TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

(Phần làm cá nhân)

Môn Nhập Môn Học Máy

Machine Learning

Người hướng dẫn: ***PGS.TS Lê Anh Cường***
Người thực hiện: ***Nguyễn Trọng Huy– 51900507***
Lớp ***: 19050402***
Khoá ***: 23***

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

(Phần làm cá nhân)

Môn Nhập Môn Học Máy

Machine Learning

Người hướng dẫn: ***PGS.TS Lê Anh Cường***
Người thực hiện: ***Nguyễn Trọng Huy– 51900507***
Lớp ***: 19050402***
Khoá ***: 23***

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy Lê Anh Cường đã cung cấp đề bài nghiên cứu và báo cáo cho chúng em. Đề bài mang tính thực tiễn và thách thức, giúp chúng em được tìm hiểu, nghiên cứu và nâng cao hiểu biết về lĩnh vực học máy.

Thầy đã tạo điều kiện và không gian để chúng em được trau dồi, phát triển kỹ năng phân tích, giải quyết bài toán, phát triển các mô hình, tránh quá khớp và cải tiến mô hình. Qua đó, chúng em có cơ hội rèn luyện tư duy logic, khả năng giải quyết vấn đề và làm việc nhóm.

Một lần nữa, em xin chân thành cảm ơn thầy đã tạo điều kiện cho chúng em trưởng thành. Kính mong thầy tiếp tục động viên, hướng dẫn và chỉ bảo để chúng em ngày càng vững vàng.

Kính chúc thầy dồi dào sức khỏe, hạnh phúc và thành công!

EM XIN CHÂN THÀNH CẢM ƠN!

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 và được sự hướng dẫn của PGS.TS 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.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào 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 tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm
Tác giả
(ký tên và ghi rõ họ tên)

Nguyễn Trọng Huy

Phần đánh giá của GV chấm bài

_____
_____
_____
_____
_____
_____

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

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CHƯƠNG I: MỞ ĐẦU

1.1 Lý do chọn đề tài

Môn học Nhập môn Học máy rất quan trọng trong lĩnh vực trí tuệ nhân tạo, em muốn tìm hiểu sâu hơn một số vấn đề trong học máy.
Với ưu thế của việc tìm hiểu sâu về hai vấn đề này, em hy vọng có thể vận dụng linh hoạt và hiệu quả hơn trong thực tiễn.
Đề tài phù hợp với hướng nghiên cứu của em trong tương lai.

1.2 Mục tiêu chọn đề tài

Tìm hiểu chi tiết cơ chế hoạt động và ưu nhược điểm của một số Optimizer phổ biến.
Đánh giá tính phù hợp và hiệu quả ứng dụng của từng loại Optimizer trong huấn luyện các mô hình học máy.
Giới thiệu khái niệm, cơ chế hoạt động và các phương pháp tiếp cận Continual Learning.
Trình bày các kỹ thuật Test Production và vai trò của nó trong quá trình xây dựng hệ thống học máy.

1.3 Phạm vi nghiên cứu

Tập trung vào một số Optimizer phổ biến Gradient descent, Momentum, RMSProp, Adam.

Continual Learning và Test Production được nghiên cứu ở mức độ tổng quan.

Phạm vi báo cáo nằm trong khuôn khổ môn học Nhập môn Học máy.

CHƯƠNG II: NỘI DUNG

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

Giới thiệu chung về Optimizer
Optimizer là một thành phần quan trọng của hệ thống quản lý cơ sở dữ liệu (DBMS) hoặc mạng neural, có nhiệm vụ phân tích các truy vấn hoặc các hàm mất mát và xác định cơ chế thực thi hiệu quả nhất. Optimizer tạo ra một hoặc nhiều kế hoạch truy vấn hoặc cập nhật trọng số cho mỗi truy vấn hoặc hàm mất mát, mỗi kế hoạch đều có thể là một cơ chế được sử dụng để chạy truy vấn hoặc hàm mất mát.

Một số Optimizer phổ biến

- Gradient descent (GD): Là một thuật toán tối ưu hóa dựa trên đạo hàm, có ý tưởng là cập nhật nghiệm theo hướng ngược với đạo hàm của hàm mất mát, để tìm

điểm cực tiểu cục bộ hoặc toàn cục. Công thức cập nhật nghiệm của GD là:

$$x_{\text{new}} = x_{\text{old}} - \alpha \nabla f(x)$$

Trong đó, ***α*** *là tốc độ học, ∇ là đạo hàm của hàm mất mát* ***f(x)*** *tại điểm* ***x****. GD có ưu điểm là đơn giản và dễ hiểu, nhưng có nhược điểm là tốc độ hội tụ chậm, phụ thuộc nhiều vào tốc độ học và điểm khởi tạo, và có thể bị kẹt ở điểm yên ngựa[3].*

- Momentum: Là một biến thể của GD, có ý tưởng là sử dụng một lượng động lượng để tăng tốc độ hội tụ và tránh bị kẹt ở các điểm yên ngựa[4]. Công thức cập nhật nghiệm của Momentum là:

$$v_{\text{new}} = \beta v_{\text{old}} - \alpha \nabla f(x)$$

$$x_{\text{new}} = x_{\text{old}} + v_{\text{new}}$$

Trong đó, ***v*** *là động lượng,β là hệ số giảm động lượng, các tham số còn lại giống như GD. Momentum có ưu điểm là tăng tốc độ hội tụ, giảm dao động và tránh bị kẹt ở các điểm yên ngựa, nhưng có nhược điểm là phải chọn thêm một tham số β và có thể bị vượt quá điểm cực tiểu.*

- RMSProp: Là một biến thể của GD, có ý tưởng là điều chỉnh tốc độ học theo từng tham số, dựa trên trung bình bình phương của đạo hàm trong quá khứ[5]. Công thức cập nhật nghiệm của RMSProp là:

$$s_{\text{new}} = \gamma s_{\text{old}} + (1-\gamma) \nabla f(x)^2$$

$$x_{\text{new}} = x_{\text{old}} - \frac{\alpha}{\sqrt{s_{\text{new}} + \epsilon}} \nabla f(x)$$

Trong đó, s là trung bình bình phương của đạo hàm, ***γ*** *là hệ số giảm trọng số, ϵ là một số nhỏ để tránh chia cho 0, các tham số còn lại giống như GD. RMSProp có ưu điểm là giảm dao động, thích ứng tốc độ học theo từng tham số, và hội tụ nhanh hơn GD, nhưng có nhược điểm là phải chọn thêm một tham số* ***γ*** *và có thể bị quên đạo hàm trong quá khứ.*

- Adam: Là một biến thể của GD, có ý tưởng là kết hợp Momentum và RMSProp, để tận dụng ưu điểm của cả hai. Công thức cập nhật nghiệm của Adam là:

$$m_{\text{new}} = \beta_1 m_{\text{old}} + (1-\beta_1)\nabla f(x)$$

$$s_{\text{new}} = \beta_2 s_{\text{old}} + (1-\beta_2)\nabla f(x)^2$$

$$\hat{m}_{\text{new}} = \frac{m_{\text{new}}}{1-\beta_1^t}$$

$$\hat{s}_{\text{new}} = \frac{s_{\text{new}}}{1-\beta_2^t}$$

$$x_{\text{new}} = x_{\text{old}} - \frac{\alpha}{\sqrt{\hat{s}_{\text{new}} + \epsilon}}\hat{m}_{\text{new}}$$

*Trong đó,**m** là trung bình động của đạo hàm,* s*là trung bình bình phương của đạo hàm, β1 và β2 là hai hệ số giảm trọng số, **t** là số vòng lặp, **hat{m}** và **hat{s}** là các ước lượng hiệu chỉnh của **m** và* s*, các tham số còn lại giống như GD. Adam có ưu điểm là kết hợp được Momentum và RMSProp, thích ứng tốc độ học theo từng tham số, và hội tụ nhanh và ổn định, nhưng có nhược điểm là phải chọn thêm hai tham sốβ1 và β2 và có thể bị sai lệch lớn.*

Một số Nguyên tắc chung để chọn Optimizer
- Nếu bài toán đơn giản và hàm mất mát lồi, có thể sử dụng GD hoặc các biến thể của nó.
- Nếu bài toán phức tạp và hàm mất mát không lồi, có thể sử dụng Momentum, RMSProp hoặc Adam.
- Nếu bài toán có nhiều tham số và dữ liệu lớn, có thể sử dụng Adam vì nó có thể thích ứng tốc độ học theo từng tham số và hội tụ nhanh.
- Nếu bài toán có nhiều đạo hàm bằng 0 hoặc gần 0, có thể sử dụng RMSProp vì nó có thể giảm tốc độ học khi đạo hàm nhỏ và tăng tốc độ học khi đạo hàm lớn.

Tìm hiểu Continual Learning và Test Production

Giới thiệu Continual Learning

Continual Learning là khả năng học liên tục từ một luồng thông tin không ngừng thay đổi, để phát triển các kỹ năng và kiến thức ngày càng phức tạp và thích ứng với môi trường. Đây là một trong những yếu tố quan trọng để xây dựng các hệ thống trí tuệ nhân tạo (AI) có thể tự học hỏi và cải thiện bản thân.

Continual Learning đòi hỏi một thuật toán học có thể tiếp nhận, cập nhật, tích lũy và sử dụng kiến thức trong suốt quá trình học, mà không bị quên hoặc xáo trộn bởi các kiến thức mới. Đây là một thách thức lớn cho các mô hình học sâu (Deep Learning), vì chúng thường bị hiện tượng quên địa phương (catastrophic forgetting), tức là khi học một nhiệm vụ mới, chúng sẽ mất đi hiệu quả của các nhiệm vụ cũ.

Các phương pháp tiếp cận Continual Learning

Có nhiều phương pháp tiếp cận Continual Learning, có thể phân loại theo ba hướng chính2:

- Phương pháp dựa trên bộ nhớ (memory-based): Sử dụng một bộ nhớ phụ để lưu trữ một số mẫu dữ liệu quan trọng của các nhiệm vụ cũ, và sử dụng chúng để hỗ trợ quá trình học nhiệm vụ mới.

- Phương pháp dựa trên kiến trúc (architecture-based): Thay đổi kiến trúc của mô hình để có thể thêm hoặc xoá các tầng hoặc các nơ-ron khi học các nhiệm vụ mới, để tránh ghi đè lên các tham số quan trọng của các nhiệm vụ cũ.

- Phương pháp dựa trên thông tin (information-based): Sử dụng các kỹ thuật như biểu diễn đa nhiệm (multi-task representation), học chuyển tiếp (transfer learning), học miền (domain adaptation), để tạo ra các biểu diễn có tính khái quát hóa cao, có thể sử dụng cho nhiều nhiệm vụ khác nhau.

Giới thiệu Test Production

Test Production là quá trình kiểm tra chất lượng và hiệu năng của sản phẩm trong môi trường thực tế, sau khi đã hoàn thành các bước kiểm tra trước khi đưa vào sản xuất. Test Production có thể giúp phát hiện và khắc phục các lỗi hoặc vấn đề mà các bước kiểm tra trước đó không thể nhận biết được, do sự khác biệt giữa môi trường thử nghiệm và môi trường thực tế.

Test Production cũng có thể giúp thu thập phản hồi từ người dùng thực tế, để cải thiện trải nghiệm người dùng và đánh giá hiệu quả của sản phẩm. Test Production là một bước quan trọng trong quy trình phát triển phần mềm, đặc biệt là khi áp dụng các phương pháp như phát triển liên tục (Continuous Development) hay phát triển linh hoạt (Agile Development).

Các phương pháp tiếp cận Test Production

- Phương pháp kiểm tra hiệu năng (performance testing): Nhằm đo lường và đánh giá các chỉ số về hiệu năng của sản phẩm, như thời gian phản hồi, khả năng mở

rộng, độ tin cậy, độ ổn định, v.v. Ví dụ: Giám sát và chẩn đoán (Monitoring and Diagnostics), Kiểm tra tải (Load Testing), Kiểm tra đột biến (Spike Testing).

- Phương pháp kiểm tra so sánh (comparative testing): Nhằm so sánh và đánh giá các phiên bản khác nhau của sản phẩm, để xem phiên bản nào có hiệu quả hơn, hoặc có sự khác biệt nào đáng kể giữa chúng. Ví dụ: Kiểm tra A/B (A/B Testing), Phát hành tăng dần (Incremental Release), Kiểm tra tích hợp (Integration Testing), Theo dõi phản hồi (Feedback Tracking).

Phân tích chương trình

Mục tiêu

Mục tiêu của báo cáo này là để giới thiệu về mô hình hồi quy tuyến tính (linear regression), cách xây dựng và huấn luyện mô hình này trên Python, cách thực hiện học liên tục (continual learning) khi có dữ liệu mới, và cách triển khai mô hình vào một web service sử dụng Flask.

Nội dung

Mô hình hồi quy tuyến tính

Mô hình hồi quy tuyến tính là một mô hình học máy (machine learning) có nhiệm vụ là dự đoán một biến phụ thuộc (dependent variable) y dựa trên một hoặc nhiều biến độc lập (independent variables) x. Mô hình này giả định rằng có một mối quan hệ tuyến tính giữa y và x, tức là y có thể được biểu diễn bằng một tổng trọng số của x cộng với một hệ số chặn (intercept). Công thức của mô hình hồi quy tuyến tính là:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon$$

Trong đó, β0,β1,...,βn là các tham số của mô hình, cần được học từ dữ liệu, và ϵ là nhiễu ngẫu nhiên. Một cách trực quan, mô hình hồi quy tuyến tính có thể được hiểu như sau:

- *Nếu chỉ có một biến độc lập x, mô hình hồi quy tuyến tính sẽ tìm một đường thẳng (line) sao cho khoảng cách từ các điểm dữ liệu đến đường thẳng là nhỏ nhất.*
- *Nếu có hai biến độc lập x1 và x2, mô hình hồi quy tuyến tính sẽ tìm một mặt phẳng (plane) sao cho khoảng cách từ các điểm dữ liệu đến mặt phẳng là nhỏ nhất.*

- *Nếu có nhiều hơn hai biến độc lập, mô hình hồi quy tuyến tính sẽ tìm một siêu mặt phẳng (hyperplane) sao cho khoảng cách từ các điểm dữ liệu đến siêu mặt phẳng là nhỏ nhất.*

Một trong những thuật toán phổ biến để học các tham số của mô hình hồi quy tuyến tính là gradient descent (GD). Thuật toán này có ý tưởng là cập nhật nghiệm theo hướng ngược với đạo hàm của hàm mất mát (loss function), để tìm điểm cực tiểu cục bộ hoặc toàn cục. Hàm mất mát thường được sử dụng cho mô hình hồi quy tuyến tính là sai số bình phương trung bình (mean squared error - MSE). Công thức của GD cho mô hình hồi quy tuyến tính là:

$$\beta_{\text{new}} = \beta_{\text{old}} - \alpha \nabla MSE(\beta)$$

Trong đó, α là tốc độ học (learning rate), ∇MSE(β) là đạo hàm của hàm mất mát MSE theo tham số β.

Xây dựng và huấn luyện mô hình hồi quy tuyến tính trên Python

Để xây dựng và huấn luyện mô hình hồi quy tuyến tính trên Python, chúng ta có thể sử dụng các thư viện như numpy, pandas và sklearn. Numpy là một thư viện hỗ trợ tính toán khoa học, cho phép làm việc với các mảng nhiều chiều và các hàm toán học. Pandas là một thư viện hỗ trợ xử lý và phân tích dữ liệu, cho phép làm việc với các đối tượng dataframe và series. Sklearn là một thư viện hỗ trợ học máy, cho phép làm việc với các mô hình, thuật toán và công cụ học máy.

Dưới đây là đoạn code Python để xây dựng và huấn luyện mô hình hồi quy tuyến tính trên Python, với các bước như sau:

- *Nhập các thư viện cần thiết*

```
from flask import Flask, request, jsonify

import joblib

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
```

- *Tạo dữ liệu mẫu cho tệp data.csv và data2.csv. Dữ liệu mẫu có hai biến độc lập x1 và x2, và một biến phụ thuộc y, với y là một hàm tuyến tính của x1 và*

x2 cộng với một nhiễu chuẩn. Dữ liệu được lưu vào các dataframe bằng pandas và sau đó được ghi vào các tệp csv.

```
# Tạo dữ liệu mẫu cho tệp data.csv
np.random.seed(42)
n_samples = 1000
x1 = np.random.rand(n_samples) * 10
x2 = np.random.rand(n_samples) * 5
noise = np.random.normal(0, 1, n_samples)
coefficients = [2, 3]
y = coefficients[0] * x1 + coefficients[1] * x2 + noise
data = pd.DataFrame({'x1': x1, 'x2': x2, 'y': y})

data.to_csv('data.csv', index=False)
# Tạo dữ liệu mẫu cho tệp data2.csv
new_data_size = 200
new_data = pd.DataFrame({
  'x1': np.random.rand(new_data_size),
  'x2': np.random.rand(new_data_size),
  'y': 2 * np.random.rand(new_data_size) + 3
})

new_data.to_csv('data2.csv', index=False)
```

- *Chia dữ liệu thành tập huấn luyện và tập kiểm tra, với tỷ lệ 75/25, bằng hàm train_test_split của sklearn.*

```
X_train, X_test, y_train, y_test = train_test_split(data[['x1', 'x2']], data['y'], test_size=0.25)
```

- *Tạo một mô hình hồi quy tuyến tính bằng hàm LinearRegression của sklearn và huấn luyện mô hình trên tập huấn luyện bằng phương thức fit.*

```
model = LinearRegression()

# Fit the model to the training data

model.fit(X_train, y_train)
```

- *Dự đoán trên tập kiểm tra bằng phương thức predict và đánh giá mô hình bằng cách tính toán sai số bình phương trung bình và căn bậc hai của sai số bình phương trung bình bằng numpy.*

```
new_y_pred = loaded_model.predict(data2[['x1', 'x2']])

# Evaluate the loaded model on new data

print('Mean squared error (continual learning):', np.mean((data2['y'] - new_y_pred) ** 2))

print('Root mean squared error (continual learning):', np.sqrt(np.mean((data2['y'] - new_y_pred)
** 2)))
```

- *Lưu mô hình đã huấn luyện vào tệp model.pkl bằng hàm dump của joblib.*

```
joblib.dump(model, 'model.pkl')

loaded_model = joblib.load('model.pkl')
```

Xem chi tiết tại:https://github.com/huycb94it/51900507_MachineLearning

CHƯƠNG III: KẾT LUẬN

Trong báo cáo này, em đã tìm hiểu và so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy, bao gồm Gradient Descent, Stochastic Gradient Descent, Momentum, Nesterov Accelerated Gradient, Adagrad, RMSprop, Adam, và AdaDelta. Qua đó đã trình bày khái niệm, mục đích, công thức, ưu điểm, và nhược điểm của từng phương pháp, cũng như minh hoạ bằng đoạn code Python để xây dựng và huấn luyện một mô hình hồi quy tuyến tính sử dụng các phương pháp này.

Em cũng đã tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó. Em đã giới thiệu khái niệm, thách thức, phương pháp, và ứng dụng của Continual Learning và Test Production, cũng như minh hoạ bằng đoạn code Python để thực hiện Continual Learning và Test Production cho mô hình hồi quy tuyến tính đã huấn luyện trước đó.

Qua báo cáo này, em hy vọng đã cung cấp cho bạn đọc những kiến thức cơ bản và hữu ích về các phương pháp Optimizer, Continual Learning và Test Production trong học máy, cũng như cách áp dụng chúng vào thực tế.

Hướng phát triển và nghiên cứu tiếp theo:

1. *Đối với các phương pháp Optimizer, em đề xuất nghiên cứu thêm về các phương pháp khác như Adadelta, Adamax, Nadam, AMSGrad, v.v., để so sánh và đánh giá hiệu quả của chúng trên các bài toán và dữ liệu khác nhau. Em cũng đề xuất nghiên cứu về cách lựa chọn và điều chỉnh các siêu tham số như tốc độ học, hệ số quán tính, v.v., để tối ưu hóa quá trình học.*
2. *Đối với Continual Learning, em đề xuất nghiên cứu thêm về các kỹ thuật khác như học đa nhiệm (multi-task learning), học siêu (meta-learning), học tăng cường (reinforcement learning), v.v., để giải quyết các thách thức như quên địa phương, xung đột biểu diễn, v.v. Em cũng đề xuất nghiên cứu về cách đánh giá và so sánh các phương pháp Continual Learning trên các tiêu chí khác nhau như độ chính xác, độ ổn định, độ linh hoạt, v.v.*
3. *Đối với Test Production, em đề xuất nghiên cứu thêm về các phương pháp khác như kiểm tra hồi quy (regression testing), kiểm tra khả năng chịu lỗi (fault tolerance testing), kiểm tra bảo mật (security testing), v.v., để đảm bảo chất lượng và an toàn của sản phẩm. Em cũng đề xuất nghiên cứu về cách thu thập, phân tích và tổng hợp kết quả từ các loại kiểm thử khác nhau để đưa ra quyết định về việc triển khai sản phẩm vào môi trường sản xuất.*