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

PHẠM HOÀNG SƠN - 520H0150

BÀI CUỐI KỲ
NHẬP MÔN HỌC MÁY

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Thầy Lê Anh Cường, người đã hướng dẫn và hỗ trợ tận tình cho nhóm chúng em trong quá trình thực hiện báo cáo này. Sự kiên nhẫn, kiến thức chuyên môn, và kinh nghiệm quý báu của Thầy đã giúp nhóm chúng em nắm bắt được những kiến thức cần thiết và áp dụng chúng một cách hiệu quả để hoàn thiện báo cáo. Sự giảng dạy và sự hỗ trợ của Thầy không chỉ giúp chúng em vượt qua những khó khăn trong quá trình làm báo cáo, mà còn là nguồn cảm hứng để chúng em phấn đấu và học hỏi nhiều hơn nữa. Chúng em xin chân thành cảm ơn và mong rằng sẽ tiếp tục nhận được sự hỗ trợ và hướng dẫn của Thầy trong tương lai.

TP. Hồ Chí Minh, ngày 23 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

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

1.1 Optimizer là gì?

Trong học máy, một "Optimizer" hay phương pháp tối ưu hóa, là một thuật toán hoặc phương pháp được sử dụng để thay đổi các thuộc tính của mô hình học máy như trọng số và tốc độ học, nhằm giảm thiểu các sai số hoặc mất mát trong quá trình huấn luyện. Mục tiêu của việc tối ưu hóa là tìm ra bộ tham số tốt nhất cho mô hình để nó có thể dự đoán hoặc phân loại dữ liệu một cách chính xác nhất.

1.1.1 Chức năng cơ bản

- Giảm Thiểu Hàm Mất Mát: Optimizer cố gắng tìm ra bộ tham số mà tại đó, hàm mất mát (đo lường sự sai lệch giữa dự đoán của mô hình và dữ liệu thực tế) đạt giá trị nhỏ nhất.
- Cập Nhật Tham Số Mô Hình: Quá trình này bao gồm việc tính toán gradient của hàm mất mát theo các tham số và sử dụng thông tin này để cập nhật các tham số mô hình theo một hướng mà hàm mất mát giảm.

1.1.2 Các loại Optimizer phổ biến

- Gradient Descent: Cơ bản nhất, sử dụng toàn bộ dữ liệu để cập nhật tham số sau mỗi lần lặp.
- Stochastic Gradient Descent (SGD): Cập nhật tham số dựa trên từng mẫu dữ liệu, giúp tăng tốc độ tính toán và có khả năng vượt qua cực tiểu địa phương.
- Mini-Batch Gradient Descent: Kết hợp giữa GD và SGD, cập nhật tham số dựa trên một nhóm nhỏ các mẫu dữ liệu.
- Momentum: Cải tiến của SGD, giúp thuật toán nhanh hơn và ổn định hơn bằng cách tích lũy gradient của các bước trước đó.

- Adam (Adaptive Moment Estimation): Một trong những optimizer phổ biến nhất, kết hợp ý tưởng của Momentum và RMSprop, tự động điều chỉnh tốc độ học.

1.2 Một số loại Optimizer phổ biến

1.2.1 Gradient Descent (GD)

Phương pháp tối ưu hóa Gradient Descent (GD) là một trong những thuật toán cơ bản và phổ biến nhất trong lĩnh vực học máy và học sâu. Nó được sử dụng để tối ưu hóa hàm mất mát (loss function) của mô hình. Dưới đây là một cái nhìn chi tiết về cách hoạt động của Gradient Descent:

1.2.1.1 Khái niệm cơ bản

- Mục tiêu: Tìm ra bộ tham số của mô hình (ví dụ: trọng số của mạng nơ-ron) để giảm thiểu hàm mất mát.
- Hàm mất mát: Là một hàm toán học định lượng sai số giữa dự đoán của mô hình và dữ liệu thực tế.
- Gradient: Là đạo hàm của hàm mất mát theo các tham số mô hình. Nó chỉ ra hướng tăng hoặc giảm nhanh nhất của hàm mất mát.

1.2.1.2 Cơ chế hoạt động

- Khởi tạo ngẫu nhiên: Tham số mô hình được khởi tạo một cách ngẫu nhiên.
- Tính Gradient: Tính gradient (đạo hàm) của hàm mất mát tại điểm tham số hiện tại.
- Cập nhật tham số: Sử dụng gradient để cập nhật tham số theo công thức:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Trong đó:

- θ: Tham số mô hình (ví dụ: trọng số).
- η: Tốc độ học (learning rate), là một siêu tham số quyết định bước nhảy trong quá trình cập nhật.
- $J(\theta)$: Hàm mất mát.
- $\nabla_\theta J(\theta)$: Gradient của hàm mất mát theo θ.

