TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN**

**MACHINE LEARNING**

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

*Người thực hiện*: **TRẦN HỮU KHÁNH – 520H0542**

Lớp **: 20H50202**

Khoá  **: 24**

**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**



**BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN**

**MACHINE LEARNING**

*Người hướng dẫn*: **ThS LÊ ANH CƯỜNG**

*Người thực hiện*: **TRẦN HỮU KHÁNH – 520H0542**

Lớp **: 20H50202**

Khoá  **: 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

I would like to express my sincere gratitude to Mr. Lê Anh Cường for creating conditions for me to conduct and complete this final report in a fair, transparent and best manner, and I also thank my friends who helped me a little bit in designing this report and more complete to complete this part of the report.

In the process of making and completing this report, errors are inevitable. At the same time, due to the limited theoretical level as well as the experience of making reports, it is inevitable that there will be shortcomings, I hope that the subject teachers will judge the time and fairness so that the report can get the best score.

Thank you sincerely!

*Hồ Chí Minh, December, 19th 2023*

*Author*

*Khanh*

*Tran Huu Khanh*

**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 / chúng tôi và được sự hướng dẫn của ThS. 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).

*Hồ Chí Minh, December, 19th 2023*

*Author*

*Khanh*

*Tran Huu Khanh*

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)

TÓM TẮT

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc154267836)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc154267837)

[TÓM TẮT iv](#_Toc154267838)

[MỤC LỤC 1](#_Toc154267839)

[PART 1: TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY 3](#_Toc154267840)

[I. Optimizer là gì? 3](#_Toc154267841)

[II. Các phương pháp Optimizer trong huấn luyện mô hình học máy? 3](#_Toc154267842)

[III. So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy? 15](#_Toc154267843)

[PART 2: CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY 17](#_Toc154267844)

[I. Continual Learning 17](#_Toc154267845)

[II. Test Production 20](#_Toc154267846)

[TÀI LIỆU THAM KHẢO 23](#_Toc154267847)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

PART 1: TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1. Optimizer là gì?

* Optimizer (thuật toán tối ưu) là cách để tối ưu hóa và cập nhật các trọng số của mô hình dựa trên dữ liệu huấn luyện. Nó đóng vai trò quan trọng trong việc điều chỉnh mô hình để nó học và dự đoán tốt hơn.
* Mỗi mô hình học máy có các tham số hay trọng số cần được điều chỉnh để cải thiện hiệu suất dự đoán. Công việc của optimizer là tìm ra các giá trị tối ưu cho các tham số này dựa trên hàm mất mát (loss function) và dữ liệu huấn luyện.
* Các phương pháp optimizer khác nhau có thể áp dụng các thuật toán khác nhau để tối ưu hóa mô hình.

1. Các phương pháp Optimizer trong huấn luyện mô hình học máy?
2. Gradient Descent
   1. Định nghĩa:

* Gradient Descent là một thuật toán tối ưu hóa được sử dụng trong học máy để giảm thiểu hàm chi phí hoặc hàm mất mát của mô hình. Mục tiêu của nó là tìm ra các tham số tối ưu (trọng số và bias) cho một mô hình sao cho giảm thiểu sai số giữa giá trị dự đoán và giá trị thực tế. Quá trình này bao gồm việc điều chỉnh các tham số này theo hướng giảm dốc (ngược với đạo hàm) của hàm chi phí.
  1. Các thành phần chính:
* Objective Function: Hàm cần được giảm thiểu hoặc làm tối đa hóa trong quá trình huấn luyện mô hình học máy.
* Gradient: đại diện cho đạo hàm bậc nhất của hàm mất mát theo các tham số của mô hình. Nó thể hiện hướng và tốc độ tăng hoặc giảm mạnh nhất của hàm tại một điểm cụ thể trong không gian tham số.
  1. Cách hoạt động:
* B1 - Khởi tạo: Bắt đầu với một ước lượng ban đầu cho các tham số.
* B2 - Tính Toán Độ Dốc: Tính toán độ dốc của hàm mục tiêu theo các tham số tại điểm hiện tại.
* B3 - Cập Nhật Tham Số: Di chuyển theo hướng ngược lại của độ dốc để giảm giá trị của hàm mục tiêu. Việc cập nhật này được thực hiện lặp lại dựa trên một learning rate (step size).
* B4 - Lặp lại các bước 2 và 3 cho đến khi điều kiện dừng được đáp ứng.
  1. Mục tiêu:
* Mục tiêu chính của Gradient Descent là tìm ra các tham số để tối thiểu hóa hoặc tối đa hóa một hàm chi phí hoặc mất mát, giúp các mô hình học và cải thiện dự đoán hoặc phù hợp tốt hơn với dữ liệu không biết trước.
  1. Ưu và nhược điểm của thuật toán:
* Ưu điểm:
* Ứng Dụng Rộng Rãi: Có thể áp dụng cho nhiều bài toán tối ưu hóa, bao gồm hồi quy tuyến tính, mạng neural và các mô hình phức tạp khác.
* Đa Dạng: Hoạt động với các hàm có khả năng khả vi, cho phép tối ưu hóa trong các lĩnh vực khác nhau.
* Hiệu Quả: Gradient Descent là một trong những phương pháp phổ biến và hiệu quả để tối ưu hóa hàm mất mát trong học máy và học sâu.
* Tính Toán Đơn Giản: Các phép toán trong Gradient Descent tương đối đơn giản và có thể thực hiện hiệu quả.
* Tính khả thi: Dễ dàng triển khai
* Nhược điểm:
* Learning Rate: Việc chọn learning rate thích hợp rất quan trọng. Learning Rate quá lớn có thể dẫn đến phân kỳ, trong khi quá nhỏ có thể làm chậm quá trình hội tụ.
* Initialization: Giá trị tham số ban đầu có thể ảnh hưởng đến quá trình hội tụ và kết quả cuối cùng.
* Dữ liệu huấn luyện lớn: Gradient Descent thường đòi hỏi một lượng lớn dữ liệu để huấn luyện hiệu quả, đặc biệt đối với mô hình phức tạp.
* Nhạy Cảm Với Dữ Liệu Nhiễu: Dữ liệu nhiễu có thể làm giảm hiệu suất của Gradient Descent.

