catastrophic forgetting" when new knowledge overwrites old knowledge. Second, the continuity of the data can make it easy for the model to overfit to the specific data of each learning stage. In addition, continuous learning from multiple data sets requires the model to be highly multitasking, creating complexity in the training process and requiring a lot of computational resources. Finally, implementing continuous learning also requires special methods to manage continuous learning and integrate new data into the model, adding complexity to real-world implementations.

**\*Test Prodcution**

**-Concept?**

Test Production in machine learning refers to the deployment of a machine learning model into a production or real-world environment, where the model will be used to perform predictions or classify real-world data. Production testing typically involves deploying the model to a production system, testing the model's stability and performance in a real-world environment, and ensuring that the model functions properly and does not cause problems. Unexpected problems occur during use. This often requires collaboration between machine learning teams, software engineers, and other stakeholders to ensure that the model operates efficiently and safely in production environments.

**- Advantages:**

1.Performance Validation: Production testing allows evaluating the actual performance of the model in a production environment, thereby providing valuable information about how the model performs and improving conditions. adjust if necessary.

2. Determine stability: The production test process helps determine the stability of the model under real conditions, thereby helping to predict the reliability of the model when deployed.

3. Detect and fix problems: By deploying the model into a real environment, test production enables early detection of performance-related issues or problems, thereby helping in implementation timely corrective measures.

4. Optimize interaction with the system: The production testing process opens up opportunities to optimize the interaction between the machine learning model and the production system or real application, ensuring integration model in a reliable and efficient way.

**- How to work:**

**1. Prepare real data:** To begin with, real data from the production or application environment—where the model will be used—must be prepared. To get ready for model testing, this data must be gathered and analyzed.

**2. Model deployment:** Along with the system or process it will integrate into, the machine learning model is put into use in the production setting.

**3. Collect real output data:** To provide predictions or classifications, a real-world environment is used to run the machine learning model. The model's output data will be gathered and scrutinized.

**4. Performance measurement**: The collected output data will be used to measure the performance of the model in a real-world environment. Performance metrics such as accuracy, reliability, response time, or any other metric appropriate to the specific application.

**5. Diagnostics and modifications (if necessary):** To guarantee strong performance in production settings, gathered data is compared to performance standards, and the model is modified or enhanced as needed.

References:

<https://edu.soict.ai/wp-content/uploads/2023/05/LinhNV_MML2023.pdf>

<https://wiki.continualai.org/the-continualai-wiki/introduction-to-continual-learning>

<https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/>

<https://www.aiplusinfo.com/blog/what-is-the-adam-optimizer-and-how-is-it-used-in-machine-learning/>

<https://iq.opengenus.org/adagrad/>