- Lặp lại: Quá trình này được lặp lại nhiều lần cho đến khi hàm mất mát đạt đến giá trị tối thiểu hoặc một điều kiện dừng khác được đáp ứng.

1.2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của thuật toán Gradient Descent (GD) truyền thống, được sử dụng rộng rãi trong huấn luyện các mô hình học máy và học sâu. Nó mang lại hiệu quả cao đặc biệt khi làm việc với lượng dữ liệu lớn. Dưới đây là một cái nhìn chi tiết về SGD:

1.2.2.1 Khái niệm Cơ Bản

- Mục tiêu: Tương tự như GD, SGD cũng nhằm mục đích tối ưu hóa hàm mất mát của mô hình bằng cách điều chỉnh tham số mô hình.
- Phương pháp: Thay vì sử dụng toàn bộ dữ liệu để tính toán gradient cho mỗi lần cập nhật như trong GD, SGD cập nhật tham số mô hình dựa trên gradient của mỗi mẫu dữ liệu hoặc một nhóm nhỏ các mẫu.

1.2.2.2 Cơ Chế Hoạt Động

- Khởi Tạo: Tham số mô hình được khởi tạo ngẫu nhiên.
- Chọn Mẫu: Trong mỗi lần lặp, chọn ngẫu nhiên một mẫu (hoặc một nhóm nhỏ các mẫu) từ tập dữ liệu.
- Tính Gradient: Tính gradient của hàm mất mát dựa trên mẫu (hoặc nhóm mẫu) đã chọn.

- Cập Nhật Tham Số: Cập nhật tham số mô hình sử dụng công thức:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

Trong đó $x^{(i)}, y^{(i)}$ là mẫu dữ liệu được chọn.

- Lặp lại: Lặp lại quá trình trên cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp nhất định, sự cải thiện của hàm mất mát dưới một ngưỡng nhất định).

1.2.3 Mini-Batch Gradient Descent

Mini-Batch Gradient Descent là một biến thể của Gradient Descent (GD), kết hợp ưu điểm của cả Stochastic Gradient Descent (SGD) và Gradient Descent truyền thống. Nó thường được sử dụng trong huấn luyện các mô hình học máy và học sâu, đặc biệt khi làm việc với lượng dữ liệu lớn.

1.2.3.1 Khái Niệm Cơ Bản

- Mục Tiêu: Tối ưu hóa hàm mất mát của mô hình bằng cách cập nhật tham số dựa trên gradient.
- Phương Pháp: Tính toán gradient dựa trên một nhóm nhỏ các mẫu dữ liệu (gọi là "mini-batch") thay vì toàn bộ dữ liệu (GD) hoặc từng mẫu riêng lẻ (SGD).

1.2.3.2 Cơ Chế Hoạt Động

- Khởi Tạo: Các tham số mô hình được khởi tạo ngẫu nhiên.
- Chia Mini-Batch: Tập dữ liệu được chia thành các nhóm nhỏ (mini-batches). Kích thước của mỗi mini-batch thường là một siêu tham số được chọn trước.
- Vòng Lặp Huấn Luyện:
- Chọn Mini-Batch: Mỗi lần lặp, một mini-batch được chọn ngẫu nhiên từ tập dữ liệu.
- Tính Gradient: Tính gradient của hàm mất mát dựa trên dữ liệu trong mini-batch hiện tại.

- Cập Nhật Tham Số: Tham số mô hình được cập nhật dựa trên gradient tính được:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; X_{\text{mini-batch}}, Y_{\text{mini-batch}})$$

- Lặp Lại Quá Trình: Quá trình trên được lặp lại cho đến khi đạt được điều kiện dừng (số lần lặp nhất định, sự cải thiện hàm mất mát dưới ngưỡng xác định, v.v.).

1.2.4 Momentum

Phương pháp Momentum là một cải tiến của thuật toán Gradient Descent, được thiết kế để tăng tốc độ học và giảm thiểu việc mắc kẹt tại các điểm cực tiểu địa phương trong quá trình tối ưu hóa. Nó được lấy cảm hứng từ vật lý, cụ thể là từ khái niệm động lượng trong chuyển động học.