1. Stochastic Gradient Descent
   1. Định nghĩa:

* SGD là một biến thể của Gradient Descent, thay vì tính toán gradient trên toàn bộ tập dữ liệu, nó sử dụng chỉ một điểm dữ liệu hoặc một mini-batch nhỏ để tính gradient trong mỗi lần cập nhật tham số.
* SGD cho phép tốc độ học nhanh hơn vì nó không cần phải xử lý toàn bộ dữ liệu trong mỗi vòng lặp, làm giảm thời gian tính toán.
  1. Các thành phần chính:
* Tính ngẫu nhiên: SGD đưa ra sự ngẫu nhiên bằng cách chọn một tập con nhỏ của mẫu huấn luyện (mini-batch) tại mỗi vòng lặp để tính toán gradient. Tính ngẫu nhiên này giúp tránh rơi vào các điểm cực tiểu cục bộ và cung cấp sự hội tụ nhanh hơn so với gradient descent thông thường.
* Gradient Calculation: Đối với mỗi mini-batch, gradient của hàm mất mát theo các thông số của mô hình được tính toán. Gradient này biểu thị hướng và độ lớn thay đổi cần thiết để làm giảm thiểu mất mát.
* Cập nhật Tham số: Sau khi tính toán gradient cho mini-batch, SGD cập nhật các tham số của mô hình bằng thông tin gradient này. Nó điều chỉnh các tham số theo hướng ngược lại với gradient, được tỉ lệ bởi tỷ lệ học, để giảm thiểu hàm mất mát theo từng vòng lặp.
* Learning Rate: Đây là một yếu tố quan trọng để kiểm soát quá trình hội tụ và ổn định của quá trình tối ưu hóa.
  1. Cách hoạt động:
* B1 - Khởi Tạo: Bắt đầu với các giá trị ban đầu cho các tham số cần tối ưu hóa.
* B2 - Chọn Ngẫu Nhiên Dữ Liệu: Mỗi lần lặp, chọn ngẫu nhiên một điểm dữ liệu hoặc một mini-batch từ tập dữ liệu huấn luyện.
* B3 - Tính Toán Gradient: Tính toán gradient của hàm mất mát (loss function) chỉ dựa trên điểm dữ liệu hoặc mini-batch đã chọn.
* B4 - Cập Nhật Tham Số: Sử dụng gradient tính được để cập nhật các tham số theo quy tắc cập nhật của Gradient Descent.
* B5 - Lặp lại quá trình từ bước 2 đến bước 4 cho đến khi điều kiện dừng được đáp ứng.
  1. Mục tiêu:
* Mục tiêu chính của Stochastic Gradient Descent (SGD) là tối ưu hóa các tham số của mô hình dựa trên dữ liệu huấn luyện để đạt được giảm thiểu của hàm mất mát.
  1. Ưu điểm và nhược điểm của thuật toán:
* Ưu điểm:
* Tính Tính Toán Thấp Hơn: So với Gradient Descent thông thường, SGD thường nhanh hơn vì nó chỉ sử dụng một mẫu dữ liệu hoặc một số mẫu dữ liệu nhỏ để cập nhật trọng số.
* Phù Hợp với Dữ Liệu Lớn: Thích hợp với các tập dữ liệu lớn do không yêu cầu tính toán trên toàn bộ tập dữ liệu.
* Tính Đa Dạng: SGD có thể thoải mái vượt qua các điểm tối ưu cục bộ và di chuyển đến các khu vực tốt hơn trong không gian tham số.
* Có Thể Dừng Bất Kỳ Lúc Nào: Có thể dừng thuật toán bất cứ lúc nào và lấy mô hình hiện tại làm kết quả.
* Nhược điểm:
* Bất Ổn Định: SGD có thể không ổn định và dao động nhiều hơn trong quá trình học.
* Khó Điều Chỉnh learning rate: Learning rate có thể khó điều chỉnh vì nó có thể ảnh hưởng đến sự hội tụ của mô hình.
* Cần Quan Tâm Đến Dữ Liệu Nhiễu: SGD có thể nhạy cảm với dữ liệu nhiễu, có thể ảnh hưởng đến quá trình học.
* Dễ Dẫn Đến Bão Hòa: Khi đạt tới điểm tối ưu, SGD có thể dễ bão hòa nhanh hơn so với các phương pháp khác.

