

**TIỂU LUẬN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY**

**BÁO CÁO CUỐI KÌ**

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

*Người thực hiện*: **ĐÀO TƯỜNG LÂM QUYỀN ANH – 520H0509**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**



**TIỂU LUẬN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY**

**BÁO CÁO CUỐI KÌ**

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

*Người thực hiện*: **ĐÀO TƯỜNG LÂM QUYỀN ANH – 520H0509**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

# LỜI CẢM ƠN

Trước hết, tôi xin chân thành cảm ơn trường Đại học Tôn Đức Thắng, cũng như Đoàn Khoa Công nghệ Thông tin trường Đại học Tôn Đức Thắng đã tạo điều kiện tốt nhất để tôi có thể hoàn thành được bài tiểu luận này. Và tôi cũng xin gửi lời cảm ơn đến thầy Lê Anh Cường đã hướng dẫn tôi trong quá trình học tập và báo cáo cuối kì để tôi có thể đạt kết quả tốt nhất.

**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 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 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 12 tháng 05 năm 2022 Sinh viên 1*

*(ký tên và ghi rõ họ tên)*

*Trần Lê Thái Sơn Sinh viên 2*

*(ký tên và ghi rõ họ tên)*

*Đỗ Hoàng Duy Sinh viên 3*

*(ký tên và ghi rõ họ tên)*

*Đào Tường Lâm Quyền Anh*

# 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

[LỜI CẢM ƠN 1](#_Toc154258331)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 3](#_Toc154258332)

[MỤC LỤC 4](#_Toc154258333)

[CHƯƠNG 1 – SO SÁNH CÁC PHƯƠNG PHÁP OTIMIZER 5](#_Toc154258334)

[1.1 ) Optimizer là gì, tại sao phải dùng nó? 5](#_Toc154258335)

[1.2 ) Các phương pháp của otimizer 6](#_Toc154258336)

[CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION 9](#_Toc154258337)

# CHƯƠNG 1 – SO SÁNH CÁC PHƯƠNG PHÁP OTIMIZER

## 1.1 ) Optimizer là gì, tại sao phải dùng nó?

Trong ngữ cảnh của máy học và tối ưu hóa, "optimizer" (bộ tối ưu hóa) là một thành phần quan trọng của quá trình đào tạo mô hình máy học. Bộ tối ưu hóa giúp cập nhật các trọng số của mô hình sao cho hàm mất mát (hàm đo lường sự chênh lệch giữa đầu ra thực tế và đầu ra dự đoán) đạt được giá trị nhỏ nhất. Mục tiêu là tìm ra các trọng số tối ưu để mô hình có thể dự đoán đầu ra chính xác nhất cho dữ liệu đầu vào mới.

Dưới đây là một số lý do mà chúng ta cần sử dụng bộ tối ưu hóa trong quá trình đào tạo mô hình máy học:

* Tối ưu hóa trọng số: Mục tiêu của quá trình đào tạo là điều chỉnh các trọng số của mô hình để làm giảm thiểu hàm mất mát. Bộ tối ưu hóa giúp định rõ cách cập nhật các trọng số này theo chiều giảm độ dốc của hàm mất mát.
* Hạn chế tỷ lệ học (learning rate): Bộ tối ưu hóa thường đi kèm với một siêu tham số quan trọng là tỷ lệ học, quyết định đến mức độ cập nhật trọng số. Tối ưu hóa giúp điều chỉnh tỷ lệ học để đảm bảo quá trình học diễn ra một cách hiệu quả.
* Tìm kiếm không gian tham số: Bộ tối ưu hóa có thể giúp mô hình điều chỉnh các tham số một cách thông minh để tìm kiếm không gian tham số và đạt được giá trị tối ưu.
* Tránh quá mức đào tạo (overfitting): Một số bộ tối ưu hóa có các kỹ thuật như regularization để giảm nguy cơ quá mức đào tạo bằng cách kiểm soát sự phức tạp của mô hình.
* Hiệu suất tính toán: Bộ tối ưu hóa thường được tối ưu hóa để chạy mô hình trên các nguồn tài nguyên tính toán một cách hiệu quả.

## 1.2 ) Các phương pháp của otimizer

a) Gradient Descent (GD)

Đầu tiên, chúng ta muốn tìm giá trị nhỏ nhất của một hàm số trong các bài toán tối ưu. Thông thường, điểm đó là nơi mà đạo hàm của hàm số bằng 0. Tuy nhiên, trong nhiều trường hợp, việc tính toán đạo hàm trực tiếp có thể phức tạp hoặc thậm chí là không khả thi đối với các hàm số nhiều biến.