1.2.4.1 Khái Niệm Cơ Bản

- Mục Tiêu: Tối ưu hóa hàm mất mát của mô hình một cách nhanh chóng và hiệu quả hơn so với Gradient Descent truyền thống.
- Ý Tưởng Chính: Tích lũy gradient của các bước trước đó để xác định hướng và tốc độ cập nhật tham số, giúp "đẩy" quá trình tối ưu hóa vượt qua các điểm cực tiểu địa phương.

1.2.4.2 Cơ Chế Hoạt Động

- Khởi Tạo: Các tham số mô hình và vectơ động lượng được khởi tạo (vectơ động lượng thường khởi tạo bằng vectơ không).
- Tính Gradient: Tính gradient của hàm mất mát tại tham số mô hình hiện tại.
- Cập Nhật Động Lượng: Cập nhật vectơ động lượng dựa vào gradient hiện tại và động lượng trước đó:

$$v = \gamma v - \eta \nabla_\theta J(\theta)$$

Trong đó:

- v: Vectơ động lượng.
- γ: Hệ số động lượng, thường nằm trong khoảng từ 0.9 đến 0.99.
- η: Tốc độ học.
- $J(\theta)$: Hàm mất mát.
- $\nabla_\theta J(\theta)$: Gradient của hàm mất mát theo θ.

- Cập Nhật Tham Số: Tham số mô hình được cập nhật sử dụng vectơ động lượng:

$$\theta = \theta + v$$

- Lặp Lại Quá Trình: Lặp lại các bước trên cho đến khi đạt được điều kiện dừng.

1.2.5 Adam (Adaptive Moment Estimation)

Phương pháp Optimizer Adam (Adaptive Moment Estimation) là một trong những thuật toán tối ưu hóa phổ biến và hiệu quả được sử dụng rộng rãi trong huấn luyện mô hình học sâu. Adam kết hợp các ưu điểm của hai thuật toán tối ưu hóa khác là Momentum và RMSprop, cung cấp cơ chế tự động điều chỉnh tốc độ học.

1.2.5.1 Khái Niệm Cơ Bản

- Mục Tiêu: Giảm thiểu hàm mất mát của mô hình bằng cách hiệu quả cập nhật tham số mô hình.
- Phương Pháp: Kết hợp tính toán moment đầu tiên (tương tự như Momentum) và moment thứ hai (tương tự như RMSprop) của gradient.

1.2.5.2 Cơ Chế Hoạt Động

- Khởi Tạo: Các tham số mô hình được khởi tạo. Đồng thời, khởi tạo hai vectơ m và v để lưu trữ trung bình động của gradient và bình phương gradient, tương ứng.

- Tính Gradient: Tính gradient của hàm mất mát tại tham số mô hình hiện tại.
- Cập Nhật Moment Đầu Tiên m: Cập nhật trung bình động của gradient (tương tự như Momentum).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t$$

Trong đó g_t là gradient tại thời điểm t, và β_1 là hệ số suy giảm.

- Cập Nhật Moment Thứ Hai v: Cập nhật trung bình động của bình phương gradient (tương tự như RMSprop).

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

- Hiệu Chỉnh Bias: Điều chỉnh moment cho trường hợp bias ban đầu do khởi tạo m và v bằng 0.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

- Cập Nhật Tham Số Mô Hình: Sử dụng moment đã hiệu chỉnh để cập nhật tham số.

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$$

Trong đó η là tốc độ học, và ϵ là một số nhỏ để tránh chia cho 0.

1.2.6 RMSprop

Optimizer RMSprop (Root Mean Square Propagation) là một phương pháp tối ưu hóa được thiết kế đặc biệt để huấn luyện các mạng nơ-ron sâu. Nó giải quyết một số vấn đề của thuật toán Stochastic Gradient Descent (SGD) thông qua việc điều chỉnh tốc độ học dựa trên một cách tiếp cận thích nghi.

1.2.6.1 Khái Niệm Cơ Bản

- Mục Tiêu: Tối ưu hóa hàm mất mát của mô hình bằng cách điều chỉnh tốc độ học một cách thích nghi.

- Phương Pháp: Tính toán trung bình động của bình phương gradient và sử dụng nó để điều chỉnh tốc độ học cho mỗi tham số mô hình.