1. Momentum
   1. Định nghĩa:

* Momentum là một kỹ thuật nhằm giảm độ dao động và tăng tốc độ hội tụ trong quá trình tối ưu hóa.
* Hạn chế hiện tượng quả bóng điều hòa (oscillation) khi Gradient Descent di chuyển chậm qua các vùng hẹp hoặc lồi.
  1. Các thành phần chính:
* Momentum: Là một tham số (thường là một số từ 0 đến 1) đại diện cho tỉ lệ mà đà của bước cập nhật trước đó được giữ lại. Nó quyết định mức độ mà gradient của bước cập nhật trước đó được tính vào bước cập nhật hiện tại.
* Update Step: Là việc cập nhật trọng số của mô hình dựa trên gradient hiện tại và đà từ các bước cập nhật trước đó. Công thức cập nhật được điều chỉnh để tính đến cả gradient hiện tại và đà từ các bước trước, do đó, tốc độ hội tụ có thể tăng lên.
  1. Cách hoạt động:
* Lưu trữ Momentum: Thay vì chỉ dựa vào gradient hiện tại để cập nhật trọng số, Momentum lưu giữ một phần của động lượng từ các bước cập nhật trước đó thông qua tham số momentum.
* Kết hợp Momentum vào cập nhật: Khi thực hiện bước cập nhật tiếp theo, Gradient Descent sẽ kết hợp momentum từ các bước cập nhật trước với gradient hiện tại để xác định hướng và khoảng động lượng mà trọng số cần thay đổi.
* Tăng Tốc Độ Hội Tụ: Việc kết hợp đà giúp tăng tốc độ của quá trình hội tụ bằng cách cho phép mô hình vượt qua các vùng địa phương và hướng tới giải pháp tối ưu toàn cục một cách ổn định hơn.
  1. Mục tiêu:
* Mục tiêu của Momentum trong quá trình tối ưu hóa, đặc biệt là trong các phương pháp như Gradient Descent, là tăng tốc độ hội tụ đến một giải pháp tối ưu bằng cách sử dụng đà từ các bước cập nhật trước đó.
  1. Ưu và nhược điểm của thuật toán:
* Ưu điểm:
* Tăng Tốc Quá Trình Học: Momentum giúp tăng tốc độ học của thuật toán Gradient Descent bằng cách tích lũy thông tin về hướng và tốc độ cập nhật trọng số.
* Vượt Qua Được Điểm Tối Ưu Cục Bộ: Momentum giúp tránh được rơi vào các điểm tối ưu cục bộ bằng cách vượt qua chúng và di chuyển tới các điểm tối ưu toàn cục.
* Giảm Độ Dao Động: Đối với các hàm lỗi có nhiều dao động, momentum có thể giúp giảm bớt độ dao động và đưa thuật toán về đích nhanh hơn.
* Nhược điểm:
* Khả Năng Vượt Quá Đích Nhanh Chóng: Đôi khi, momentum có thể làm cho thuật toán vượt quá đích (global minimum) nhanh chóng, gây ra hiện tượng bất ổn hoặc overshooting.
* Yêu Cầu Điều Chỉnh Hyperparameters: Điểm mạnh của momentum cũng là một điểm yếu khi cần điều chỉnh thêm hyperparameters như hệ số momentum để có hiệu suất tốt nhất.
* Độ Phức Tạp: Thêm một thông số mới (hệ số momentum) có thể làm tăng độ phức tạp của thuật toán, đặc biệt trong việc điều chỉnh và tối ưu hóa.

