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

ĐỒ ÁN CUỐI KÌ

MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **GV. LÊ ANH CƯỜNG**

Người thực hiện: **PHẠM HOÀNG TRUNG KIÊN – 52100904**

Lớp **: 21050301**

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

ĐỒ ÁN CUỐI KÌ

MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **GV. LÊ ANH CƯỜNG**

Người thực hiện: **PHẠM HOÀNG TRUNG KIÊN – 52100904**

Lớp **: 21050301**

Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn đến các thầy cô Phòng Khoa Công nghệ thông tin đã tạo điều kiện để chúng em được học hỏi, tiếp cận với một môn học mới. Đặc biệt là thầy Lê Anh Cường, cảm ơn thầy vẫn luôn nhiệt tình giúp đỡ, truyền đạt kiến thức môn học thông qua từng buổi học, từng bài tập, từng lời giảng. Nhờ vậy chúng em mới có đủ cơ sở để hoàn thành bài báo cáo cuối kỳ môn nhập môn học máy hôm nay.

Cảm ơn thầy đã đồng hành và bên cạnh chúng em trong quá trình học tập vừa qua, chúc thầy có nhiều sức khỏe và đạt được nhiều thành công trong cuộc sống.

TP. Hồ Chí Minh, ngày 5 tháng 12 năm 2023
Tác giả
(Ký tên và ghi rõ họ tên)
(Đã ký)
Phạm Hoàng Trung Kiên

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

Em xin cam đoan đây là công trình nghiên cứu của riêng em và được sự hướng dẫn khoa học của thầy 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 cuối kỳ 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 cuối kỳ 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 18 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

(Đã ký)

Phạm Hoàng Trung Kiên

TÓM TẮT

Bài báo cáo này trình bày những nội dung mà em sẽ tìm hiểu được cũng như quá trình hiện thực đồ án qua từng giai đoạn khác nhau. Đầu tiên là phần làm cá nhân về một bài nghiên cứu, đánh giá của em về vấn đề sau :

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

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.

Ngoài ra trong đồ án cuối kì này còn phần làm bài tập nhóm để đưa ra bài toán dự đoán có thể giải quyết bằng học máy (machine leanring) với các yêu cầu:

- Số Feature/Attribute gồm nhiều kiểu: categorial và numerical;
- Dữ liệu phải chưa được học, thực tập trên lớp và trong bài tập về nhà

1. Phân tích thống kê trên dữ liệu, vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu. Tìm hiểu các đặc trưng và đánh gía vai trò của các đặc trưng đối với mục tiêu bài toán.
2. Ứng dụng các mô hình học máy cơ bản để giải quyết bài toán, bao gồm cả các mô hình thuộc Ensemble Learing.
3. Sử dụng Feed Forward Neural Network và Reccurent Neural Network (hoặc mô thuộc loại này) để giải quyết bài toán.
4. Áp dụng các kỹ thuật tránh Overfiting trên các mô hình của câu (2) và câu (3) để giải quyết bài toán.
5. Sau khi huấn luyện xong mô hình thì muốn cải thiện độ chính xác, ta sẽ làm gì để giải quyết nó? Phân tích các trường hợp sai, đề ra giải pháp và thực hiện nó, sau đó đánh giá xem có cải tiến so với trước không.

Phần làm nhóm thì sẽ thực hiện code, data và sẽ được nộp cùng với bài báo cáo này. Vì vậy trong bài báo cáo này chỉ thực hiện phần làm cá nhân.

MỤC LỤC

CHƯƠNG 1 – PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 OPTIMIZER

- Phương pháp Optimizer trong huấn luyện mô hình học máy là một thành phần quan trọng để tối ưu hóa hàm mất mát và điều chỉnh các tham số của mô hình để đạt được hiệu suất tốt nhất. Mục tiêu của bước tối ưu hóa là tìm ra bộ tham số tối ưu cho mô hình dựa trên dữ liệu huấn luyện.

- Công việc của một optimizer là thay đổi các tham số của mô hình dựa trên các giá trị gradient của hàm mất mát. Gradient là đạo hàm riêng của hàm mất mát theo từng tham số và cho biết hướng và độ lớn thay đổi của hàm mất mát khi các tham số thay đổi.

- Các phương pháp Optimizer

- Gradient Descent (GD)
- Adam (Adaptive Moment Estimation)
- RMSprop (Root Mean Square Propagation)
- AdaGrad (Adaptive Gradient)
- Adadelta

1.2 GRADIENT DESCENT (GD)

- Gradient Descent (GD) là một phương pháp tối ưu hóa cơ bản trong học máy và tối ưu lồi (convex optimization), được sử dụng để tìm giá trị tối thiểu của một hàm mất mát. Phương pháp này dựa trên việc điều chỉnh các tham số của mô hình dựa trên đạo hàm riêng của hàm mất mát theo từng tham số.