1.2.6.2 Cơ Chế Hoạt Động

- Khởi Tạo: Các tham số mô hình được khởi tạo. Ngoài ra, một vectơ v được khởi tạo để lưu trữ trung bình động của bình phương gradient.
- Tính Gradient: Tính gradient của hàm mất mát tại tham số mô hình hiện tại.
- Cập Nhật Trung Bình Động của Bình Phương Gradient: Vectơ v được cập nhật dựa trên gradient mới:

$$v_t = \beta v_{t-1} + (1 - \beta) \cdot g_t^2$$

Trong đó g_t là gradient tại thời điểm t, và β là hệ số suy giảm (thường được chọn là 0.9).

- Cập Nhật Tham Số Mô Hình: Sử dụng trung bình động này để điều chỉnh tốc độ học cho mỗi tham số:

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

1.3 So sánh các phương pháp Optimizer

Optimizers	Ưu Điểm	Nhược Điểm	Ứng Dụng
Gradient Descent (GD)	Đơn giản và dễ hiểu.	Không hiệu quả với dữ liệu lớn; có nguy cơ mắc kẹt tại điểm cực tiểu địa phương.	Thích hợp với các bộ dữ liệu nhỏ.
Stochastic Gradient Descent (SGD)	Hiệu quả với dữ liệu lớn; có khả năng thoát khỏi	Có thể gây ra sự dao động lớn trong quá trình học.	Thích hợp với dữ liệu lớn; thường cần nhiều epoch để hội tụ.

	cực tiểu địa phương.		
Mini-Batch Gradient Descent	Cân bằng giữa hiệu quả tính toán của GD và tính linh hoạt của SGD.	Cần chọn kích thước mini-batch phù hợp.	Phổ biến trong học sâu; thích hợp cho hầu hết các loại dữ liệu.
Momentum	Tăng tốc quá trình học và giảm dao động.	Có thể vượt qua điểm tối ưu.	Thường được kết hợp với SGD; hiệu quả trong học sâu.
Adam (Adaptive Moment Estimation)	Hiệu suất cao, ổn định và nhanh chóng hội tụ.	Cần chọn siêu tham số phù hợp.	Rất phổ biến trong học sâu, đặc biệt là với dữ liệu lớn và mạng nơ-ron phức tạp.
RMSprop	Giảm bớt sự dao động trong quá trình học; thích nghi tốt với các thay đổi của gradient.	Cũng cần chọn siêu tham số phù hợp.	Hiệu quả trong học sâu, đặc biệt là khi có gradient thay đổi lớn.

CHƯƠNG 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.

2.1 Continual Learning

Continual Learning, còn được biết đến với các tên gọi khác như Lifelong Learning hoặc Incremental Learning, là một lĩnh vực nghiên cứu trong học máy và trí tuệ nhân tạo. Mục tiêu của Continual Learning là phát triển các mô hình có khả năng học liên tục từ dữ liệu mới mà không quên những kiến thức đã học trước đó. Điều này đối lập với cách tiếp cận truyền thống, nơi mô hình chỉ được huấn luyện một lần trên một tập dữ liệu cố định.

2.1.1 Đặc Điểm của Continual Learning

- **Học Từ Dữ Liệu Liên Tục**: Mô hình được huấn luyện để thích ứng với dữ liệu mới, đồng thời duy trì kiến thức đã học từ dữ liệu cũ.
- **Giảm Vấn Đề Quên**: Một trong những thách thức lớn nhất trong Continual Learning là vấn đề "catastrophic forgetting", nghĩa là mô hình quên đi kiến thức cũ khi học kiến thức mới.
- **Chia Sẻ Kiến Thức**: Kỹ năng hoặc thông tin học được từ một tác vụ có thể được áp dụng cho các tác vụ khác.

2.1.2 Các Phương Pháp Tiếp Cận

- **Regularization Based Methods**: Sử dụng kỹ thuật regularization để ngăn chặn việc quên kiến thức cũ. Ví dụ: Elastic Weight Consolidation (EWC) giúp "bảo vệ" trọng số quan trọng cho tác vụ cũ.
- **Architecture Based Methods**: Thay đổi kiến trúc mô hình để thích nghi với dữ liệu mới mà không làm mất kiến thức cũ. Ví dụ: Progressive Neural Networks sử dụng các cột mạng nơ-ron riêng biệt cho mỗi tác vụ mới.

- **Rehearsal Based Methods**: Kết hợp dữ liệu mới với một số dữ liệu cũ khi huấn luyện, giúp mô hình không quên kiến thức trước đó.

2.1.3 Ứng Dụng

- **Ứng Dụng Thực Tế**: Có thể được áp dụng trong nhiều lĩnh vực như robot tự học, hệ thống đề xuất, phân tích dữ liệu trực tuyến và nhiều ngữ cảnh khác nơi dữ liệu không ngừng thay đổi và phát triển.
- **Hệ Thống Tự Điều Chỉnh**: Hệ thống có khả năng tự cải thiện và thích nghi với sự thay đổi của môi trường mà không cần can thiệp thủ công.

2.1.4 Thách Thức

- **Quản Lý Dữ Liệu**: Quản lý và xử lý lượng lớn dữ liệu từ nhiều nguồn khác nhau.
- **Đánh Giá Hiệu Suất**: Đánh giá hiệu suất mô hình trong môi trường động và phức tạp.
- **Cân Bằng giữa Học Mới và Quên Cũ**: Tìm cách cân bằng giữa việc tiếp nhận kiến thức mới và không làm mất đi kiến thức cũ.

Continual Learning đang mở ra những khả năng mới trong lĩnh vực học máy, cho phép tạo ra các mô hình linh hoạt và thích ứng với sự thay đổi liên tục của dữ liệu và môi trường.

2.2 Test Production

"Test Production" trong bối cảnh xây dựng giải pháp học máy thường ám chỉ quá trình kiểm thử mô hình học máy trong một môi trường sản xuất thực tế hoặc môi trường gần giống sản xuất. Đây là một bước quan trọng để đảm bảo rằng mô hình hoạt động đúng đắn và hiệu quả khi được triển khai thực tế.

2.2.1 Các Bước Trong Test Production

- **Triển Khai Mô Hình Trong Môi Trường Thử Nghiệm**: Đầu tiên, mô hình được triển khai trong một môi trường kiểm thử, nơi nó có thể tương tác với dữ liệu thực tế hoặc dữ liệu giả lập môi trường thực.
- **Kiểm Thử Tích Hợp**: Đảm bảo rằng mô hình học máy hoạt động đúng đắn khi được tích hợp với các hệ thống khác, như cơ sở dữ liệu, front-end applications, hoặc các API.
- **Kiểm Tra Hiệu Suất**: Đánh giá hiệu suất của mô hình dựa trên các chỉ số thực tế như độ chính xác, thời gian phản hồi, và tài nguyên sử dụng.
- **Phát Hiện và Xử Lý Lỗi**: Theo dõi mô hình để phát hiện và xử lý các lỗi không lường trước, bao gồm cả lỗi kỹ thuật và lỗi về dữ liệu.
- **Cập Nhật và Tinh Chỉnh Mô Hình**: Dựa trên kết quả kiểm thử, mô hình có thể cần được cập nhật hoặc tinh chỉnh để cải thiện hiệu suất hoặc khắc phục các vấn đề phát hiện được.

2.2.2 Tại Sao Test Production Lại Quan Trọng

- **Đánh Giá Hiệu Suất Thực Tế**: Kiểm tra xem mô hình có hoạt động hiệu quả trong môi trường thực hay không, không chỉ dựa trên dữ liệu kiểm thử.
- **Phát Hiện Lỗi Tiềm Ẩn**: Các lỗi hoặc vấn đề không dễ nhận biết trong quá trình phát triển có thể được phát hiện khi mô hình đối mặt với dữ liệu thực tế hoặc điều kiện môi trường không lường trước.
- **Đảm Bảo Tích Hợp**: Đảm bảo rằng mô hình hoạt động tốt với các hệ thống và quy trình khác trong môi trường sản xuất.

2.2.3 Thách Thức

- **Mô Phỏng Dữ Liệu Thực Tế**: Khó khăn trong việc tạo ra hoặc thu thập dữ liệu thử nghiệm phản ánh chính xác môi trường sản xuất.
- **Quản Lý Thay Đổi**: Mô hình có thể cần được cập nhật liên tục để thích nghi với sự thay đổi của dữ liệu hoặc yêu cầu kinh doanh.

- **Bảo Mật và Quyền Riêng Tư**: Đảm bảo rằng quá trình kiểm thử tuân thủ các quy định về bảo mật và quyền riêng tư, đặc biệt khi làm việc với dữ liệu nhạy cảm.

Test Production là một bước không thể thiếu trong quá trình phát triển giải pháp học máy, giúp đảm bảo rằng mô hình không chỉ hoạt động tốt trong môi trường kiểm thử mà còn đối mặt hiệu quả với thách thức của thế giới thực.