1. Adaptive Gradient Descent
   1. Định nghĩa:

* AdaGrad điều chỉnh tỷ lệ học cho mỗi tham số một cách linh hoạt trong quá trình huấn luyện.
* Giảm bớt vấn đề chọn tỷ lệ học cố định bằng cách thích ứng dựa trên lịch sử của gradient cho mỗi tham số.
  1. Các thành phần chính:
* Tùy chỉnh Learning Rate: AdaGrad điều chỉnh Learning Rate cho mỗi tham số tại mỗi vòng lặp dựa trên lịch sử của gradient. Nó điều chỉnh Learning Rate riêng lẻ cho mỗi tham số, cung cấp các cập nhật nhỏ hơn cho các tham số xuất hiện thường xuyên và cập nhật lớn hơn cho các tham số ít xuất hiện.
* Tích Lũy Gradient Bình Phương: AdaGrad theo dõi tổng bình phương của gradient trong quá khứ cho mỗi tham số. Lịch sử tích lũy này của gradient trong mẫu số của công thức tỷ lệ học giúp trong việc điều chỉnh tỷ lệ học.
* Learning Rates được tính toán: Learning Rate cho mỗi tham số được tính bằng cách chia một tỷ lệ học ban đầu được định nghĩa trước cho căn bậc hai của tổng bình phương các gradient cho đến thời điểm hiện tại cho tham số đó. Việc tỷ lệ này cho phép cập nhật lớn hơn cho các tham số ít được cập nhật và cập nhật nhỏ hơn cho các tham số được cập nhật thường xuyên.
  1. Cách hoạt động:
* Tính Toán Tỷ Lệ Học Động: AdaGrad sử dụng lịch sử của gradient để điều chỉnh tỷ lệ học. Nó cập nhật tỷ lệ học cho từng tham số dựa trên bình phương của gradient đã tích lũy cho tham số đó.
* Chia Tỷ Lệ Học Tự Động: AdaGrad chia tỷ lệ học ban đầu cho căn bậc hai của tổng bình phương của gradient đã tích lũy cho đến thời điểm hiện tại. Điều này dẫn đến việc tỷ lệ học được điều chỉnh tự động, lớn hơn đối với các tham số có gradient ít biến động và nhỏ hơn đối với các tham số có gradient biến động mạnh.
* Tối Ưu Hóa Tính Toán: AdaGrad nhắm đến việc tối ưu hóa quá trình học bằng cách tự động điều chỉnh tỷ lệ học cho từng tham số dựa trên lịch sử của gradient. Điều này giúp ổn định quá trình học và tăng cường khả năng di chuyển trong không gian tham số.
  1. Mục tiêu: là tăng cường quá trình tối ưu hóa bằng cách tự động điều chỉnh tỷ lệ học, từ đó giúp quá trình học của mô hình học máy hội tụ một cách hiệu quả và ổn định, đặc biệt trong những trường hợp các tham số có biến đổi gradient khác nhau
  2. Ưu và nhược điểm của thuật toán:
* Ưu điểm:
* Tính Tương Thích Với Dữ Liệu Khó Huấn Luyện: AdaGrad tương đối hiệu quả khi huấn luyện trên dữ liệu có tính không đồng nhất, nhiễu, hoặc thưa (sparse) vì nó có khả năng điều chỉnh tự động learning rate tùy theo độ dốc của từng tham số.
* Tự Động Điều Chỉnh Learning Rate: AdaGrad tự động điều chỉnh learning rate cho mỗi tham số dựa trên tần suất xuất hiện của đạo hàm của nó. Điều này giúp tối ưu hóa hiệu quả hơn trong việc di chuyển đến vùng lõi của hàm mất mát.
* Tính Tính Toán Hiệu Quả: AdaGrad không yêu cầu quá nhiều thời gian để tìm kiếm learning rate tối ưu. Nó có thể giúp giảm bớt việc chọn thử nghiệm nhiều learning rate khác nhau.
* Nhược điểm:
* Learning Rate Tăng Theo Thời Gian: AdaGrad có thể gây ra vấn đề learning rate giảm quá nhanh theo thời gian, dẫn đến việc dừng quá sớm trong quá trình học.
* Khó Khắc Phục Vấn Đề Zeros: Trong một số trường hợp, AdaGrad có thể gây ra vấn đề "zeros", khiến cho các tham số không cập nhật và không học được từ dữ liệu.
* Độ Phức Tạp Về Tính Toán: AdaGrad yêu cầu lưu trữ độ dốc được tích lũy, điều này có thể làm tăng độ phức tạp về mặt tính toán và bộ nhớ.

