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

BÁO CÁO BÀI TẬP LỚN CUỐI KỲ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGS TS LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN PHÚC MINH ĐĂNG**

Lớp **: 21H50302**

Khoá **: 25**

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 BÀI TẬP LỚN CUỐI KỲ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGS TS LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN PHÚC MINH ĐĂNG – 521H0497**

Lớp **: 21H50302**

Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Cảm ơn thầy Lê Anh Cường đã nhiệt tình giảng dạy, hướng dẫn những kiến thức lý thuyết và thực hành của môn NHẬP MÔN HỌC MÁY để em có thể hoàn thành báo cáo bài tập lớn này.

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 20 tháng 12 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Phúc Minh Đăng

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

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

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

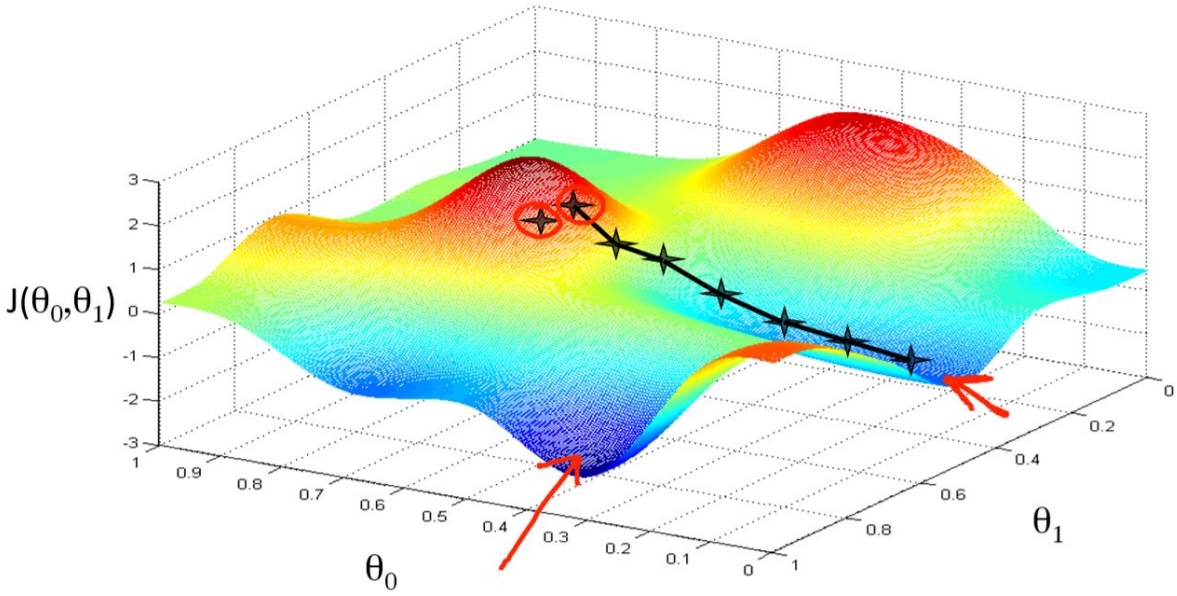
_____

_____

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

MỤC LỤC

CHƯƠNG 1 – OPTIMIZER



1.1 Optimizer là gì, tại sao phải dùng nó?

Optimizer là các thuật toán hoặc phương pháp được sử dụng để giảm thiểu hàm lỗi hàm mất mát(loss function), hoặc tối đa hóa hiệu suất sản xuất bằng cách điều chỉnh các tham số của mô hình để mô hình có thể học được từ dữ liệu huấn luyện một cách tối ưu nhất.

Trong học sâu, Optimizer được sử dụng để thay đổi các thuộc tính của mạng neural, chẳng hạn như trọng số và tốc độ học, để giảm thiểu mất mát hoặc chi phí trong quá trình lan truyền ngược.

Việc sử dụng Optimizer rất quan trọng vì nó giúp cải thiện hiệu suất của mô hình học máy hoặc học sâu. Nó điều chỉnh các tham số của mô hình để giảm thiểu sai số giữa dự đoán của mô hình và dữ liệu thực tế, giúp mô hình dự đoán chính xác hơn. Một số Optimizer phổ biến bao gồm Gradient Descent, Stochastic Gradient Descent (SGD), Adam, và nhiều hơn nữa. Mỗi Optimizer có những đặc điểm riêng, do đó việc chọn Optimizer phù hợp sẽ phụ thuộc vào bài toán cụ thể bạn đang giải quyết.

Trong machine learning, Optimizer đóng vai trò quan trọng trong việc cải thiện hiệu suất của mô hình và giảm thiểu sai số. Dưới đây là một số lý do chúng ta cần sử dụng Optimizer:

- **Giảm thiểu hàm mất mát**: Optimizer giúp điều chỉnh các tham số của mô hình để giảm thiểu hàm mất mát. Hàm mất mát là một hàm số đo lường sai số giữa dự đoán của mô hình và dữ liệu thực tế.
- **Cải thiện hiệu suất mô hình**: Optimizer giúp cải thiện hiệu suất của mô hình bằng cách tối ưu hóa các tham số của mô hình, giúp mô hình dự đoán chính xác hơn.
- **Tăng tốc độ học**: Một số Optimizer như Adam có thể điều chỉnh tốc độ học trong quá trình huấn luyện, giúp mô hình học nhanh hơn và hiệu quả hơn.
- **Tránh bẫy tại cực tiểu địa phương**: Một số Optimizer như Momentum hoặc RMSprop có thể giúp mô hình vượt qua các cực tiểu địa phương, nơi mà Gradient Descent có thể bị mắc kẹt.
- **Điều chỉnh tự động**: Một số Optimizer như Adagrad, RMSprop, hoặc Adam có khả năng điều chỉnh tự động tốc độ học dựa trên quá trình huấn luyện, giúp tối ưu hóa quá trình học.

Vì vậy, việc chọn đúng Optimizer phù hợp với bài toán cụ thể là rất quan trọng để đạt được hiệu suất tốt nhất từ mô hình. Tóm lại, sử dụng optimizer trong machine learning giúp mô hình học tốt hơn, tối ưu hóa hiệu suất và cải thiện độ chính xác của dự đoán.

1.2 Các thuật toán tối ưu:

1.2.1 Gradient Descent (GD)

là một thuật toán tối ưu hóa được sử dụng rộng rãi, thường được dùng trong machine learning để điều chỉnh các tham số của mô hình nhằm tối thiểu hóa hàm mất mát và cải thiện khả năng dự đoán chính xác trên dữ liệu mới.

- Ưu điểm**:**

1. Được sử dụng rộng rãi: Gradient Descent và các biến thể của nó được sử dụng rộng rãi trong machine learning và các vấn đề tối ưu hóa vì chúng hiệu quả và dễ triển khai.
2. Hiệu quả về mặt tính toán: Tất cả dữ liệu huấn luyện được xử lý cùng một lúc. Do đó, chỉ cần một số ít chu kỳ máy.
3. Ít dao động và hội tụ dễ dàng đến cực tiểu toàn cục.

- Nhược điểm**:**

1. Dù ít gặp phải cực tiểu địa phương, nhưng nếu gặp phải, nó sẽ không thể thoát ra do không có bước nhiễu.
2. Mặc dù hiệu quả về mặt tính toán nhưng không nhanh. Vì nó cần nhiều bộ nhớ để tải toàn bộ dữ liệu vào bộ nhớ cùng một lúc.

Vì vậy, việc lựa chọn đúng biến thể của Gradient Descent phù hợp với bài toán cụ thể là rất quan trọng để đạt được hiệu suất tốt nhất từ mô hình của chúng ta.

1.2.2 Momentum

Để khắc phục các hạn chế trên của thuật toán Gradient Descent người ta dùng gradient descent with momentum. Vậy gradient with momentum là gì ?
Gradient with momentum là một kỹ thuật tối ưu hóa trong quá trình huấn luyện mô hình machine learning, đặc biệt là trong việc cập nhật trọng số (weights) của mạng neural. Kỹ thuật này được thiết kế để giúp tăng tốc độ hội tụ và tránh được những dao động không mong muốn trong quá trình tối ưu hóa.
Cơ chế hoạt động của gradient with momentum dựa trên việc tích luỹ đà (momentum) từ các gradient trước đó để cập nhật trọng số. Thay vì chỉ sử dụng gradient hiện tại để điều chỉnh trọng số, gradient with momentum tính toán sự thay đổi của trọng số dựa trên gradient hiện tại và đà tích luỹ từ các bước trước.

Công thức cập nhật trọng số theo gradient with momentum thường được biểu diễn như sau:

$$\Delta w(t) = \alpha \cdot \Delta w(t-1) - \eta \cdot \nabla J(w)$$

Trong đó:

- $\Delta w(t)$ là bước cập nhật trọng số tại thời điểm t
- $\Delta w(t-1)$ là đà tích luỹ từ bước trước.
- α là hệ số đà, thường là một giá trị từ 0 đến 1 để điều chỉnh mức độ ảnh hưởng của đà tích luỹ.

- η là learning rate.
- $\nabla J(w)$ là gradient của hàm mất mát J theo trọng số w.

Kỹ thuật gradient with momentum giúp tránh được việc mô hình bị rơi vào các cực tiểu cục bộ, giúp tăng tốc độ hội tụ và giảm thiểu đào động của quá trình tối ưu hóa. Nó cũng giúp cho quá trình cập nhật trọng số diễn ra một cách mượt mà hơn.

