Vietnam General Confederation of Labor

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT

MACHINE LEARNING

Instructor: **Mr. LE ANH CUONG**

Student: **DUONG NGOC BAO NHI – 521K0143**

Class :**21K50301**

Year **: 2023-2024**

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: **DUONG NGOC BAO NHI – 521K0143**

Class :**21K50301**

Year **: 2023-2024**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENT

I would like to thank the lecturer Le Anh Cuong. During the process of studying and learning about the subject, I have learned a lot of knowledge as well as many interesting things from your lessons. He has helped us expand new and in-depth knowledge about this subject. During the learning process, we are still weak, but from the knowledge learned, I have gained important knowledge about specialized skills as well as knowledge to explore and solve problems.

In the learning process, shortcomings are inevitable. I hope to receive feedback from the teacher to make my report more complete.

In addition, I would like to thank the school for creating conditions for us to have a good learning environment so that I can gain more of this knowledge.

We wish you health, happiness and success in your teaching career.

Ho Chi Minh city, 14th December, 2023

Author

(Sign and write full name)

Duong Ngoc Bao Nhi

CONFIRMATION AND ASSESSMENT SECTION

Instructor confirmation section

Ho Chi Minh 14 December, 2023

Duong Ngoc Bao Nhi

Evaluation section for grading instructor

Ho Chi Minh, 14 December 2023

Duong Ngoc Bao Nhi

THE PROJECT IS COMPLETED

AT TON DUC THANG UNIVERSITY

Our team would like to assure that this is our own research project and is under the scientific guidance of Lê Anh Cường Teacher. 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 report 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, our team will take full responsibility for the content of our IT Project Report 2. Ton Duc Thang University is not involved in copyright violations caused by us during the implementation process (if any).

Ho Chi Minh city, 14th December, 2023

Author

(Sign and write full name)

Duong Ngoc Bao Nhi

TABLE OF CONTENTS

CHAPTER 1 – Optimizer

1.1 Introduction

1.2 Optimizer's optimization algorithms

When you’re training a neural network (which is like teaching a computer to make specific types of decisions), you use something called an optimizer. This is like a set of instructions for the computer.

The optimizer’s job is to adjust different parts of the neural network (like the weights and learning rate) to make the network better at its job. The ‘weights’ are the importance given to different inputs, and the ‘learning rate’ is how quickly the network changes its mind based on new data.

The goal of the optimizer is to reduce mistakes (or ‘losses’) and make the network’s decisions as accurate as possible. So, when you’re using an optimizer, you’re basically fine-tuning your neural network to make it the best decision-maker it can be!

1.2.1 Gradient Descent (GD)

Gradient Descent is a basic but important tool used to train neural networks. It’s like a guide that helps adjust the network’s parameters to reduce mistakes. It’s used in predicting values (linear regression) and sorting data (classification).

Backpropagation is part of this process. It’s like a feedback system that passes information backwards through the network to help adjust its parameters and improve accuracy.

1.2.1.1Algorithm: $\theta=\theta-\alpha\cdot\nabla J(\theta)$

1.2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a version of Gradient Descent. Instead of updating the model’s parameters (like weights) once per cycle (as in regular Gradient Descent), SGD updates them after each training example. So, if you have 1000 examples, SGD will update the parameters 1000 times in one cycle. This makes SGD update the model more frequently.

1.2.2.1 Algorithm:

$$\theta=\theta-\alpha\cdot\nabla J(\theta;x(i);y(i))$$

, where {x(i) ,y(i)} are the training examples.

As the model parameters are frequently updated parameters have high variance and fluctuations in loss functions at different intensities.

1.2.3 Mini-Batch Gradient Descent

Batch Gradient Descent is a type of Gradient Descent that's even better. It improves the model after each batch of data. Think of it like this: if your data is a loaf of bread, Batch Gradient Descent updates the model after each slice, not just at the end!

1.2.3.1 Algorithm:

$$\theta=\theta-\alpha\cdot\nabla J(\theta;\ B(i))$$

, where {B(i)} are the batches of training examples.

1.2.4 Momentum

Momentum is a technique used to help SGD work better. It's like a car using a GPS to stay on the right path and avoid unnecessary turns. It helps the model learn faster and smoother. The 'γ' is just a setting we adjust to control how much momentum to use.

1.2.4.1 Algorithm:

$$V(t)=\gamma V(t-1)+\alpha.\nabla J(\theta)$$

Now, the weight are updated by $\theta=\theta-V(t)$**.**

The momentum term γ **is usually set to 0.9 or a similar value.**

1.2.5 Nesterov Accelerated Gradient

Momentum is like a speed boost for learning, but too much can cause overshooting. Nesterov Accelerated Gradient (NAG) fixes this by predicting future positions and making adjustments based on that. It uses a formula to calculate the cost of the future position, not just the current one, helping the model learn more efficiently.

1.2.5.1 Algorithm:

$$V(t)=\gamma V(t-1)+\alpha.\ \nabla J(\ \theta-\gamma V(t-1)\ )$$

and then update the parameters using $\theta=\theta-V(t)$.

1.2.6 Root Mean Square Propagation (RMSProp)

RMSProp is like a personal trainer for a neural network. It adjusts the learning speed for each part of the network. It does this by remembering past mistakes (gradients), but gives more attention to recent ones. This helps the network learn more efficiently and forget less important details

1.2.7

1.2.7 Adaptive Moment Estimation (Adam)

Adam is a method that helps make learning more efficient. It’s like a car with brakes - it slows down to avoid missing important details. It keeps track of average past gradients (directions of change) and squared gradients (magnitude of change).

M(t) and V(t) are the average and variance of these gradients. We use these to adjust the learning process. The parameters β1, β2, and ‘ϵ’ are just settings we tweak to control how Adam works.

1.2.7.1 How does it work?

Calculate the gradient and square of the gradient: Adam calculates the gradient of the loss function according to each parameter of the model and stores the square of this gradient.

Adjusting the magnitude of the gradient: Similar to RMSprop, Adam uses a weighted average of the squares of the gradient, but it also uses a weighted average of the formal gradient.

Updating model weights: Adam calculates the tuning value for each model parameter based on two weighted averages: one for the gradient and one for the gradient squared.

1.2.8 Summary:

gorithm	Advantages	Disadvantages
cent (GD)	utation plement derstand	t local minima , we adjust 'weights' based on set. If the dataset is huge, this l take a very long time to find the rge memory to calculate gradient set
adient Descent	appen often, so learning is memory because it doesn't nber all past mistakes l new	nce in model parameters even after achieving global same convergence as gradient to slowly reduce the value of
Gradient Descent	update the model parameters ess variance edium amount of memory	
	e oscillations and high e parameters faster than gradient descent	hyper-parameter is added which lected manually and accurately
elerated	niss the local minima inima's are occurring	yperparameter needs to be ally
	with spares gradients, unstable	dditional hyperparameter tuning

	d is too fast and converges anishing learning rate, high	onally costly

CHAPTER 2 – Continual Learning and Test Production

A. Continual Learning

Continual Learning is like a brain that never stops learning. It keeps adding new knowledge to the existing model without forgetting what it already knows. It's used when data keeps coming in, like stock trends or user profiles.

Unlike traditional learning, which uses a fixed set of data, Continual Learning adapts and improves over time. It aims to learn a series of tasks, each with its own data. The goal is to get better or stay the same at all tasks, even as new ones come in.

One big challenge is 'catastrophic forgetting', where the model gets worse at a task after learning a new one. This can be mathematically described as:

Here, m^t and v^t are the first and second moment estimates (mean and variance) of the gradients, β_1^t and β_2^t are the decay rates, α is the learning rate, *w* are the parameters of the model, and ε is a small smoothing term to avoid division by zero.

To avoid catastrophic forgetting, we use techniques like regularization (adding rules to prevent overfitting) and memory-augmented networks (using memory to store important information).

Continual Learning (CL) is vital in various fields. It’s key in these fields to learn from new information while remembering past data.

Healthcare: Continual Learning aids in diagnostics, predictive analytics, and real-time patient management.

Recommender Systems: It’s used in systems like Netflix and Amazon to adapt to user interactions.

Data Streams or Big Data: It’s used in areas like stock prediction and user profiling where data is continuously updated.

Personalization and On-device Learning: It helps in tailoring user experiences and learning directly on devices.

B. Test Production:

Test Production in Machine Learning is like a final quality check for a model when it’s put to work in the real world. It’s similar to how a car is tested for safety and performance before it hits the road.

Here are some benefits of model testing:

Eliminating Bugs and Errors: It’s like a cleaner removing bugs and errors in the data steps, learning algorithm, or prediction pipeline.

Checking Data and Model Integrity: It’s like a health inspector ensuring that the data is accurate and the model works as expected.

Ensuring Robustness of ML Model: It’s like a stress-test checking the model’s performance on unseen data and under different conditions.

Key step in testing production:

- Validating input data
- Monitoring Model Performance
- Detecting Model and Data Drift
- Testing Integration Between Pipeline Components
- Model Retraining
- A/B Testing
- Testing Quality of Live Model on Served Data