1. RMSprop
   1. Định nghĩa:

* RMSprop là một thuật toán tối ưu hóa giải quyết vấn đề điều chỉnh tỷ lệ học trong gradient descent bằng cách điều chỉnh tỷ lệ học một cách riêng biệt cho mỗi tham số.
* Nó chia learning rate cho giá trị trung bình bình phương giảm dần theo cấp số nhân của mỗi tham số
  1. Các thành phần chính:
* Trung Bình Động theo Trọng Số Mũ (EWMA): RMSprop sử dụng EWMA để theo dõi trung bình động của gradient bình phương cho mỗi tham số. Việc tính toán trung bình động này tập trung vào gradient gần đây trong khi giảm trọng số của gradient cũ.
* Tăng Cường Learning rates: Nó điều chỉnh learning rates cho mỗi tham số bằng cách chia gradient hiện tại cho căn bậc hai của EWMA của gradient bình phương trong quá khứ. Cơ chế tăng cường này điều chỉnh tỷ lệ học cho từng tham số dựa trên lịch sử gradient của chúng.
* Chuẩn Hóa Gradient: RMSprop thực hiện chuẩn hóa gradient bằng cách chia gradient hiện tại cho một tỷ lệ xấp xỉ của gradient gần đây. Việc chuẩn hóa này giúp kiểm soát ảnh hưởng của gradient lớn đối với quá trình tối ưu hóa.
  1. Cách hoạt động:
* Trung Bình Bình Phương Gradient: RMSprop duy trì một trung bình động của bình phương gradient cho mỗi tham số.
* Tính Toán learning rate: Điều chỉnh tỷ lệ học cho mỗi tham số bằng cách chia nó cho căn bậc hai của trung bình bình phương gradient.
* Ngăn Chặn Gradient Phồng Lên: Giúp ngăn chặn tỷ lệ học trở nên quá lớn và gradient phồng lên trong mạng neural sâu.
  1. Mục tiêu: nhằm tăng cường tính ổn định và hiệu quả của quá trình tối ưu hóa trong học máy, đặc biệt khi mô hình đối mặt với các biến đổi độ lớn khác nhau của gradient.
  2. Ưu và nhược điểm của thuật toán
* Ưu điểm:
* Tối Ưu Hóa Tốt trên Dữ Liệu Khó Huấn Luyện: RMSprop thường hoạt động tốt trên dữ liệu không đồng nhất và có độ nhiễu cao, nhờ khả năng điều chỉnh learning rate tự động cho từng tham số.
* Hiệu Quả Trong Việc Kiểm Soát Learning Rate: RMSprop cải thiện vấn đề learning rate giảm quá nhanh trong AdaGrad bằng cách sử dụng trung bình bình phương độ dốc.
* Giảm Vấn Đề Zeros: RMSprop giúp giảm thiểu vấn đề "zeros" trong AdaGrad bằng cách sử dụng trung bình bình phương độ dốc thay vì độ dốc thực tế.
* Nhược điểm:
* Nhạy cảm với các siêu tham số: RMSprop vẫn cần sự điều chỉnh cẩn thận của các siêu tham số như learning rate và beta.
* Khó Điều Chỉnh Learning Rate Tối Ưu: Mặc dù đã cải thiện so với AdaGrad, RMSprop vẫn không phải là giải pháp hoàn hảo cho vấn đề learning rate giảm quá nhanh.
* Chi Phí Tính Toán: RMSprop vẫn yêu cầu nhiều tính toán hơn so với các phương pháp cải tiến khác như Adam.

1. Adam
   1. Định nghĩa:

* Adam là một thuật toán tối ưu hóa tính toán tỷ lệ học thích nghi cho mỗi tham số bằng cách lưu trữ cả hai moment đầu tiên và thứ hai của gradient.
* Adam kết hợp những mặt lợi của cả hai phương pháp tối ưu hóa dựa trên momentum và RMSprop.
  1. Các thành phần chính:
* Learning rate Thích Ứng: Adam điều chỉnh Learning rate cho mỗi tham số dựa trên trung bình của gradient trong quá khứ và bình phương của chúng. Cơ chế thích ứng này cho phép learning rate cá nhân cho các tham số khác nhau.
* Trung Bình Động Theo Trọng Số Mũ: sử dụng trung bình động theo trọng số mũ của gradient và bình phương của gradient cho mỗi tham số. Các trung bình động này giúp ước lượng các khoảng thời gian đầu và thứ hai của gradient.
* Hiệu Chỉnh Bias: Adam sử dụng cơ chế hiệu chỉnh bias trong các bước tối ưu hóa ban đầu để khắc phục bias về không trong việc ước lượng các khoảng thời gian.
* Tối Ưu momentum: Nó tích hợp momentum bằng cách duy trì một trung bình động bổ sung cho khoảng thời gian đầu của gradient. Điều này giúp tạo ra một quá trình tối ưu hóa ổn định và có hướng hơn.
* Cập Nhật Tham Số: Bước cập nhật bao gồm kết hợp tỷ lệ học thích ứng, đà và ước lượng của khoảng thời gian đầu và thứ hai của gradient để cập nhật các tham số của mô hình.
  1. Cách hoạt động:
* Ước Lượng Các Moment: Adam duy trì các ước lượng về mean (moment đầu tiên) và variance (moment thứ hai) của gradient.
* Hiệu Chỉnh Bias: Điều chỉnh bias trong việc ước lượng moment ở đầu quá trình huấn luyện.
* Cập Nhật Tham Số: Cập nhật tham số bằng cách sử dụng sự kết hợp của moment đầu tiên và thứ hai của gradient và tỷ lệ học được điều chỉnh động.
  1. Mục tiêu: nhằm cải thiện hiệu suất và khả năng hội tụ của quá trình tối ưu hóa, cung cấp một cơ chế linh hoạt để cập nhật các tham số của mô hình học máy một cách hiệu quả và ổn định.
  2. Ưu và nhược điểm của thuật toán:
* Ưu điểm:
* Hiệu Quả và Linh Hoạt: Adam kết hợp cả Momentum và RMSprop, giúp tận dụng ưu điểm của cả hai thuật toán và cải thiện hiệu suất tối ưu hóa.
* Tính Linh Hoạt Của Tham Số: Adam tự điều chỉnh learning rate cho từng tham số trong quá trình huấn luyện, giúp tối ưu hóa tỷ lệ học và tốc độ hội tụ.
* Hiệu Quả Trong Huấn Luyện Mạng Học Sâu: Adam thường hoạt động tốt trên mạng nơ-ron học sâu, giúp giảm thời gian huấn luyện và tăng độ chính xác của mô hình.
* Nhược điểm:
* Yêu Cầu Điều Chỉnh Siêu Tham Số: Adam vẫn cần sự điều chỉnh thủ công của các siêu tham số như beta1, beta2 và learning rate.
* Khó Kiểm Soát Tốt Khi Dữ Liệu Đa Dạng: Trong một số trường hợp, Adam có thể hoạt động không hiệu quả trên các dữ liệu có tính đồng nhất thấp hoặc nhiễu cao.
* Khó Khắc Phục Vấn Đề Về Bộ Nhớ: Trong một số trường hợp, Adam có thể tạo ra sự phụ thuộc lớn vào lịch sử gradient, dẫn đến vấn đề về bộ nhớ và ảnh hưởng đến tốc độ học.