- Các bước chính của Gradient Descent:

1. Khởi tạo giá trị ban đầu cho các tham số của mô hình.
2. Lặp lại các bước sau cho một số lượng vòng lặp hoặc cho tới khi điều kiện dừng được đáp ứng:
    a) Tính đạo hàm riêng của hàm mất mát theo từng tham số.
    b) Cập nhật các tham số bằng cách đi theo hướng ngược với gradient và nhân với một learning rate (tỷ lệ học).
    c) Tính toán lại hàm mất mát với các tham số mới.

- Có ba biến thể chính của Gradient Descent:

1. Batch Gradient Descent: Trong biến thể này, toàn bộ tập dữ liệu huấn luyện được sử dụng để tính gradient và cập nhật tham số trong mỗi vòng lặp. Điều này đòi hỏi nhiều tài nguyên tính toán và không phù hợp cho các tập dữ liệu lớn.

2. Stochastic Gradient Descent (SGD): Thay vì sử dụng toàn bộ tập dữ liệu, SGD chỉ chọn ngẫu nhiên một mẫu trong mỗi vòng lặp để tính gradient và cập nhật tham số. Điều này giúp giảm tài nguyên tính toán và thích hợp cho các tập dữ liệu lớn. Tuy nhiên, SGD có thể gây ra sự dao động trong quá trình tối ưu và có thể không đạt được giá trị tối thiểu chính xác.

3. Mini-batch Gradient Descent: Đây là một biến thể kết hợp giữa Batch Gradient Descent và SGD. Thay vì sử dụng toàn bộ tập dữ liệu hoặc một mẫu duy nhất, Mini-batch Gradient Descent sử dụng một lượng nhỏ (mini-batch) mẫu để tính gradient và cập nhật tham số. Điều này giúp tiết kiệm tài nguyên tính toán và hội tụ nhanh hơn so với SGD.

- Chúng ta cùng xem xét một ví dụ đơn giản về việc sử dụng Gradient Descent để tìm giá trị tối thiểu của một hàm đơn giản là hàm bậc hai: f(x) = x^2.

Bước 1: Khởi tạo giá trị ban đầu cho tham số x, chẳng hạn x = 3.
Bước 2: Định nghĩa hàm mất mát J(x) dựa trên hàm f(x), trong trường hợp này, J(x) = f(x) = x^2.
Bước 3: Tính đạo hàm riêng của hàm mất mát J(x) theo tham số x: dJ(x)/dx = 2x.
Bước 4: Cập nhật tham số x bằng cách di chuyển ngược dọc theo đạo hàm với một learning rate (tỷ lệ học) α:

$$x = x - \alpha * dJ(x)/dx = x - \alpha * 2x.$$

Lặp lại bước 3 và bước 4 cho một số vòng lặp cho đến khi đạt được điều kiện dừng hoặc đạt được giá trị tối thiểu mong muốn.

Ví dụ, giả sử chúng ta chọn learning rate α = 0.1 và số vòng lặp là 5:
Vòng lặp 1:

- Giá trị ban đầu: x = 3
- Đạo hàm riêng: dJ(x)/dx = 2x = 2 * 3 = 6
- Cập nhật tham số: x = x - α * dJ(x)/dx = 3 - 0.1 * 6 = 2.4

Vòng lặp 2:

- Giá trị ban đầu: x = 2.4
- Đạo hàm riêng: dJ(x)/dx = 2x = 2 * 2.4 = 4.8
- Cập nhật tham số: x = x - α * dJ(x)/dx = 2.4 - 0.1 * 4.8 = 1.92

Tiếp tục lặp lại các bước trên cho đến khi đạt được số vòng lặp đã cho. Kết quả cuối cùng có thể tiến gần đến giá trị tối thiểu của hàm mất mát, trong trường hợp này là xấp xỉ bằng 0.

Điều quan trọng là chọn learning rate (α) phù hợp. Nếu learning rate quá lớn, quá trình tối ưu hóa có thể không hội tụ và dao động. Ngược lại, nếu learning rate quá nhỏ, quá trình tối ưu hóa có thể diễn ra rất chậm. Việc điều chỉnh learning rate và số vòng lặp là một quá trình thử và sai để đạt được hiệu suất tối ưu cho bài toán cụ thể.

- Mục tiêu của Gradient Descent là tìm giá trị tối thiểu của hàm mất mát bằng cách di chuyển dần các tham số theo hướng ngược với gradient. Quá trình này tiếp tục cho đến khi đạt được điều kiện dừng, chẳng hạn như số lượng vòng lặp đã đủ hoặc thay đổi của hàm mất mát rất nhỏ.

1.3 ADAM (ADAPTIVE MOMENT ESTIMATION)

- Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa được sử dụng rộng rãi trong học máy và mạng nơ-ron. Nó kết hợp các ưu điểm của hai thuật toán khác là Momentum và RMSprop để cung cấp tốc độ hội tụ nhanh và ổn định hơn trong quá trình tối ưu hóa.

- Adam sử dụng thông tin từ gradient của các tham số và quỹ đạo lịch sử của chúng để điều chỉnh learning rate cho từng tham số trong quá trình tối ưu. Nó duy trì hai bộ nhớ đệm (momentum và RMSprop) để ước tính độ lớn của gradient và độ lớn của gradient squared.

- Các bước chính của Adam:

1. Khởi tạo giá trị ban đầu cho các tham số và bộ nhớ đệm (ví dụ: momentum và RMSprop).
2. Lặp lại các bước sau cho mỗi vòng lặp:

a. Tính gradient của hàm mất mát theo từng tham số.

b. Cập nhật bộ nhớ đệm momentum bằng cách tích lũy gradient theo hướng trước đó.

c. Cập nhật bộ nhớ đệm RMSprop bằng cách tích lũy gradient squared theo hướng trước đó.

d. Điều chỉnh momentum và RMSprop để tính toán mức độ sửa đổi của các tham số.

e. Cập nhật các tham số bằng cách sử dụng mức độ sửa đổi được tính toán và một learning rate.

f. Lặp lại các bước từ a đến e cho đến khi đạt được điều kiện dừng.

- Adam sử dụng một số siêu tham số để điều chỉnh quá trình tối ưu hóa, bao gồm learning rate (α), hệ số decay cho momentum (β1), hệ số decay cho RMSprop (β2), và một epsilon nhỏ để tránh chia cho 0.

- Ưu điểm chính của Adam là khả năng tự động điều chỉnh learning rate cho từng tham số dựa trên gradient và quỹ đạo lịch sử. Điều này giúp Adam hiệu quả hơn trong việc tìm hiểu các mô hình phức tạp và hội tụ nhanh hơn đối với các bài toán khó khăn.

- Tuy nhiên, việc chọn các siêu tham số phù hợp vẫn là một thách thức và cần thử nghiệm để đạt được hiệu suất tối ưu của thuật toán Adam.

1.4 RMSPROP (ROOT MEAN SQUARE PROPAGATION)

- RMSprop (Root Mean Square Propagation) là một thuật toán tối ưu hóa thường được sử dụng trong học máy và mạng nơ-ron để cải thiện quá trình tối ưu hóa và tăng tốc độ hội tụ.

- RMSprop sử dụng một bộ nhớ đệm để duy trì quỹ đạo lịch sử của gradient. Nó điều chỉnh learning rate cho từng tham số dựa trên giá trị trung bình của các bình phương gradient trước đó.

$$E[g^2]_t = 0{,}9E[g^2]_{t\text{-}1} + 0{,}1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

- Các bước chính của RMSprop:

1. Khởi tạo giá trị ban đầu cho các tham số và bộ nhớ đệm (ví dụ: quỹ đạo lịch sử gradient).
2. Lặp lại các bước sau cho mỗi vòng lặp:

a. Tính gradient của hàm mất mát theo từng tham số.

b. Cập nhật quỹ đạo lịch sử gradient bằng cách tích lũy các bình phương của gradient trước đó.

c. Tính toán giá trị trung bình của các bình phương gradient trước đó.

d. Điều chỉnh learning rate cho từng tham số bằng cách chia gradient hiện tại cho căn bậc hai của giá trị trung bình gradient squared.

e. Cập nhật các tham số bằng cách sử dụng learning rate đã điều chỉnh và gradient.

f. Lặp lại các bước từ a đến e cho đến khi đạt được điều kiện dừng.

- RMSprop giúp điều chỉnh learning rate dựa trên gradient trước đó. Nó tập trung vào các tham số có gradient lớn và giảm learning rate cho các tham số có gradient nhỏ. Điều này giúp ổn định việc tối ưu hóa và ngăn chặn việc đi quá xa khỏi điểm tối ưu cục bộ.

- RMSprop có một siêu tham số là decay rate (γ) để điều chỉnh mức độ ảnh hưởng của quỹ đạo lịch sử gradient. Nếu decay rate lớn, quỹ đạo lịch sử gradient sẽ có mức độ ảnh hưởng cao hơn và learning rate sẽ giảm nhanh hơn. Ngược lại, nếu decay rate nhỏ, quỹ đạo lịch sử gradient sẽ có mức độ ảnh hưởng thấp hơn và learning rate sẽ giảm chậm hơn.

- RMSprop giúp cải thiện hiệu suất tối ưu hóa và hội tụ nhanh hơn trong quá trình huấn luyện mạng nơ-ron và các mô hình học máy khác. Tuy nhiên, việc chọn các siêu tham số phù hợp vẫn là một yếu tố quan trọng để đạt được hiệu suất tối ưu của thuật toán RMSprop

1.5 ADAGRAD (ADAPTIVE GRADIENT)

- AdaGrad (Adaptive Gradient) là một thuật toán tối ưu hóa trong lĩnh vực học máy và mạng nơ-ron. Nó được thiết kế để điều chỉnh learning rate cho từng tham số dựa trên lịch sử của gradient. AdaGrad tự động thích nghi với các tham số có gradient lớn bằng cách giảm learning rate và ngược lại, tăng learning rate cho các tham số có gradient nhỏ.

- Các bước chính của AdaGrad:

1. Khởi tạo giá trị ban đầu cho các tham số và bộ nhớ đệm (ví dụ: quỹ đạo lịch sử của gradient).
2. Lặp lại các bước sau cho mỗi vòng lặp:

a. Tính gradient của hàm mất mát theo từng tham số.

b. Cập nhật quỹ đạo lịch sử gradient bằng cách tích lũy các bình phương của gradient trước đó.

c. Điều chỉnh learning rate cho từng tham số bằng cách chia learning rate ban đầu cho căn bậc hai của giá trị trung bình của quỹ đạo lịch sử gradient squared.

d. Cập nhật các tham số bằng cách sử dụng learning rate đã điều chỉnh và gradient.

e. Lặp lại các bước từ a đến d cho đến khi đạt được điều kiện dừng.

- AdaGrad giúp điều chỉnh learning rate dựa trên quỹ đạo lịch sử gradient. Nó tích lũy các bình phương của gradient từ các vòng lặp trước đó để điều chỉnh learning rate. Các tham số có gradient lớn sẽ có learning rate giảm, trong khi các tham số có gradient nhỏ sẽ có learning rate tăng lên. Điều này giúp AdaGrad xử lý tốt các tham số hiếm gặp và giúp ổn định quá trình tối ưu hóa.

- Ưu điểm của AdaGrad là nó cung cấp quá trình tối ưu hóa tự động và hiệu quả cho các bài toán với dữ liệu thưa và các tham số có độ lớn gradient khác nhau. Nó giúp giảm learning rate đối với các tham số quan trọng và tăng learning rate đối với các tham số không quan trọng, giúp tăng tốc độ hội tụ và ổn định quá trình tối ưu hóa.

- Tuy nhiên, AdaGrad có một nhược điểm là sau một số vòng lặp, quỹ đạo lịch sử gradient có thể trở nên rất lớn, dẫn đến learning rate giảm quá nhanh. Điều này có thể khiến quá trình tối ưu hóa dừng sớm và không đạt được kết quả tối ưu. Để khắc phục vấn đề này, các biến thể khác như AdaDelta và RMSprop đã được phát triển.

1.6 ADADELTA

- Adadelta là một thuật toán tối ưu hóa trong lĩnh vực học máy và mạng nơ-ron, là một biến thể của AdaGrad. Adadelta giải quyết nhược điểm của AdaGrad bằng cách giới hạn quỹ đạo lịch sử gradient và làm giảm độ giảm learning rate theo thời gian.

- Các bước chính của Adadelta:

1. Khởi tạo giá trị ban đầu cho các tham số và bộ nhớ đệm (ví dụ: quỹ đạo lịch sử gradient và quỹ đạo lịch sử của thay đổi tham số).
2. Lặp lại các bước sau cho mỗi vòng lặp:

a. Tính gradient của hàm mất mát theo từng tham số.

b. Cập nhật quỹ đạo lịch sử gradient bằng cách tích lũy các bình phương của gradient trước đó.

c. Tính toán giá trị trung bình di động của quỹ đạo lịch sử gradient.

d. Tính toán điều chỉnh learning rate bằng cách chia căn bậc hai của giá trị trung bình di động gradient squared cho căn bậc hai của giá trị trung bình di động của quỹ đạo lịch sử thay đổi tham số squared.

e. Cập nhật quỹ đạo lịch sử của thay đổi tham số bằng cách tích lũy các bình phương của thay đổi tham số trước đó.

f. Tính toán điều chỉnh tham số bằng cách nhân giá trị trung bình di động điều chỉnh learning rate với bình phương căn bậc hai của giá trị trung bình di động của quỹ đạo lịch sử thay đổi tham số squared.

g. Cập nhật các tham số bằng cách sử dụng thay đổi tham số đã điều chỉnh.

h. Lặp lại các bước từ a đến g cho đến khi đạt được điều kiện dừng.

- Adadelta sử dụng hai quỹ đạo lịch sử: một là để tích lũy các bình phương của gradient, và một là để tích lũy các bình phương của thay đổi tham số. Điều này giúp giới hạn quỹ đạo lịch sử gradient và giảm độ giảm learning rate theo thời gian, đồng thời giúp điều chỉnh tham số dựa trên sự tương quan giữa các thay đổi tham số gần đây.

- Ưu điểm của Adadelta là nó giúp loại bỏ việc cần thiết phải cấu hình một learning rate ban đầu và giảm tác động của learning rate trong quá trình tối ưu hóa. Nó tự động điều chỉnh learning rate dựa trên lịch sử của quỹ đạo lịch sử gradient và thay đổi tham số, giúp cải thiện tốc độ hội tụ và ổn định quá trình tối ưu hóa.

- Tuy nhiên, Adadelta vẫn có một số siêu tham số cần được đặt, như giá trị ban đầu của quỹ đạo lịch sử gradient và quỹ đạo lịch sử của thay đổi tham số. Việc chọn các giá trị này có thể ảnh hưởng đến hiệu suất của thuật toán.

1.7 SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER

1.7.1 Gradient Descent (GD):

Ưu điểm:

- Đơn giản và dễ hiểu.
- Dễ dàng cấu hình learning rate.
- Có thể áp dụng cho các bài toán tối ưu hóa đơn giản.
- Cập nhật các tham số bằng cách di chuyển theo ngược hướng đạo hàm của hàm mất mát.

Nhược điểm:

- Có thể bị rơi vào điểm tối ưu cục bộ.
- Không hiệu quả cho các bài toán có dữ liệu lớn và không trơn.

1.7.2 Adam (Adaptive Moment Estimation):

Ưu điểm:

- Kết hợp các ưu điểm của RMSprop và Momentum.
- Có khả năng điều chỉnh learning rate tự động và hiệu quả.
- Phù hợp cho các bài toán với dữ liệu lớn và không trơn.

Nhược điểm:

- Cần cấu hình các siêu tham số như learning rate ban đầu, beta1 và beta2.

1.7.3 RMSprop (Root Mean Square Propagation):

Ưu điểm:

- Giúp giảm learning rate cho các gradient lớn và tăng learning rate cho các gradient nhỏ.
- Thích ứng với learning rate dựa trên quỹ đạo lịch sử gradient squared.
- Hiệu quả cho các bài toán với dữ liệu thưa và các tham số có độ lớn gradient khác nhau.

Nhược điểm:

- Cần cấu hình learning rate ban đầu.

1.7.4 AdaGrad (Adaptive Gradient):

Ưu điểm:

- Thích ứng với learning rate dựa trên lịch sử gradient.
- Hiệu quả với dữ liệu thưa và các tham số có độ lớn gradient khác nhau.

Nhược điểm:

- Có thể giảm learning rate quá nhanh với quỹ đạo lịch sử gradient lớn.

1.7.5 Adadelta:

Ưu điểm:

- Giống như AdaGrad, nhưng thay vì lưu trữ lịch sử của toàn bộ gradient, nó chỉ lưu trữ một số lượng hạn chế các gradient trước đó.

- Giới hạn quỹ đạo lịch sử gradient và thay đổi tham số, giảm độ giảm learning rate theo thời gian.
- Loại bỏ việc cấu hình learning rate ban đầu.

Nhược điểm:

- Vẫn cần cấu hình các giá trị ban đầu cho quỹ đạo lịch sử gradient và quỹ đạo lịch sử thay đổi tham số.

Tóm lại, mỗi phương pháp tối ưu hóa có ưu điểm và nhược điểm riêng. Sự lựa chọn tối ưu hóa phụ thuộc vào bài toán cụ thể và đặc điểm của dữ liệu. Adam và RMSprop thường được sử dụng phổ biến trong các bài toán thực tế, trong khi AdaGrad và Adadelta thích hợp cho các bài toán với dữ liệu thưa và các tham số có độ lớn gradient khác nhau. Gradient Descent là phương pháp cơ bản và đơn giản nhất nhưng có hiệu quả hơn khi được kết hợp với các biến thể khác như Momentum, Adam, RMSprop, AdaGrad hoặc Adadelta.

CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Khái niệm

- Continual Learning (học liên tục) là một lĩnh vực trong học máy tập trung vào khả năng của một hệ thống máy tính để tiếp tục học tập và cải thiện hiệu suất của nó qua thời gian khi được cung cấp dữ liệu mới. Trong một môi trường học liên tục, hệ thống phải xử lý dữ liệu mới và tích lũy kiến thức từ dữ liệu đã học trước đó mà không quên đi kiến thức cũ.

- Test Production (sản xuất dữ liệu kiểm tra) là quá trình tạo ra dữ liệu kiểm tra để đánh giá hiệu suất của mô hình học máy. Trong ngữ cảnh của học máy, quá trình này thường bao gồm hai giai đoạn chính: tạo dữ liệu kiểm tra và đánh giá mô hình trên dữ liệu đó.

2.2 Các phương pháp và kỹ thuật Continual Learning để xử lý việc học liên tục trên dữ liệu ngày càng lớn và đa dạng

- Có nhiều phương pháp và kỹ thuật được phát triển để giải quyết vấn đề học liên tục trên dữ liệu ngày càng lớn và đa dạng. Dưới đây là một số phương pháp và kỹ thuật quan trọng trong lĩnh vực này:

1) Elastic Weight Consolidation (EWC): EWC là một phương pháp học liên tục dựa trên việc giữ vững kiến thức đã học trước đó bằng cách áp dụng một hàm mất mát đặc biệt. Hàm mất mát này đảm bảo rằng các trọng số quan trọng cho nhiệm vụ trước đó không thay đổi quá nhiều trong quá trình học nhiệm vụ mới.

2) Online EWC và Synaptic Intelligence: Đây là các biến thể của EWC được thiết kế để xử lý học liên tục trực tuyến, trong đó dữ liệu mới được đưa vào hệ thống theo từng mẫu. Những phương pháp này cập nhật kiến thức đã học ngay lập tức mà không cần lưu trữ toàn bộ dữ liệu huấn luyện.

3) Generative Replay: Phương pháp này sử dụng một mô hình sinh dữ liệu (generative model) để tạo ra các mẫu dữ liệu từ các nhiệm vụ trước đó. Các mẫu này được sử dụng để huấn luyện và giúp mô hình học liên tục bằng cách phân biệt các nhiệm vụ khác nhau.

4) Memory-based Approaches: Các phương pháp này sử dụng bộ nhớ ngoại tại để lưu trữ các mẫu dữ liệu quan trọng từ các nhiệm vụ trước đó. Bộ nhớ này được sử dụng để tái sử dụng dữ liệu trong quá trình học mới, giúp giảm hiện tượng quên đi kiến thức cũ. Một số phương pháp như Experience Replay và Memory Replay thuộc loại này.

5) Incremental Learning: Phương pháp này tập trung vào việc học liên tục khi dữ liệu mới được cung cấp theo các tầng tăng dần. Mô hình được huấn luyện trên các tập dữ liệu con trước khi được kết hợp lại để học trên tập dữ liệu toàn bộ. Các phương pháp như iCaRL và LwF thuộc loại này.

6) Model Compression: Kỹ thuật nén mô hình giúp giảm kích thước của mô hình và giảm khả năng quên đi kiến thức cũ. Kỹ thuật như Knowledge Distillation và Parameter Pruning có thể được sử dụng để nén mô hình trong quá trình học liên tục.

2.3 Các phương pháp Test Production để tạo ra dữ liệu kiểm tra thích hợp và đáng tin cậy

- Các phương pháp Test Production nhằm tạo ra dữ liệu kiểm tra thích hợp và đáng tin cậy để đánh giá hiệu suất của mô hình học máy. Dưới đây là một số phương pháp thông thường được sử dụng để tạo ra dữ liệu kiểm tra:

1) Hold-Out Method: Đây là phương pháp đơn giản nhất trong đánh giá mô hình. Dữ liệu ban đầu được chia thành hai phần: một phần được sử dụng để huấn luyện mô hình và phần còn lại được giữ lại làm dữ liệu kiểm tra. Tuy nhiên, cách chia này có thể dẫn đến sự không cân bằng giữa dữ liệu huấn luyện và dữ liệu kiểm tra.

2) Cross-Validation: Cross-Validation là một phương pháp tạo ra nhiều tập dữ liệu kiểm tra từ một tập dữ liệu ban đầu. Dữ liệu ban đầu được chia thành các phần nhỏ và mô hình được huấn luyện và đánh giá trên các tập con khác nhau. Kỹ thuật Cross-Validation giúp đánh giá mô hình một cách tổng quát hơn và giảm thiểu sự ảnh hưởng của việc chia dữ liệu.

3) Stratified Sampling: Phương pháp này được sử dụng khi dữ liệu phân phối không đồng đều giữa các lớp. Stratified Sampling đảm bảo rằng tỷ lệ các lớp trong tập dữ liệu kiểm tra được duy trì gần như như tỷ lệ ban đầu. Điều này đảm bảo mô hình được đánh giá trên các lớp dữ liệu đại diện và giảm thiểu hiện tượng mất cân bằng dữ liệu.

4) Bootstrapping: Bootstrapping là phương pháp tạo ra các tập dữ liệu kiểm tra bằng cách lấy mẫu ngẫu nhiên từ tập dữ liệu ban đầu với việc thay thế. Quá trình này cho phép tạo ra nhiều tập dữ liệu kiểm tra khác nhau, giúp đánh giá độ ổn định và độ tin cậy của mô hình.

5) Synthetic Data Generation: Đôi khi, việc thu thập hoặc tạo ra dữ liệu thực tế để kiểm tra mô hình có thể tốn kém hoặc khó khăn. Trong trường hợp này, có thể sử dụng các phương pháp tạo dữ liệu tổng hợp để tạo ra dữ liệu kiểm tra. Các phương pháp như Generative Adversarial Networks (GANs) hoặc Augmentation Techniques có thể được sử dụng để tạo ra dữ liệu kiểm tra mới từ dữ liệu có sẵn.

6) Real-World Evaluation: Đối với một số bài toán cụ thể, việc thực hiện một quá trình đánh giá trực tiếp trên môi trường thực tế có thể là tốt nhất. Điều này đòi hỏi triển khai mô hình trong môi trường thực tế và thu thập dữ liệu kiểm tra từ các tương tác thực tế.

Những phương pháp trên có thể được kết hợp hoặc điều chỉnh tùy thuộc vào bài toán và yêu cầu cụ thể. Mục tiêu là tạo ra dữ liệu kiểm tra đủ đại diện và mang tính đáng tin cậy để đánh giá hiệu suất của mô hình học máy.

2.4 Áp dụng các kiến thức đã nghiên cứu vào một bài toán cụ thể và xây dựng giải pháp học máy bằng cách sử dụng Continual Learning và Test Production.

2.4.1 Bài toán 1:

Bài toán phân loại hình ảnh trong môi trường học liên tục. Trong ví dụ này, chúng ta sẽ giả định có hai nhiệm vụ phân loại hình ảnh khác nhau: phân loại loài hoa và phân loại động vật.

Bước 1: Huấn luyện mô hình ban đầu

Sử dụng dữ liệu huấn luyện ban đầu để huấn luyện mô hình phân loại hình ảnh ban đầu cho nhiệm vụ phân loại loài hoa. Mô hình này sẽ được huấn luyện trên tập dữ liệu loài hoa và đạt được độ chính xác đáng kể trên nhiệm vụ này.

Bước 2: Giữ vững kiến thức đã học

Áp dụng phương pháp Continual Learning để giữ vững kiến thức đã học từ nhiệm vụ phân loại loài hoa. Một trong những phương pháp có thể sử dụng là Elastic Weight Consolidation (EWC). EWC sẽ đảm bảo rằng các trọng số quan trọng cho nhiệm vụ phân loại loài hoa không thay đổi quá nhiều trong quá trình học nhiệm vụ mới.

Bước 3: Tiếp tục học trên nhiệm vụ phân loại động vật

Tiếp theo, chúng ta sẽ đưa ra nhiệm vụ phân loại động vật mới. Dữ liệu huấn luyện cho nhiệm vụ này sẽ được thêm vào dần dần với các mẫu dữ liệu động vật.
Sử dụng các phương pháp Continual Learning như EWC hoặc Generative Replay để đảm bảo rằng mô hình không quên kiến thức đã học từ nhiệm vụ phân loại loài hoa trong quá trình học nhiệm vụ mới.

Bước 4: Tạo dữ liệu kiểm tra đáng tin cậy

Áp dụng các phương pháp Test Production để tạo ra dữ liệu kiểm tra đáng tin cậy cho cả nhiệm vụ phân loại loài hoa và phân loại động vật. Các phương pháp như Cross-Validation, Stratified Sampling hoặc Bootstrapping có thể được sử dụng để tạo ra các tập dữ liệu kiểm tra đại diện cho các nhiệm vụ.

Bước 5: Đánh giá mô hình

Sử dụng dữ liệu kiểm tra đã tạo ra để đánh giá hiệu suất của mô hình trên cả nhiệm vụ phân loại loài hoa và phân loại động vật. Điều này sẽ cho phép chúng ta đánh giá độ chính xác và độ tin cậy của mô hình trên các nhiệm vụ khác nhau.

-> Qua các bước thực hiện, ta đã xác định một quy trình cơ bản để huấn luyện mô hình ban đầu, giữ vững kiến thức đã học, tiếp tục học trên nhiệm vụ mới và đánh giá mô hình. Các phương pháp Continual Learning như EWC và Generative Replay đã được đề cập, cùng với các phương pháp Test Production như Cross-Validation và Stratified Sampling.

2.4.2 Bài toán 2:

Giả sử chúng ta có một bài toán phân loại email vào hai nhãn: "Spam" và "Không phải spam". Chúng ta muốn xây dựng một giải pháp học máy sử dụng Continual Learning và Test Production để đáp ứng nhu cầu này.

Bước 1: Huấn luyện mô hình ban đầu

Sử dụng một tập dữ liệu ban đầu gồm email đã được gán nhãn để huấn luyện mô hình phân loại email. Chúng ta có thể sử dụng mô hình học máy như Convolutional Neural Network (CNN) hoặc Recurrent Neural Network (RNN) để xây dựng mô hình phân loại.

Bước 2: Giữ vững kiến thức đã học

Áp dụng phương pháp Continual Learning để giữ vững kiến thức đã học từ nhiệm vụ phân loại ban đầu. Một phương pháp Continual Learning phổ biến là Elastic Weight Consolidation (EWC). EWC sẽ đảm bảo rằng các trọng số quan trọng cho nhiệm vụ phân loại email không thay đổi quá nhiều trong quá trình học nhiệm vụ mới.

Bước 3: Tiếp tục học trên nhiệm vụ mới

Đối với nhiệm vụ tiếp theo, chúng ta muốn phân loại các email vào các nhãn khác nhau, ví dụ: "Quảng cáo", "Hợp đồng", và "Cảnh báo". Dữ liệu huấn luyện cho nhiệm vụ này sẽ được thêm vào dần dần với các mẫu email thuộc các nhãn mới.
Sử dụng các phương pháp Continual Learning như EWC hoặc Generative Replay để đảm bảo rằng mô hình không quên kiến thức đã học từ nhiệm vụ phân loại email trước đó trong quá trình học nhiệm vụ mới.

Bước 4: Tạo dữ liệu kiểm tra đáng tin cậy

Áp dụng các phương pháp Test Production để tạo ra dữ liệu kiểm tra đáng tin cậy cho cả nhiệm vụ phân loại email ban đầu và nhiệm vụ phân loại email mới. Các phương pháp như Cross-Validation, Stratified Sampling hoặc Bootstrapping có thể được sử dụng để tạo ra các tập dữ liệu kiểm tra đại diện cho cả hai nhiệm vụ.

Bước 5: Đánh giá mô hình

Sử dụng dữ liệu kiểm tra đã tạo ra để đánh giá hiệu suất của mô hình trên cả nhiệm vụ phân loại email ban đầu và nhiệm vụ phân loại email mới. Điều này sẽ cho phép chúng ta đánh giá độ chính xác và độ tin cậy của mô hình trên các nhiệm vụ khác nhau.

-> Bước tiếp tục học trên nhiệm vụ mới và tạo dữ liệu kiểm tra đáng tin cậy đã được đề cập. Tuy nhiên, các phương pháp cụ thể như CNN hay RNN chưa được đề cập rõ ràng. Ngoài ra, chúng ta có thể mở rộng bài toán này bằng cách áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) để nắm bắt nội dung của email và cải thiện hiệu suất phân loại

2.5 Kết Luận

1. Continual Learning (Học liên tục):
    - Continual Learning là một phương pháp quan trọng để giữ vững kiến thức đã học từ các nhiệm vụ trước đó trong quá trình học liên tục.

    - Sử dụng phương pháp Continual Learning như EWC hoặc Generative Replay giúp đảm bảo rằng mô hình không quên kiến thức đã học và có khả năng tiếp tục học trên nhiệm vụ mới mà không gây ảnh hưởng đáng kể đến hiệu suất trên các nhiệm vụ trước đó.

2. Test Production (Sản xuất dữ liệu kiểm tra):
    - Test Production là quá trình tạo ra dữ liệu kiểm tra đáng tin cậy để đánh giá hiệu suất của mô hình học máy.

    - Các phương pháp Test Production như Cross-Validation, Stratified Sampling hoặc Bootstrapping giúp tạo ra các tập dữ liệu kiểm tra đại diện cho các nhiệm vụ khác nhau, từ đó cho phép đánh giá độ chính xác và độ tin cậy của mô hình trên các nhiệm vụ đó.

Tổng hợp lại, Continual Learning và Test Production là hai phương pháp quan trọng và hữu ích trong quá trình xây dựng và đánh giá các hệ thống học máy trong môi trường học liên tục. Chúng giúp mô hình giữ vững kiến thức đã học, tiếp tục học trên nhiệm vụ mới và đảm bảo độ chính xác và độ tin cậy của mô hình khi đánh giá trên các nhiệm vụ khác nhau.

TÀI LIỆU THAM KHẢO

[1] Optimizer- Hiểu sâu về các thuật toán tối ưu (GD,SGD,Adam,..)
https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

[2] Gradient Descent
https://machinelearningcoban.com/2017/01/16/gradientdescent2/

[3] Continual Learning
https://paperswithcode.com/task/continual-learning

[4] Introduction to Continual Learning
https://wiki.continualai.org/the-continualai-wiki/introduction-to-continual-learning

[5] Test production
https://applyingml.com/resources/testing-ml/

[6] EWC
https://machinelearning.apple.com/research/elastic-weight-consolidation

[7] AdaGrad - Optimization Wiki
AdaGrad - Optimization Wiki

[8] RMSPROP VS ADAM
RMSPROP VS ADAM