VIETNANE GENERAL CONFERDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT

INTRODUCTION TO MACHINE LEARNING

Final Project

Người hướng dẫn: **Lê Anh Cường**

Người thực hiện: **Đặng Xuân Thịnh – 521H0512**

Lớp: **21H50301**

Khoá: **21**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

VIETNANE GENERAL CONFERDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT

INTRODUCTION TO MACHINE LEARNING

Final Project

Instructor: **Giảng viên Lê Anh Cường**

Người thực hiện: **Đặng Xuân Thịnh – 521H0512**

Lớp: **21H50301**

Khoá: **21**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENT

I’d like to give to give sincere thanks and acknowledgement to the IT teachers at Ton Duc Thang University for guiding me and giving me the opportunity to explore and research this interesting topic. Special thanks for my instructor Le Anh Cuong for the teaching in the Introduction to Machine learning class, which is especially helpful for this assignment.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Giảng viên Lê Anh Cường Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 24 tháng 12 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Đặng Xuân Thịnh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày 24 tháng 12 năm 2023
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày 24 tháng 12 năm 2023
(kí và ghi họ tên)

Summary

This project summarizes and go over optimization techniques commonly used in machine learning such as learning rate tunning, gradient descent techniques. As well as explain the concept continual learning and the things to take into consideration when putting a machine learning model into production

Table of Contents

CHƯƠNG 1 - Optimizer techniques in machine learning

1.1 What is optimization in machine learning

In machine learning, cost function is a mathematical function that measures the differences between the predicted values of a model and the actual values of the target variable. Optimization is the process of adjusting hyperparameters to reduce the cost function and ultimately leads to a faster and more accurate model.

Hyperparameters are the external configuration settings for a model that are set prior to the learning process, some of which are Learning rate, number of clusters, batch size... These parameters significantly impact the performance of the model and are not updated during training. Therefore, tunning these hyperparameters properly is key to achieving optimal model performance

1.2 Feature scaling

Feature scaling is a preprocessing step in the learning process that involves transforming the input features of the dataset to a standardized or normalized range. The goal is to ensure that all features contribute equally to the process and there's no feature dominating the learning process.

1.2.1 Normalization

Normalization is the simplest yet effective method in feature scaling. Usually features scaled to a specific range, typically [0, 1].

The formula for a scaled feature is:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

1.2.2 Standardization

Also known as Z-score normalization. It involves transforming the features so that they have a mean of 0 and a standard deviation of 1. This makes the features more comparable and help certain machine learning algorithm converge faster and perform faster.

The standardization formula for a feature is:

$$X_{new} = \frac{X - mean(X)}{std(X)}$$

Where:

$mean(X)$ is the mean of feature X

$std(X)$ is the standard deviation of X

1.3 Learning rate

Learning rate is a hyperparameter that determines the step size at each iteration during the optimization process of training a model. In gradient-based optimization algorithms such as gradient descent, choosing the right learning rate is crucial to reach convergence. A too high learning rate can make the learning jump over minima and a too low learning rate will make it a lot longer to converge or get stuck in a local minima.

Usually, we select the learning rate through trial and error or based on experience. But there are lots of methods that can help choosing a desirable learning rate. One method is to use learning rate annealing, which involves lowering the learning by a fixed factor after a certain amount of iterations until a desired performance is reached. Another way is to start with a small learning rate and gradually increase it during training. Then monitor the loss of relevant metrics as the learning rate increase, the optimal learning rate is typically located where the loss decreases the fastest

1.4 Gradient descent

Gradient descent is a commonly used optimization technique in machine learning that refers to the process of adjusting the parameters of a model to find the minimum of the cost function. The basic idea is to iteratively move towards the minimum of the cost function by updating the parameters based on the gradient (or the derivative) of the cost function with respect to each of those parameters

In linear regression problems, the cost function is typically the mean square error (MSE) of the model with the formula being:

$$L = \frac{1}{2n} * \sum_{i=1}^{i=n} \left(y_{predict} - y_{actual}\right)^2$$

Where:

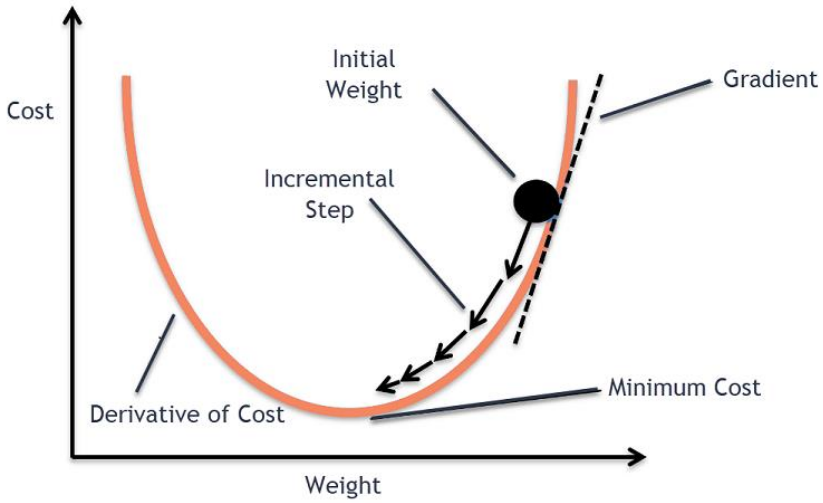
- n is the number of data points
- $y_{predict}$ is the predicted value
- y_{actual} is the actual output value

The algorithm consists of these steps:

- Step 1: Start with an initialization set of parameter values
- Step 2: Compute the gradient of the cost function with respect to each parameter
- Step 3: Update the parameters in the opposite direction of the gradient to reduce the cost function in proportional to the learning rate

$$w_i = w_i - (learning\_rate)\frac{\partial L}{\partial w_i}$$

- Step 4: Repeat step 2 and 3 until convergence or a predefined number of iterations



Choosing the correct learning is essential in gradient descent as it ensures that the algorithm converges in a reasonable time as mentioned above.

Gradient descent is a fundamental optimization algorithm and there are three types of gradient descent algorithm

1.4.1 Batch gradient descent

In batch gradient descent, the entire training dataset is used to compute the gradient of the cost function with respect to model parameters. It utilizes the full training dataset which

provides accurate estimate of the training set but is computationally expensive for large datasets

The main advantage of batch gradient descent is that it converges to the global minimum of the cost function when it exists as the entire training set is involved in the process. However, this makes the algorithm more computationally expensive especially with large datasets. To address this issue, optimization techniques such as preprocessing, normalization, early stopping, regularization,… should be taken into consideration when implementing.

Due to the nature of the algorithm, the learning rate plays a crucial role in the performance of batch gradient descent. As mentioned above, a too small learning will lead to slow convergence and a too large learning rate can cause the algorithm to diverge.

1.4.2 Stochastic gradient descent

Stochastic gradient descent: The training dataset is randomly shuffled and a single randomly chosen training example is used to update the parameters in each iteration. This variant is computationally more efficient than batch gradient descent but frequent updates can result in noisy gradients which can lead to divergence or oscillations.

This algorithm is a common choice for training large neural networks model due to its scalability and efficiency. It also has better convergence than batch gradient descent and the noise introduced can also help escape local minima. This is particularly more efficient when paired with adaptive learning rate methods such as Adam, RMSProp, AdaGrad which help adjusting the learning rate for each parameter respectively, leading to improved performance and faster convergence.

Here are steps to perform stochastic gradient descent:

- Step 1: Initialize the model with random parameters
- Step 2: Shuffle the training set to introduce randomness
- Step 3: Iterate over each batch and compute cost, compute gradient and then update the parameters
- Step 4: Repeat step 2 and 3 until convergence or for a predefined number of iterations

1.4.3 Mini-batch gradient descent

A small random subset of the training dataset, referred to as a mini-batch, is used for each update. It's the most common used variant used in deep learning as it gives a balance between stochastic gradient descent and batch gradient descent.

This approach combines the many benefits of batch gradient descent and stochastic gradient descent. Mini-batch gradient descent scales well with large datasets, requires less memory and is computationally more efficient than batch gradient descent. When compared to stochastic gradient descent, although it introduces less noise and variance, it requires more memory in comparison.

In this algorithm, the batch size, or the size of the mini-batch, is a hyperparameter that needs tunning to achieve good performance. The optimal batch-size can vary depending on the type of problem, therefore experimentation needs to be performed regularly to choose the right batch size.

Here are steps to perform mini-batch gradient descent:

- Step 1: Initialize the model with random parameters
- Step 2: Randomly shuffle the training set and divide it into small, equally sized mini-batches
- Step 3: Iterate over each min-batch and compute cost, calculate gradient and update parameters
- Step 4: Repeat step 2 and 3 until convergence or for a predefined number of iterations

1.4.4 A comparison between gradient descent techniques

Choosing the correct gradient descent algorithm depends on the specific problem and the resources available. Here's an overview of the differences between batch gradient descent, stochastic gradient descent and mini-batch gradient descent

Feature	Batch gradient descent	Stochastic gradient descent	Mini-batch gradient descent
Data processing	Uses the entire training set for each iteration	Uses one data point at a time in each iteration	Uses a small subset of training set in each iteration
Memory usage	High	Low	Medium
Convergence speed	Slow	High	Medium
Scalability	Low	High	High
Escape local minima	Longer	Faster	Medium
Advantage	Easy to implement, stable, reliable	Good scalability, faster convergence, easier to escape local minima	Good scalability, strikes a good balance
Disadvantage	Slow, computationally expensive, requires a lot of memory	Introduces noise, can lead to divergence	Requires batch-size tunning

CHƯƠNG 2 - Continual learning

Also known as Incremental learning or Lifelong learning, Continual learning is a machine learning paradigm that emphasizes on the ability of a model to learn and adapt with new data overtime from a sequence of tasks sequentially while retaining obtained knowledge from previous experience. The ultimate goal is to develop a model that can learn and generalize effectively over a large range of input data throughout its lifetime.

This is particularly important when dealing with real world problems. Unlike when learning in school, we hardly ever deal with static data in the real world, therefore the ability

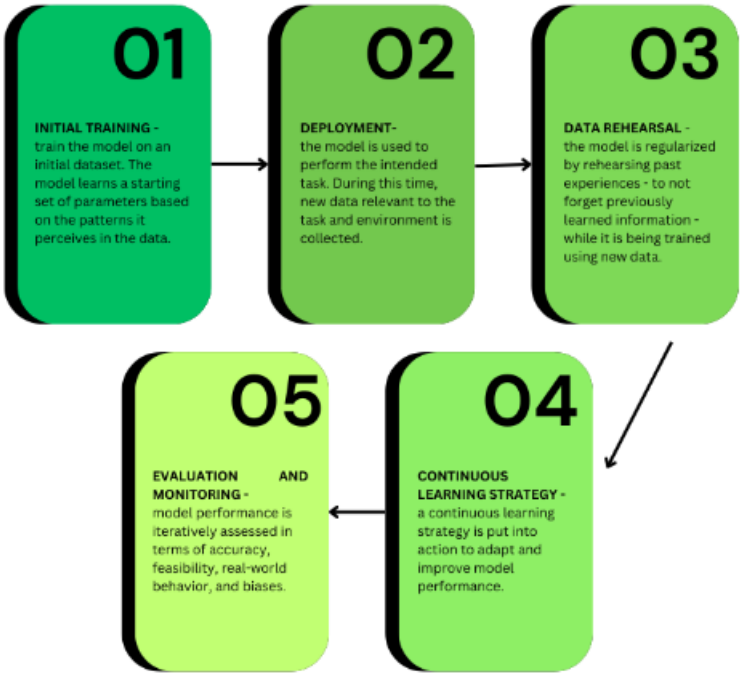


Figure 1. Continual learning process

for models to retain old knowledge is essential to building a ML-powered product.

2.1 Types of continual learning

There are three common types of continual learning scenarios: Task continual learning, Data stream continual learning and Domain continual learning.

2.1.1 Task continual training

In task continual learning, the model is exposed to a sequence of tasks over time. The model then has to solve each task while retaining knowledge from previous ones. Some example scenarios of this include classifying handwritten digits (MNIST dataset), recognizing objects in images (CIFAR-10 dataset).

Challenges:

- Catastrophic forgetting: The model might forget information about previous tasks in the process of learning new ones
- Interference: Learning new tasks might negatively interfere with the performance on previous ones

2.1.2 Data stream continual learning

In data stream continual learning, the model is exposed to a continuous stream of data. The model then must adapt to the changing patterns, concepts, or distributions within the data stream. Real world examples of this scenario can include processing real-time data from financial transactions, social media, monitoring network traffic for anomalies

Challenges:

- Concept drift: The statistical properties of the data can change randomly over time
- Resource management: Managing computational resources and memory over time is essential

2.1.3 Domain continual learning

In data stream continual learning, the model will perform in different domains or environments over time witch each domain having different characteristics and conditions, the model then have to adapt to variations in the input space. Changes in domain and environment

can occur in real world scenarios like speech recognition systems adapting to different accents, automatic vehicles adapting to different driving conditions (highway, off-road, urban,…)

Challenges:

- Domain shift: Differences in the statistical properties of different domains
- Transfer learning: Using knowledge obtained from one domain to improve performance in another one

2.2 Catastrophic forgetting

Catastrophic forgetting is a phenomenon where a model's acquired knowledge is interfered with or erased during the process of learning new information. This is a well-known problem and also the main problem faced in the continual learning field.

In neural networks, there's a common challenge when it comes to continual learning which is the stability-plasticity dilemma. In short, a model has to be plastic enough to learn new information and stable enough to retain acquired information. A balance in this is especially challenging in neural networks since neural networks are high plastic and can quickly adapt to new information, this leads to the overpowering of stability

2.3 Techniques used in continual learning

To combat the main issue in continual learning, catastrophic forgetting, a number of techniques and strategies are used to help the model retain old knowledge while learning new information

2.3.1 Rehearsal or replay

When the model learns any data, the replay samples are stored and used while the model is being trained on new data to help the model revisit and reinforce knowledge of earlier tasks.

This is a commonly used and effect strategy in continual learning, but it comes with some challenges and trade-offs, two of which are computational cost and storage requirements. Specifically, maintaining a replay buffer for rehearsal comes with a computational cost especially when the dataset is large, and storing past experience require careful memory management according to the size of the replay

2.3.2 Regularization

In continual learning, regularization techniques can also be used to add a penalty term to the cost function to penalize large changes in important parameters which helps to model preserve old knowledge and prevent them from being overwritten by new information.

A well-known regularization technique used in neural network to prevent catastrophic forgetting is Elastic Weight Consolidation (EWC). The object function for training a model on a new task consists of the task loss function and the EWC regularization term:

$$J(\theta) = Loss(h_\theta(x), y) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta_i^*)^2$$

Where:

- $J(\theta)$: The objective function
- $Loss(h_\theta(x), y)$: loss of the current task
- λ: regularization strength
- F_i: Fisher information matrix for parameter θ_i, representing the sensitivity of the loss with respect to θ_i
- θ_i^*: Parameter values at the end on training on the previous task

2.3.3 Knowledge distillation

Knowledge distillation is used in continual learning to transfer previous obtained knowledge from a trained model (teacher) to a new model (student). The goal is to use the information acquired by the teacher model to enhance the learning process of the student model, thereby reducing the effect of catastrophic forgetting.

This serves as a form of regularization, helping the student model retain information from previous tasks and help it generalizes better on new tasks

CHƯƠNG 3 - Test production

After building our machine learning model, in order to take it to production, there are sets of practices and considerations to ensure that the model perform as expected in a real-world experience. This step is essential to making the model reliable, performant and work as intended

3.1 Evaluation metrics

After training a model, one essential step to perform is model evaluation. It involves comparing the model's prediction to the true labels in a dataset to gain insights on how well the model generalizes. The are lots of different evaluation metrics and it's important to evaluate our model on different metrics to gauge specific aspects of the model's performance.

Some evaluation metrics for classification models:

- Confusion matrix: Shows the counts of true positive, true negative, false positive, false negative
- Accuracy: The proportion of accurate predicted instances out of the total instances
- Recall: The ability for the model to capture all positive instances
- Precision: The ability for the model to correctly identify positive instances among all instances that are predicted as positive
- F1 score: The harmonic mean of precision and recall

Some evaluation metrics for regression models:

- Mean square error (MSE): The average squared difference between the predicted values and the true values
- Mean absolute error (MAE): The average absolute difference between the predicted values and the true values

3.2 Cross validation

Cross validation helps evaluate the performance of a model on unseen data. This ensures that our model has picked up the correct patterns from the training data without getting too much noise. The process involves dividing the available data into multiple subsets and use it to train the model multiple times. After training, the results of each validation are taken to estimate the performance and reliability of the model. This also greatly helps preventing overfitting, a common issue faced in machine learning that makes the model too accustomed to the training data and perform poorly on new data

3.3 Unit testing

Just like in any software, unit testing is fundamental in building reliable machine learnings and pipelines. It involves testing components or units of the model to ensure that they perform their functions correctly.

Some important components to test during unity testing includes:

- Data preprocessing: Test functions or classes that handle data cleaning, scaling, transforming, categorical encoding,…
- Feature engineering: Test functions or classes that creates, removes or transforms features
- Model training: Test functions or classes that handle the training of the model, making sure that the model is learning from the training data and is giving expected results
- Prediction: Test functions or classes that is responsible for taking the trained model to make predictions, ensuring that the predicted result is reasonable and expected

REFERENCES

English

1. Serokell. (2020, December 1). ML Optimization Methods and Techniques. [Blog post]. Retrieved from https://serokell.io/blog/ml-optimization
2. Secherla, S. (2023, November 1). Understanding Optimization Algorithms in Machine Learning. [Blog post]. Towards Data Science. https://towardsdatascience.com/understanding-optimization-algorithms-in-machine-learning-edfdb4df766b
3. Full Stack Deep Learning. (2022, November 10). Lecture 6: Continual Learning. [Webpage]. Retrive from https://fullstackdeeplearning.com/course/2022/lecture-6-continual-learning/
4. DataCamp. (2023, March 15). What is Continuous Learning? Revolutionizing Machine Learning & Adaptability. [Blog post]. Retrieved from https://www.datacamp.com/blog/what-is-continuous-learning