1. So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy?

* Tiêu chí so sánh:
* Độ ổn định và dao động
* Hiệu quả tính toán
* Tính chất của tham số
* Khả năng vượt qua điểm tối ưu cục bộ

1. Độ ổn định và dao động

* Gradient Descent: Độ ổn định cao vì cập nhật dựa trên toàn bộ tập dữ liệu, nhưng có thể dao động nếu dữ liệu có nhiều nhiễu.
* Stochastic Gradient Descent: Thường có dao động lớn hơn do cập nhật dựa trên mẫu dữ liệu ngẫu nhiên, không ổn định hơn Gradient Descent.
* Momentum: Giảm độ dao động và tăng độ ổn định với momentum từ các bước trước.
* AdaGrad: Điều chỉnh tỷ lệ học cho các tham số khác nhau, có thể giảm độ dao động.
* RMSprop: Ngăn chặn độ phồng lên của gradient, cải thiện độ ổn định.
* Adam: Kết hợp cả momentum và tỷ lệ học thích nghi, giúp giảm độ dao động và tăng độ ổn định.

1. Hiệu quả tính toán

* Gradient Descent: Tính toán tốn kém với dữ liệu lớn, vì cập nhật trên toàn bộ tập dữ liệu.
* Stochastic Gradient Descent: Tính toán hiệu quả hơn với dữ liệu lớn, vì cập nhật trên mẫu dữ liệu nhỏ.
* Momentum: Tính toán không quá tốn kém, tương tự như Gradient Descent.
* AdaGrad: Tính toán đôi khi tốn kém vì phải tính toán trung bình bình phương gradient.
* RMSprop: Tính toán không quá tốn kém, hiệu quả so với AdaGrad.
* Adam: Tính toán không quá tốn kém, phù hợp với nhiều loại dữ liệu và mô hình.

1. Tính chất của tham số

* Gradient Descent, Stochastic Gradient Descent, Momentum: Cần chỉnh tham số learning rate.
* AdaGrad, RMSprop: Tự điều chỉnh tỷ lệ học theo gradient, ít cần điều chỉnh tham số.
* Adam: Cần điều chỉnh nhiều tham số siêu như tốc độ học và tham số beta.

1. Khả năng vượt qua điểm tối ưu cục bộ

* Gradient Descent: Dễ rơi vào điểm tối ưu cục bộ do không có yếu tố ngẫu nhiên trong quá trình cập nhật.
* Stochastic Gradient Descent: Có khả năng vượt qua điểm tối ưu cục bộ do sử dụng ngẫu nhiên dữ liệu.
* Momentum, AdaGrad, RMSprop, Adam: Có khả năng vượt qua điểm tối ưu cục bộ hơn Gradient Descent vì có yếu tố ngẫu nhiên hoặc điều chỉnh tỷ lệ học.
* Kết luận:
* Mỗi thuật toán có ưu điểm và nhược điểm riêng, và lựa chọn thuật toán tối ưu hóa thường phụ thuộc vào loại dữ liệu, bài toán cụ thể và việc điều chỉnh các siêu tham số.
* Stochastic Gradient Descent, RMSprop, Adam: Thường là lựa chọn phổ biến, đa năng và hiệu quả trong nhiều tình huống khác nhau.
* AdaGrad: Hiệu quả cho dữ liệu có độ dốc biến đổi.
* Momentum: Giảm độ dao động và tăng độ ổn định, phù hợp với các dữ liệu chứa nhiều điểm tối ưu cục bộ, điểm bằng phẳng hoặc khu vực có độ cong khác nhau.
* Gradient Descent: Sử dụng ít hơn trong các tình huống có dữ liệu lớn và phức tạp.

PART 2: CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY

1. Continual Learning
2. Continual Learning là gì?

* Continual Learning là ý tưởng cập nhật mô hình khi có dữ liệu mới xuất hiện; điều này giúp mô hình theo kịp với phân phối dữ liệu hiện tại.
* Một khi mô hình được cập nhật, nó không thể được tung ra mà không kiểm thử. Nó cần được kiểm thử để đảm bảo rằng nó an toàn và tốt hơn mô hình hiện tại.