Đó là lý do chúng ta áp dụng phương pháp Gradient Descent (GD), hay giảm dần độ dốc. Ý tưởng chính của GD là bắt đầu từ một điểm ngẫu nhiên và sau đó di chuyển dần dần theo hướng ngược của độ dốc (đạo hàm) để tiến gần đến điểm cực tiểu.

Công thức cập nhật trong mỗi vòng lặp của GD là:

xnew = xold - learning\_rate \* gradient(xold)

Ưu điểm:

* Có thể đạt được giải pháp toàn cục với đạo hàm chính xác.
* Không yêu cầu nhiều bộ nhớ, vì chỉ cần tính toán đạo hàm toàn bộ dữ liệu.

Nhược điểm:

* Tính toán đạo hàm trên toàn bộ dữ liệu có thể tốn kém với dữ liệu lớn.
* Không phản ánh được sự biến động của dữ liệu.

b) Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của phương pháp Gradient Descent (GD) được sử dụng rộng rãi trong quá trình đào tạo mô hình máy học. Trong SGD, thay vì tính toán đạo hàm của toàn bộ dữ liệu đào tạo để cập nhật trọng số của mô hình, chúng ta chỉ sử dụng một mẫu dữ liệu ngẫu nhiên (hoặc một nhóm nhỏ mẫu, còn được gọi là mini-batch) để tính đạo hàm và thực hiện cập nhật trọng số.

SGD có một số ưu điểm quan trọng:

* Tính hiệu quả tính toán: So với GD, SGD yêu cầu ít tài nguyên tính toán hơn vì nó chỉ sử dụng một mẫu hoặc một nhóm nhỏ mẫu trong mỗi bước cập nhật.
* Khả năng làm việc với dữ liệu lớn: Đối với tập dữ liệu lớn, việc tính toán đạo hàm trên toàn bộ dữ liệu có thể rất tốn kém. SGD giúp giảm bớt gánh nặng tính toán bằng cách sử dụng mẫu ngẫu nhiên.
* Có thể thoát khỏi cực tiểu cục bộ: SGD, thông qua việc sử dụng mẫu ngẫu nhiên, có khả năng thoát khỏi cực tiểu cục bộ và tìm ra giải pháp tốt hơn.

Ưu điểm:

* Hiệu quả tính toán với dữ liệu lớn.
* Có thể thoát khỏi cực tiểu cục bằng sự ngẫu nhiên.

Nhược điểm:

* Dao động lớn và không ổn định.
* Có thể đạt được giải pháp cục bộ thay vì toàn cục.

c) Momentum

Momentum là một phương pháp tối ưu hóa được sử dụng trong quá trình đào tạo mô hình máy học, đặc biệt là trong việc cập nhật trọng số của mô hình trong các thuật toán tối ưu hóa gradient descent. Phương pháp này giúp cải thiện quá trình học bằng cách giảm bớt dao động và tăng tốc quá trình hội tụ đến giải pháp tối ưu.

Ý tưởng cơ bản của Momentum là tích hợp thông tin về độ dốc trước đó để ổn định và tăng tốc quá trình cập nhật trọng số. Trong mỗi bước lặp, nó giữ lại một phần của bước trước và thêm nó vào bước cập nhật hiện tại.

Hệ số momentum giúp làm giảm dao động trong quá trình tối ưu hóa và giữ cho quá trình di chuyển hướng xuống cực tiểu diễn ra mạnh mẽ và ổn định. Nếu đạo hàm có hướng giống với hướng trước đó, momentum tăng tốc bước đi và giúp vượt qua các điểm yên ngựa hoặc các vùng địa phương tối ưu.

Ưu điểm:

* Giảm dao động và giúp vượt qua các điểm yên ngựa.
* Tăng tốc quá trình hội tụ.

Nhược điểm:

* Cần điều chỉnh thêm siêu tham số (ví dụ: hệ số momentum).
* Có thể quá mức tăng tốc và vượt qua giải pháp tối ưu.

d) Adagrad

Adagrad là một thuật toán tối ưu hóa được sử dụng trong quá trình đào tạo mô hình máy học. Thuật toán này điều chỉnh tỷ lệ học (learning rate) của mỗi trọng số tùy thuộc vào lịch sử của các độ dốc trước đó. Cụ thể, Adagrad giảm tỷ lệ học của các trọng số có độ dốc lớn và tăng tỷ lệ học của các trọng số có độ dốc nhỏ. Điều này giúp làm giảm tỷ lệ học tự động thích ứng với từng trọng số.

Điều đặc biệt ở Adagrad là nó thích ứng với từng trọng số của mô hình dựa trên lịch sử của độ dốc. Tức là các trọng số có độ dốc lớn sẽ có tỷ lệ học giảm, trong khi các trọng số có độ dốc nhỏ sẽ có tỷ lệ học tăng. Điều này giúp làm giảm đi sự nhảy nhót của tỷ lệ học và tăng khả năng hội tụ của thuật toán.

Tuy nhiên, Adagrad cũng có một số hạn chế, chủ yếu là việc tích lũy bình phương của độ dốc có thể dẫn đến việc tỷ lệ học giảm quá mức và làm chậm quá trình học ở các giai đoạn sau. Để giải quyết vấn đề này, đã xuất hiện nhiều biến thể như RMSprop và Adam.

Ưu điểm:

* Tích hợp tỷ lệ học tự động thích ứng.
* Hiệu quả với dữ liệu có các đặc trưng thưa thớt.

Nhược điểm:

* Tích lũy quá mức thông tin và giảm tỷ lệ học quá nhanh.
* Có thể dẫn đến hiện tượng "vanishing learning rate".

e) RMSprop

RMSprop (Root Mean Square Propagation) là một thuật toán tối ưu hóa gradient descent được thiết kế để giảm độ nhạy của tỷ lệ học (learning rate) trong quá trình đào tạo mô hình máy học. RMSprop được phát triển để khắc phục một số hạn chế của Adagrad, đặc biệt là vấn đề của tỷ lệ học giảm quá mức.

So với Adagrad, RMSprop giữ được tính chất tự động thích ứng của tỷ lệ học, nhưng nó giảm tác động của các độ dốc cũ, giúp tránh tình trạng giảm quá mức của tỷ lệ học. Điều này làm cho RMSprop thích hợp hơn cho các bài toán mà có sự biến động lớn trong độ dốc.

Tuy nhiên, RMSprop vẫn có nhược điểm là có thể tích lũy quá nhiều thông tin về độ dốc và dẫn đến việc giảm tỷ lệ học quá mức ở các giai đoạn sau. Adam, một biến thể của RMSprop, đã được thiết kế để giải quyết một số vấn đề này.

Ưu điểm:

* Giảm tác động của độ dốc cũ, giúp tránh vấn đề "vanishing learning rate".
* Tự động thích ứng với độ dốc của mỗi trọng số.

Nhược điểm:

* Có thể tích lũy quá nhiều thông tin và giảm tỷ lệ học quá nhanh.
* Cần điều chỉnh thêm siêu tham số (ví dụ: hệ số giảm trọng số).

# CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION

## 2.1) Continial learning

Continual Learning (hoặc còn được gọi là Lifelong Learning) là một lĩnh vực quan trọng trong học máy, nơi mà mô hình được huấn luyện liên tục trên dữ liệu mới mà nó chưa từng thấy trước đó. Trong nhiều trường hợp, mô hình sẽ phải xử lý các nhiệm vụ mới mà không quên đi kiến thức đã học được từ các nhiệm vụ trước đó.

Thách thức chính của Continual Learning:

* Lãng quên (Catastrophic Forgetting): Mô hình có thể quên kiến thức đã học khi đối mặt với dữ liệu mới.
* Tích hợp kiến thức mới: Làm thế nào để tích hợp kiến thức mới mà không ảnh hưởng đến khả năng giữ kiến thức cũ.

Các phương pháp Continual Learning:

* Regularization: Sử dụng các phương pháp như Elastic Weight Consolidation để giảm thiểu ảnh hưởng đến các trọng số quan trọng của mô hình.
* Memory Networks: Sử dụng bộ nhớ ngoại để lưu trữ thông tin quan trọng từ các nhiệm vụ trước.
* Dynamic Architectures: Tạo ra mô hình có khả năng mở rộng để chứa kiến thức mới mà không làm ảnh hưởng đến kiến thức cũ.

## 2.1) Test production

Test Production là quá trình tạo ra và triển khai các bài kiểm tra để đánh giá hiệu suất của mô hình học máy. Điều này bao gồm việc xây dựng tập dữ liệu kiểm tra, lựa chọn các phương pháp đánh giá, và thực hiện các thử nghiệm để đảm bảo rằng mô hình hoạt động hiệu quả trên dữ liệu mới và không biết trước.

Quá trình Test Production thường bao gồm các bước sau:

* Xây dựng Tập Dữ Liệu Kiểm Tra: Tạo ra một tập dữ liệu độc lập từ tập dữ liệu mà mô hình chưa từng thấy để đánh giá khả năng tổng quát của nó.
* Lựa Chọn Phương Pháp Đánh Giá: Xác định các metric đánh giá phù hợp để đo lường hiệu suất của mô hình (ví dụ: độ chính xác, độ đo F1, mất mát).
* Thực Hiện Thử Nghiệm: Chạy mô hình trên tập dữ liệu kiểm tra và đo lường các metric đánh giá để đưa ra đánh giá về hiệu suất của mô hình.
* Phân Tích Kết Quả: Xem xét kết quả của thử nghiệm để hiểu rõ hơn về điểm mạnh và điểm yếu của mô hình, từ đó đề xuất cải tiến.