1.2.3 Adagrad

Không giống như các thuật toán trước đó thì learning rate hầu như giống nhau trong quá trình training (learning rate là hằng số), Adagrad coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên sau mỗi thời điểm t.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

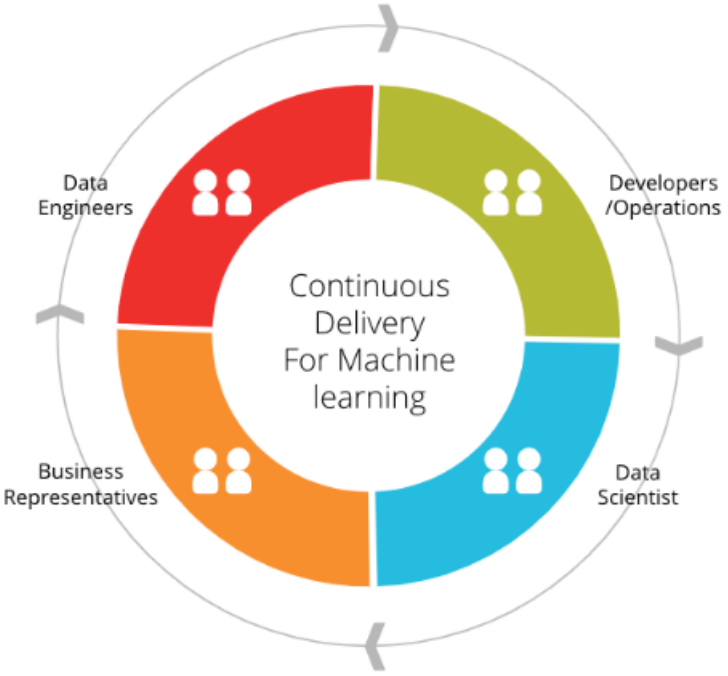
Trong đó:

- n : hằng số
- gt : gradient tại thời điểm t
- ϵ : hệ số tránh lỗi (chia cho mẫu bằng 0)
- G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

Ưu điểm: Một lợi ích dễ thấy của Adagrad là tránh việc điều chỉnh learning rate bằng tay, chỉ cần để tốc độ học default là 0.01 thì thuật toán sẽ tự động điều chỉnh.
Nhược điểm: tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual learning:



2.2.1 Định nghĩa và đặc trưng của Continual Learning

Continual Learning, còn được gọi là Lifelong Learning, là quá trình học liên tục về thế giới bên ngoài để cho phép phát triển tự động, tăng cường kỹ năng và kiến thức phức tạp hơn. Một hệ thống học liên tục có thể được định nghĩa là một thuật toán thích ứng có khả năng học từ một dòng thông tin liên tục, với thông tin này trở nên dần dần có sẵn theo thời gian và nơi số lượng nhiệm vụ cần học không được xác định trước.

Một số điểm quan trọng về Continual Learning:

- **Tránh quên hoàn toàn**: Khi tiếp nhận thông tin mới, việc này nên diễn ra mà không gây ra quên hoàn toàn hoặc can thiệp.

- **Cập nhật đại diện nội bộ**: Một mô hình học liên tục cần xây dựng và cập nhật động các đại diện nội bộ khi phân phối của nhiệm vụ thay đổi động qua thời gian.
- **Đại diện chung và đặc biệt**: Một phần của các đại diện nội bộ sẽ chung và bất biến đủ để có thể tái sử dụng qua các nhiệm vụ tương tự, trong khi một phần khác nên bảo tồn và mã hóa đại diện đặc biệt cho nhiệm vụ.

2.1.2 Tại sao Continual Learning lại cần thiết ?

Ngày nay, dữ liệu được tạo ra và cập nhật liên tục từ nhiều nguồn khác nhau, và dữ liệu cũ thường không còn đáng tin cậy như trước do sự thay đổi của môi trường hoặc xu hướng mới. Trong thế giới thực tế, các mô hình học máy phải linh hoạt, có khả năng học và thích nghi với dữ liệu mới để duy trì hiệu suất và độ chính xác.
Continual Learning giúp giải quyết vấn đề "quên" khi mô hình học máy chỉ được đào tạo trên tập dữ liệu cụ thể và không còn hoạt động tốt khi có dữ liệu mới. Nó giúp mô hình duy trì khả năng học và nhớ thông tin quan trọng từ dữ liệu cũ và cũng có khả năng học thêm thông tin mới mà không làm mất đi kiến thức đã học.

2.1.3 Các bước chính của Continual Learning

a. Đánh giá và lựa chọn mô hình phù hợp: Chọn các mô hình học máy có khả năng học liên tục và tương thích với việc tích hợp dữ liệu mới.
b. Xây dựng kiến trúc cho việc học liên tục: Tạo ra các phương pháp, thuật toán để mô hình có thể học và nhớ thông tin cũ trong quá trình học dữ liệu mới.
c. Quản lý dữ liệu: Tạo các chiến lược để quản lý dữ liệu mới, dữ liệu cũ và cách tích hợp chúng vào quá trình học mà không gây ảnh hưởng đến hiệu suất.
d. Tối ưu hóa việc học liên tục: Tối ưu hóa các thuật toán để đảm bảo rằng mô hình không quên thông tin quan trọng khi học dữ liệu mới.

2.1.4 Thách thức của Continual Learning:

a. Quên thông tin quan trọng: Mô hình có thể quên đi thông tin quan trọng từ dữ liệu cũ khi tiếp nhận dữ liệu mới, gây ra hiện tượng "catastrophic forgetting".
b. Phức tạp về tính toán: Continual Learning đòi hỏi các thuật toán tính toán phức tạp để đảm bảo rằng mô hình có khả năng học liên tục mà không tốn kém quá nhiều tài nguyên.
c. Hiệu suất và độ chính xác: Đạt được cân bằng giữa việc học dữ liệu mới và duy trì hiệu suất trên dữ liệu cũ là một thách thức lớn.

Continual Learning đang trở thành một lĩnh vực nghiên cứu quan trọng trong học máy, mang lại triển vọng cho việc xây dựng các mô hình học máy thông minh, linh hoạt và có khả năng thích ứng với môi trường thay đổi không ngừng. Sự tiến bộ trong Continual Learning không chỉ cung cấp những ứng dụng thực tế mạnh mẽ mà còn đặt nền móng cho việc tiến xa hơn trong lĩnh vực trí tuệ nhân tạo.

2.2 Test Production

Test Production trong học máy là một quá trình quan trọng nhằm đảm bảo rằng mô hình máy học hoạt động đúng đắn và hiệu quả trước khi triển khai vào môi trường thực tế. Nó đóng vai trò quan trọng trong việc đánh giá chất lượng của mô hình, xác nhận khả năng hoạt động và hiệu suất của nó trước khi đưa ra sử dụng thực tế trong sản xuất.

Quá trình Test Production thường bắt đầu sau khi mô hình đã được huấn luyện và kiểm định trên dữ liệu huấn luyện và dữ liệu kiểm tra. Tuy nhiên, điểm khác biệt chính là việc triển khai mô hình này vào môi trường sản xuất thực tế để kiểm tra tính ổn định, hiệu suất và khả năng mô hình hoạt động với dữ liệu mới, không nhất thiết phải giống với dữ liệu được sử dụng trong quá trình huấn luyện.

Một số bước quan trọng trong quá trình Test Production bao gồm:

- Chuẩn bị dữ liệu mới: Dữ liệu thực tế trong quá trình sản xuất có thể khác biệt so với dữ liệu được sử dụng trong quá trình huấn luyện. Do đó, việc chuẩn bị và làm sạch dữ liệu mới để kiểm tra mô hình là cực kỳ quan trọng.
- Triển khai mô hình: Mô hình được triển khai vào môi trường thực tế để kiểm tra hoạt động của nó. Điều này bao gồm cấu hình hạ tầng cần thiết và kết nối với hệ thống để mô hình có thể xử lý dữ liệu thời gian thực hoặc các yêu cầu từ môi trường sản xuất.
- Kiểm tra hiệu suất: Mô hình được đánh giá về hiệu suất trong môi trường thực tế, với việc thu thập các thông số như độ chính xác, độ phân loại, hoặc các chỉ số đo lường khác phụ thuộc vào loại bài toán máy học.
- Giám sát và cập nhật: Quá trình Test Production không chỉ dừng lại sau khi mô hình được triển khai. Việc giám sát liên tục hiệu suất của mô hình trong môi trường thực tế và cập nhật mô hình khi cần thiết là bước quan trọng để duy trì và cải thiện hiệu suất theo thời gian.
- Bảo mật và tuân thủ quy định: Bảo mật dữ liệu và tuân thủ các quy định pháp luật là một yếu tố quan trọng trong quá trình Test Production, đặc biệt khi mô hình xử lý dữ liệu cá nhân hoặc nhạy cảm.

Quá trình Test Production không chỉ giúp đánh giá mô hình mà còn giúp cải thiện và tối ưu hóa nó để phản ánh tốt nhất khả năng trong môi trường thực tế. Sự chính xác và hiệu suất của mô hình trong sản xuất có thể có ảnh hưởng lớn đến kết quả kinh doanh và trải

nghiệm người dùng cuối cùng, do đó, việc thực hiện Test Production một cách cẩn thận và toàn diện là rất quan trọng.

THE END