1. Tại sao Continual Learning quan trọng?

* Thích ứng với môi trường động: trong các tình huống thực tế, phân phối dữ liệu có thể thay đổi theo thời gian. Continual Learning giúp các mô hình thích nghi và tiến triển với các mẫu mới, đảm bảo chúng luôn phù hợp và hiệu quả trong môi trường động.
* Tận Dụng Tài Nguyên Hiệu Quả: Huấn luyện lại mô hình từ đầu với mỗi tập dữ liệu mới có thể tốn kém về tài nguyên. Continual Learning cho phép mô hình học tập theo cách gia tăng, giảm thiểu việc huấn luyện lại một cách cơ bản và tối ưu hóa việc sử dụng tài nguyên.
* Tránh Quên Kiến Thức: Các mô hình truyền thống thường quên những kiến thức đã học khi được huấn luyện trên dữ liệu mới. Continual Learning nhằm giảm thiểu hiện tượng 'quên kiến thức đột ngột' bằng cách duy trì và tích hợp kiến thức trước đó khi học thông tin mới.
* Cải Thiện Suốt Đời: Continual Learning hỗ trợ việc học suốt đời bằng cách cho phép các mô hình tích luỹ kiến thức theo thời gian. Khả năng này quan trọng trong các ứng dụng đòi hỏi sự thích nghi và cải thiện liên tục.

1. Những thách thức của Continual Learning

* Catastrophic forgetting: Khi một mô hình học một nhiệm vụ mới, có thể ghi đè hoặc gây can thiệp vào các tham số quan trọng cho các nhiệm vụ trước đó, dẫn đến mất mát hiệu suất đáng kể.
* Hạn chế về bộ nhớ và tính toán: Học liên tục thường liên quan đến một luồng dữ liệu và nhiệm vụ lớn và đa dạng, có thể không vừa vào bộ nhớ hoặc sức mạnh tính toán của mô hình. Do đó, mô hình cần lưu trữ, truy cập và cập nhật thông tin liên quan một cách hiệu quả.
* Đánh giá và so sánh: Các phương pháp học liên tục khó đánh giá và so sánh, vì không có tiêu chuẩn hoặc chỉ số chuẩn nào bao quát được tất cả các khía cạnh của vấn đề. Các phương pháp khác nhau có thể có các sự đánh đổi khác nhau giữa sự ổn định và sự linh hoạt, hoặc giữa tổng quát trong nhiệm vụ và tổng quát giữa các nhiệm vụ.
* Ứng dụng trong thực tế: Các phương pháp học liên tục cần được mạnh mẽ và linh hoạt để thích nghi với môi trường thực tế động và phức tạp, như robot học, xử lý ngôn ngữ tự nhiên, thị giác máy tính và các hệ thống tự động.

1. Các giai đoạn của Continual Learning:

* Thu thập và Tiền Xử lý Dữ Liệu: Giai đoạn ban đầu này bao gồm việc thu thập dữ liệu đa dạng liên quan đến nhiệm vụ và tiền xử lý để đảm bảo chất lượng và tương thích với mô hình học. Thường bao gồm việc tăng cường dữ liệu, chuẩn hóa, làm sạch và cấu trúc hóa để làm cho dữ liệu phù hợp cho việc huấn luyện.
* Huấn Luyện và Thích Nghi Mô Hình: Ở đây, mô hình được huấn luyện trên tập dữ liệu ban đầu và có khả năng thích nghi với dữ liệu mới. Giai đoạn này tập trung vào việc đảm bảo mô hình có khả năng học và giữ lại kiến thức từ tập dữ liệu ban đầu trong khi cho phép việc học bổ sung khi có dữ liệu mới.
* Giữ Kiến Thức và Chuyển Giao: Học liên tục nhấn mạnh việc giữ lại kiến thức học từ các nhiệm vụ hoặc bộ dữ liệu trước đó trong khi kết hợp thông tin mới. Các kỹ thuật như regularization, rehearsal và transfer learning giúp bảo toàn và chuyển giao kiến thức từ kinh nghiệm trước đó sang các nhiệm vụ mới mà không quên đi.
* Đánh Giá và Điều Chỉnh: Việc đánh giá liên tục hiệu suất của mô hình và khả năng thích nghi với dữ liệu mới rất quan trọng. Cần điều chỉnh, điều chỉnh tinh chỉnh để đảm bảo tính hiệu quả của mô hình và ngăn chặn hiện tượng quên nghiêm trọng, một hiện tượng mà mô hình quên đi kiến thức từ quá khứ khi học thông tin mới.

1. Áp dụng vào xây dựng mô hình học máy

* Phân đoạn dữ liệu và huấn luyện liên tục: Chia dữ liệu thành các phân đoạn hoặc luồng dữ liệu nhỏ để huấn luyện mô hình một cách tuần tự. Mỗi phân đoạn có thể đại diện cho một giai đoạn mới hoặc dữ liệu mới.
* Học Bổ Sung (Incremental Learning): Cho phép mô hình học từ dữ liệu mới mà không ảnh hưởng đến kiến thức đã học từ dữ liệu cũ. Các kỹ thuật này đảm bảo rằng mô hình không quên hoặc mất đi kiến thức quan trọng.
* Transfer Learning: Sử dụng kiến thức học được từ các tác vụ hoặc dữ liệu trước đó để giúp việc học cho các tác vụ mới. Transfer learning giúp cải thiện hiệu suất học liên tục bằng cách chuyển giao kiến thức từ một tác vụ sang tác vụ khác.
* Regularization và Memory Replay: Sử dụng các kỹ thuật như regularization (chế độ hóa) và memory replay (tái hiện bộ nhớ) để giữ lại thông tin quan trọng từ dữ liệu cũ và giảm thiểu hiện tượng quên nghiêm trọng (catastrophic forgetting).
* Đánh giá và Kiểm tra Liên Tục: Thực hiện kiểm tra và đánh giá thường xuyên để đảm bảo rằng mô hình vẫn hoạt động hiệu quả khi học từ dữ liệu mới và không bị ảnh hưởng bởi việc quên kiến thức cũ.
* Trong thực tế, đôi khi cũng có thể cần thay đổi kiến ​​trúc mô hình (tức là lặp lại mô hình). Áp dụng học liên tục vào xây dựng mô hình học máy đòi hỏi có chiến lược cụ thể. Khi nào cần lặp lại mô hình thì cần phải chạy thí nghiệm trên các tác vụ cụ thể để tìm ra khi nào cần lặp lại mô hình và dữ liệu.

1. Test Production
2. What is Test Production?

* Trong machine learning, "Test production" thường đề cập đến quá trình tạo ra hoặc tạo các bộ dữ liệu kiểm thử hoặc các kịch bản để đánh giá hiệu suất của các mô hình học máy. Khi phát triển và huấn luyện một mô hình học máy, việc đánh giá hiệu suất của nó trên dữ liệu mà nó chưa nhìn thấy trong quá trình huấn luyện rất quan trọng. Đây là nơi các bộ dữ liệu kiểm thử có vai trò quan trọng.
* Các bộ dữ liệu kiểm thử này được tách riêng biệt so với dữ liệu huấn luyện và được sử dụng độc quyền để đánh giá hiệu suất của mô hình và khả năng tổng quát hóa của nó đối với dữ liệu mới, chưa nhìn thấy trước đó.

1. Thử nghiệm mô hình trong sản xuất:

* A/B Testing: Triển khai các phiên bản khác nhau của mô hình (phiên bản hiện tại so với phiên bản mới) để so sánh hiệu suất trong thời gian thực. Điều này giúp đánh giá tác động của các thay đổi đối với trải nghiệm người dùng hoặc các chỉ số kinh doanh.
* Giám sát và Ghi Nhật Ký: Triển khai giám sát liên tục để theo dõi hiệu suất của mô hình, phát hiện bất thường và ghi nhật ký dự đoán. Điều này đảm bảo mô hình hoạt động như mong đợi và cho phép phát hiện nhanh vấn đề.
* Shadow Mode Deployment: triển khai mô hình mới cùng với mô hình hiện tại mà không ảnh hưởng đến dự đoán thực tế. Điều này cho phép so sánh kết quả của cả hai mô hình mà không ảnh hưởng đến các quy trình thực tế.

1. Thử nghiệm trong chiến lược sản xuất:

* Canary Releases: Chiến lược này đưa ra phiên bản mới của ứng dụng cho một nhóm người dùng nhỏ (như một con chim canary trong một mỏ mài), sau đó theo dõi hiệu suất và lỗi trước khi triển khai rộng rãi.
* Shadow Traffic: Điều hướng một phần nhỏ của lưu lượng truy cập tới phiên bản mới để kiểm tra hiệu suất, nhưng không ảnh hưởng đến người dùng thực tế.
* Rollback Strategies: Có các kế hoạch đẩy ngược lại (rollback) tức thì nếu có vấn đề nghiêm trọng xảy ra trong quá trình triển khai.
* Gradual Rollouts (Phased Rollouts): Triển khai mới cho một phần nhỏ người dùng trước, sau đó tăng dần quy mô nếu không có vấn đề nào được phát hiện.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

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

[2] Sentayho, "Optimizer là gì và tại sao phải dùng nó," Sentayho, [Online]. Available: <https://sentayho.com.vn/optimizer-la-gi.html#Optimizer-la-gi-tai-sao-phai-dung-no>.

**Tiếng Anh**

[1] S. Rodriguez, "Continual Learning and Test in Production," GitHub, 2023. [Online]. Available: <https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md>.

[2] IBM, "Gradient descent is an optimization algorithm used to minimize some function," IBM, [Online]. Available: <https://www.ibm.com/topics/gradient-descent#:~:text=Gradient%20descent%20is%20an%20optimization,each%20iteration%20of%20parameter%20updates>.

[3] D2L.AI, "Gradient Descent," Dive into Deep Learning, [Online]. Available: <https://d2l.ai/chapter_optimization/gd.html>.

[4] "Optimizing Gradient Descent: Gradient Descent Variants," Ruder.io, [Online]. Available: <https://www.ruder.io/optimizing-gradient-descent/#gradientdescentvariants>.

[5] GeeksforGeeks, "Optimization Techniques for Gradient Descent," GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/optimization-techniques-for-gradient-descent/>.

[6] JavaTpoint, "Gradient Descent in Machine Learning," JavaTpoint, [Online]. Available: <https://www.javatpoint.com/gradient-descent-in-machine-learning>.

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn