TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐINH HOÀNG PHÚC - 52100290**

**BÁO CÁO CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

**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**



**ĐINH HOÀNG PHÚC - 52100290**

**BÁO CÁO CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

Người hướng dẫn

**PSG.TS.LÊ ANH CƯỜNG**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Trong quá trình học tập và làm báo cáo, em đã tiếp thu được những kiến thức Nhập Môn Học Máy từ những tiết học trên lớp của PSG.TS Lê Anh Cường. Sự hỗ trợ trong việc tìm kiếm tài liệu của thầy để hoàn thành bài báo cáo và những kỹ năng làm bài. Em xin chân thành cảm ơn thầy thầy đã hỗ trợ cho em một cách tốt nhất để có thể hoàn thành bài báo cáo này. Cảm ơn thầy đã thông cảm cho những sai sót trong những tiết học. Tuy nhiên, kiếm thức và kỹ năng về môn học này của em vẫn còn nhiều hạn chế nên bài báo cáo của em không tránh khỏi những sai sót. Do vậy, em rất mong nhận đucợ sự góp ý của Thầy để em có điều kiện hoàn thiện hơn kiến thức của bản thân.

Em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày 18 tháng 12 năm 2023*

*Tác giả*

*Phuccc*

*Đinh Hoàng Phúc*

*(Ký tên và ghi rõ họ tên)*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS.TS Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự 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 Dự 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 18 tháng 12 năm 2023.*

*Tác giả*

*Phuccc*

*Đinh Hoàng Phúc*

*(Ký tên và ghi rõ họ tên)*

**TÓM TẮT**

(Time New Romans – 13)

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vi](#_Toc24513)

[DANH MỤC BẢNG BIỂU vii](#_Toc9187)

[DANH MỤC CÁC CHỮ VIẾT TẮT viii](#_Toc16430)

[CHƯƠNG 1. TÌM HIỂU VÀ SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY 1](#_Toc16998)

[1.1 Tổng quan về optimizer trong học máy. 1](#_Toc17125)

[1.1.1 Optimizer trong học máy là gì. 1](#_Toc18083)

[1.1.2 Cách thức Optimizer hoạt động. 1](#_Toc7080)

[1.2 Các phương pháp Optimizer phổ biến. 1](#_Toc27660)

[1.2.1 Gradient Descent. 2](#_Toc31729)

[1.2.2 Batch Gradient Descent. 2](#_Toc15040)

[1.2.3 Stochastic Gradient Descent. 3](#_Toc7627)

[1.2.4 Momentum. 5](#_Toc5496)

[1.2.5 Adagrad. 7](#_Toc3702)

[1.2.6 RMSProp. 9](#_Toc31770)

[1.2.7 Adam. 10](#_Toc32584)

[1.2.8 Adadelta 12](#_Toc20383)

[CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION 14](#_Toc22114)

[2.1 Continual Learning. 14](#_Toc2673)

[2.1.1 Continual Learning là gì? 14](#_Toc17966)

[2.1.2 Ưu điểm của việc học liên tục. 14](#_Toc14443)

[2.1.3 Các phương pháp học liên tục. 15](#_Toc10697)

[2.1.4 Các giai đoạn của continuous learning. 16](#_Toc5149)

[2.1.5 Các thách thức của học liên tục. 16](#_Toc5573)

[2.1.6 Các giải pháp cho việc học liên tục. 16](#_Toc6926)

[2.1.7 Các ứng dụng của việc học liên tục. 16](#_Toc18405)

[2.2 Test Production 16](#_Toc14592)

[2.2.1 Các loại kiểm tra. 16](#_Toc13754)

[2.2.2 Các phương pháp kiểm tra. 16](#_Toc27285)

[2.2.3 Thánh thức. 16](#_Toc14683)

[2.2.4 Lợi ích. 16](#_Toc31162)

[2.2.5 Bắt đầu từ đâu.TÀI LIỆU THAM KHẢO 16](#_Toc5245)

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

# DANH MỤC CÁC CHỮ VIẾT TẮT

# TÌM HIỂU VÀ SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

## Tổng quan về optimizer trong học máy.

### Optimizer trong học máy là gì.

- Optimizer là một hàm hoặc thuật toán điều chỉnh các thuộc tính của mạng nơ-ron, chẳng hạn như trọng số và tốc độ học tập. Chúng cho phép các neural networks học hỏi từ dữ liệu bằng cách cập nhật lặp đi lặp lại các trọng số và độ lệch. Các Optimizer biến bao gồm Stochastic Gradient Descent (SGD), Adam, and RMSprop.. Mỗi trình tối ưu hóa có các quy tắc cập nhật, tốc độ học tập và động lượng cụ thể để tìm các tham số mô hình tối ưu nhằm cải thiện hiệu suất.

- Vì vậy, nó giúp giảm tổn thất tổng thể và cải thiện độ chính xác. Vấn đề chọn trọng số phù hợp cho mô hình là một nhiệm vụ khó khăn, vì mô hình deep learning thường có rất nhiều tham số. Nó đặt ra nhu cầu chọn một thuật toán tối ưu hóa phù hợp mô hình.

### Cách thức Optimizer hoạt động.

- Iteration: Đưa các dữ liệu đã được huấn luyện vào model.

- Prediction: Mô hình nhận được một loạt dữ liệu (ví dụ: hình ảnh, văn bản) và đưa ra dự đoán dựa trên các tham số hiện tại(trọng số và độ lệch).

- Loss calculation: Hàm mất mát được tính toán, đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực tế.

- Parameter update: Optimizer sử dụng hàm tổn thất và các tham số của mô hình sau đó liên kết lại với nhau để cập nhật các tham số của mô hình theo hướng giảm thiểu tổn thất. Sự cập nhật này thường bao gồm việc thực hiện từng bước nhỏ theo hướng đi xuống dốc nhất của hàm mất mát.

## Các phương pháp Optimizer phổ biến.

### Gradient Descent.

- Để đánh giá mức độ tốt của mô hình, chúng ta sử dụng một hàm mất mát, đo lường sự khác biệt giữa dự đoán của mô hình và giá trị thực tế. Gradient descent giúp chúng ta tìm được bộ tham số (weights and biases) của mô hình sao cho giá trị của hàm mất mát là nhỏ nhất, tức là mô hình dự đoán chính xác nhất.

- Gradient descent là một thuật toán hoạt động bằng cách di chuyển theo hướng giảm dần của hàm mất mát (loss function), giống như một chiếc xe đang lao xuống đáy một thung lũng.

- Gradient Descent giúp tìm giá trị tiệm cận giá trị nhỏ nhất của hàm số f(x) bằng cách giảm dần giá trị các điểm x0 cho đến khi độ dốc hay đạo hàm tại điểm đầy đủ nhỏ chấp nhận được khi đó hàm f(x) sẽ tiệm cận giá trị nhỏ nhất. Việc giảm dần các giá trị này không phải là thực hiện một cách ngẫu nhiên mà phải giảm theo cách có quy tắc.

Công thức của Fradient Descent:

x = x - η∇x\*f(x)

Trong đó:

x là vector tham số của mô hình.

η là tốc độ học (learning rate)

∇x\*f(x) là vector độ dốc của hàm f(x) tại điểm x

- Cụ thể, chúng ta sẽ trừ đi một phần của gradient từ vector tham số hiện tại. Phần gradient mà chúng ta trừ đi được xác định bởi tốc độ học η. Khi η quá nhỏ, thuật toán sẽ mất nhiều bước để đi xuống dốc. Khi η quá lớn, thuật toán có thể nhảy qua điểm tối ưu.

- Có nhiều biến thể của Gradient Descent, bao gồm: Batch Gradient Descent, Stochastic Gradient Descent,…

### Batch Gradient Descent.

- Batch Gradient Descent hoạt động bằng cách lặp qua toàn bộ tập dữ liệu nhiều lần (còn gọi là epoch), tính toán độ dốc của hàm mất mát đối với tất cả các mẫu dữ liệu, và cập nhật tham số mô hình theo hướng giảm thiểu hàm mất mát đó.

- Độ dốc của một hàm tại một điểm là hướng mà hàm tăng nhanh nhất tại điểm đó. Gradient Descent sử dụng độ dốc để tìm hướng đi xuống dốc của hàm mất mát. Tốc độ học là một tham số kiểm soát mức độ nhanh chóng mà thuật toán sẽ đi xuống dốc.

Công thức của Batch Gradient Descent:

x = x - η∇xf(x)

Trong đó:

x là vector tham số của mô hình.

η là tốc độ học (learning rate)

∇x\*f(x) là vector độ dốc của hàm f(x) tại điểm x

- Ở mỗi bước của thuật toán Batch Gradient Descent, chúng ta sẽ tính gradient của toàn bộ tập dữ liệu. Gradient này sẽ cho chúng ta biết hướng mà hàm mất mát đang tăng nhanh nhất. Sau đó, chúng ta sẽ cập nhật vector tham số x theo hướng ngược lại của gradient..

- Ứng dụng của Batch Gradient Descent:

+ Linear Regression: Batch Gradient Descent được sử dụng để tìm các tham số của mô hình hồi quy tuyến tính.

+ Logistic Regression: Batch Gradient Descent được sử dụng để tìm các tham số của mô hình hồi quy logistic.

+ Support Vector Machines: Batch Gradient Descent được sử dụng để tìm các tham số của mô hình Support Vector Machines.

+ Neural Networks: Batch Gradient Descent được sử dụng để tìm các tham số của các mạng nơ-ron nhân tạo

### Stochastic Gradient Descent.

- Thay vì sử dụng toàn bộ tập huấn luyện thì Stochastic Gradient Descent (SGD) sẽ lấy ngẫu nhiên 1 phần tử ở tập huấn luyện và thực hiện tính lại vector độ dốc dựa chỉ dựa trên 1 điểm dữ liệu, sau đó lặp đi lặp lại đến khi kết thúc. Và việc tính toán dựa trên 1 điểm dữ liệu sẽ khiến thuật toán chạy nhanh hơn bởi có rất ít dữ liệu cần xử lý ở mỗi vòng lặp. Và điều này cũng giúp mô hình có thể được huấn luyện với những dữ liệu lớn vì mỗi vòng lặp chỉ cần đưa 1 điểm dữ liệu vào trong bộ nhớ.

 xt =xt-1  - η∇fx{t-1}(k)

Trong đó:

xt là tham số mô hình tại thời điểm t

Xt-1 là tham số mô hình tại thời điểm t - 1

η là bước học (learning rate)

∇fx{t-1}(k) là gradient của hàm mất mát f tại điểm dữ liệu k với tham số của x tại thời điểm {t-1}.

- Tham số mô hình được cập nhật theo hướng dốc âm của hàm mất mát tại điểm dữ liệu k. Bước học η điều chỉnh độ lớn của cập nhật.

- Ứng dụng của Stochastic Gradient Descent:

+ Huấn luyện mạng nơ ron: SGD là một thuật toán phổ biến để huấn luyện mạng nơ ron, bao gồm cả mạng nơ ron tích chập (CNN) và mạng nơ ron tái phát (RNN).

+ Huấn luyện các mô hình hồi quy: SGD cũng có thể được sử dụng để huấn luyện các mô hình hồi quy, chẳng hạn như mô hình đường hồi quy và mô hình dự đoán.

+ Huấn luyện các mô hình phân loại: SGD cũng có thể được sử dụng để huấn luyện các mô hình phân loại, chẳng hạn như mô hình phân loại logistic và mô hình hỗ trợ vector machine (SVM).

#### Ưu điểm.

- Nhanh hơn Batch Gradient Descent: SGD chỉ cần tính toán gradient cho một điểm dữ liệu tại một thời điểm, do đó nó nhanh hơn Batch Gradient Descent, đặc biệt là đối với các tập dữ liệu lớn.

- Ít tốn bộ nhớ: SGD không cần lưu trữ toàn bộ tập dữ liệu trong bộ nhớ cùng một lúc, do đó nó tiết kiệm bộ nhớ hơn Batch Gradient Descent.

- Khó bị mắc kẹt ở local minima: SGD có xu hướng ít bị mắc kẹt ở local minima hơn Batch Gradient Descent do nó cập nhật tham số theo hướng dốc âm tại mỗi điểm dữ liệu.

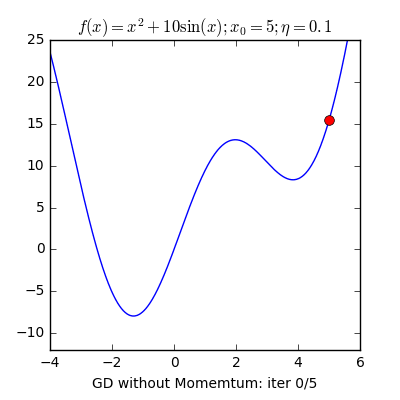
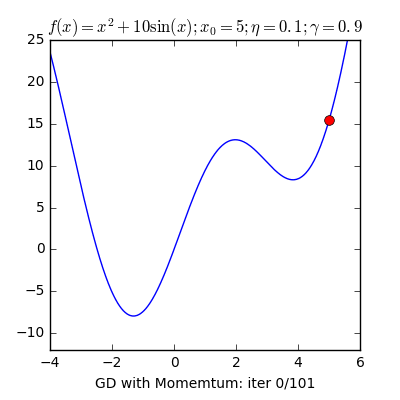
#### Nhược điểm.

- Dao động hơn Batch Gradient Descent: SGD có thể dao động hơn Batch Gradient Descent do nó chỉ cập nhật tham số theo hướng dốc âm tại một điểm dữ liệu tại một thời điểm.

- Yêu cầu điều chỉnh tham số: Việc lựa chọn bước học và các tham số khác của SGD có thể ảnh hưởng đến hiệu quả của SGD.

### Momentum.

- Momentum được sử dụng kết hợp với Gradient Descent để tăng tốc độ hội tụ của thuật toán. Giảm độ dốc dựa trên động lượng là một phương pháp tối ưu hóa giúp tăng cường giảm độ dốc thông thường bằng cách xem xét vận tốc của độ dốc. Trọng số của mạng nơ-ron được thay đổi theo độ dốc giảm dần dựa trên động lượng bằng cách sử dụng tổng trọng số của độ dốc trước đó và độ dốc hiện tại. Điều này làm cho các trọng số có thể chuyển hướng trôi chảy hơn, dẫn đến sự hội tụ nhanh hơn và có lẽ là kết quả tốt hơn.



- Hiểu một cách đơn giản thì Gradient Descent sẽ dùng kết hợp với Momentum khi mà nhiều điểm local minimum không mong muốn xuất hiện khi chúng ta xây dựng một mô hình nào đó. Để giải quyết vấn đề đó thì chúng ta sử dụng Momentum để tạo thêm bước đà và vượt qua được các điểm local minimum không mong muốn xuất hiện.

Công thức của Momentum:

vt = βvt-1 + η∇f(xt-1)

xt = xt-1 + vt

Trong đó:

v\_t là vector vận tốc tại thời điểm t

β là hệ số momentum

η là bước học (learning rate)

∇f(xt-1) là gradient của hàm mất mát J đối với tham số xt-1

- Vector vận tốc v\_t được cập nhật theo hướng dốc âm của hàm mất mát và hệ số momentum β. Tham số mô hình x\_t được cập nhật theo vector vận tốc v\_t..

- Ứng dụng của Momentum:

+ Huấn luyện mạng nơ ron: Momentum là một kỹ thuật phổ biến để huấn luyện mạng nơ ron, bao gồm cả mạng nơ ron tích chập (CNN) và mạng nơ ron tái phát (RNN).

+ Huấn luyện các mô hình hồi quy: Momentum cũng có thể được sử dụng để huấn luyện các mô hình hồi quy, chẳng hạn như mô hình đường hồi quy và mô hình dự đoán.

+ Huấn luyện các mô hình phân loại: Momentum cũng có thể được sử dụng để huấn luyện các mô hình phân loại, chẳng hạn như mô hình phân loại logistic và mô hình hỗ trợ vector machine (SVM).

#### Ưu điểm.

- Giảm dao động: Momentum giúp giảm dao động của SGD, dẫn đến quá trình học ổn định hơn.

- Tăng tốc hội tụ: Momentum có thể giúp tăng tốc quá trình hội tụ của SGD.

#### Nhược điểm.

- Yêu cầu điều chỉnh tham số: Hệ số momentum β là một tham số quan trọng cần được điều chỉnh để đảm bảo hiệu quả của Momentum.

### Adagrad.

- Thuật toán Adagrad được thiết kế để tự động điều chỉnh tốc độ học của mỗi tham số trong quá trình huấn luyện dựa trên lịch sử của các gradient trước đó.

- Cách thức hoạt động:

+ Tính toán gradient: Tại mỗi bước cập nhật, Adagrad tính toán gradient của hàm mất mát đối với các tham số mô hình.

+ Ước lượng độ lớn gradient: Adagrad lưu trữ một ước lượng độ lớn của gradient đối với từng tham số. Ước lượng này được cập nhật theo công thức sau:

Gt = Gt-1 + gt2

Trong đó:

Gt là ước lượng độ lớn của gradient tại thời điểm t

Gt-1 là ước lượng độ lớn của gradient tại thời điểm t-1

gt là gradient tại thời điểm t

+ Cập nhật tham số: Adagrad sử dụng ước lượng độ lớn của gradient để điều chỉnh bước học cho từng tham số, như sau:

xt = xt-1 - gt

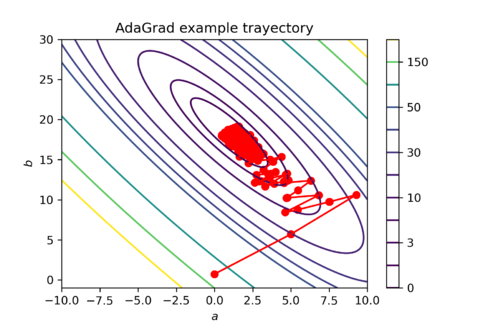
Trong đó:

xt là giá trị tham số tại thời điểm t

xt-1 là giá trị tham số tại thời điểm t-1

η là bước học (learning rate)

là một hằng số nhỏ để tránh chia cho 0



- Giống như các thuật toán tối ưu hóa khác, Adagrad cũng bắt đầu bằng việc tính toán gradient của hàm mất mát đối với các tham số mô hình. Gradient cho biết hướng mà hàm mất mát giảm xuống. Adagrad sử dụng một ước lượng độ lớn của gradient để điều chỉnh bước học.

#### Ưu điểm của Adagrad.

- Hiệu quả: Adagrad thường hội tụ nhanh hơn SGD, đặc biệt với các mô hình phức tạp và dữ liệu có độ phân tán gradient cao.

- Ổn định: Adagrad giúp giảm dao động trong quá trình huấn luyện, dẫn đến quá trình học ổn định hơn.

- Thích ứng: Adagrad tự động điều chỉnh bước học cho từng tham số, giúp cải thiện hiệu quả huấn luyện cho những tham số có độ phân tán gradient khác nhau.

#### Nhược điểm của Adagrad.

- Yêu cầu điều chỉnh tham số: Việc lựa chọn bước học và các tham số khác của Adagrad có thể ảnh hưởng đến hiệu quả của thuật toán.

- Bộ nhớ: Adagrad cần lưu trữ ước lượng độ lớn của gradient cho từng tham số, do đó yêu cầu bộ nhớ nhiều hơn so với SGD.

### RMSProp.

- RMSprop giải quyết vấn đề tỷ lệ học giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

- Cách thức hoạt động:

+ Tính toán gradient: Tại mỗi bước cập nhật, RMSprop tính toán gradient của hàm mất mát đối với các tham số mô hình.

+ Ước lượng trung bình bậc hai: RMSprop lưu trữ một ước lượng trung bình bậc hai của gradient đối với từng tham số.

Et[g2] = 0.9 \* Et-1[g^2] + 0.1 \* gt2

- Cập nhật tham số RMSprop sử dụng ước lượng trung bình bậc hai để điều chỉnh bước học cho từng tham số, như sau:

x\_t = xt-1 - gt\* + ε

Trong đó:

xt là giá trị tham số tại thời điểm t

xt-1 là giá trị tham số tại thời điểm t-1

η là bước học (learning rate)

ε là một hằng số nhỏ để tránh chia cho 0

#### Ưu điểm của RMSprop.

- Hiệu quả: RMSprop thường hội tụ nhanh hơn SGD và Momentum, đặc biệt với các mô hình phức tạp và dữ liệu có độ phân tán gradient cao.

- Ổn định: RMSprop giúp giảm dao động trong quá trình huấn luyện, dẫn đến quá trình học ổn định hơn.

- Thích ứng: RMSprop tự động điều chỉnh bước học cho từng tham số, giúp cải thiện hiệu quả huấn luyện cho những tham số có độ phân tán gradient khác nhau.

#### Nhược điểm của RMSprop.

- Yêu cầu điều chỉnh tham số: Việc lựa chọn bước học và các tham số khác của RMSprop có thể ảnh hưởng đến hiệu quả của thuật toán.

- Bộ nhớ: RMSprop cần lưu trữ ước lượng trung bình bậc hai cho từng tham số, do đó yêu cầu bộ nhớ nhiều hơn so với SGD.

### Adam.

- Thuật toán này kết hợp các kỹ thuật của Momentum và RMSProp để đạt được tốc độ hội tụ nhanh và độ chính xác cao.

- Thuật toán Adam sử dụng hai biến trạng thái để theo dõi hướng và cường độ của gradient:

+ Momentum: Đây là một biến trạng thái được cập nhật theo hướng của gradient, với tốc độ giảm dần theo thời gian.

+ RMSProp: Đây là một biến trạng thái được cập nhật theo cường độ của gradient, với trọng số giảm dần theo thời gian.

Sau mỗi lần cập nhật, các biến trạng thái này được sử dụng để tính toán một bước cập nhật cho các tham số của mô hình

- Một trong những thành phần chính của Adam là các trung bình động trọng số

mũ (hay còn được gọi là trung bình rò rỉ) để ước lượng cả động lượng và mô-men bậc

hai của gradient. Cụ thể, nó sử dụng các biến trạng thái

vt = β1vt-1 + (1 − β1)gt (1)

st = β2 st-1+(1−β2)gt2 (2)

Trong đó:

β1 và β2 là các tham số trọng số không âm thường được chọn các giá trị là β1 = 0.9 và β2 = 0.999 . Điều này có nghĩa là ước lượng phương sai di chuyển chậm hơn nhiều so với số hạng động lượng. Nếu ta khởi tạo s0 = v0 = 0, thuật toán sẽ có độ chệch ban đầu đáng kể về các giá trị nhỏ hơn. Vấn đề này có thể được giải quyết bằng cách sử dụng để chuẩn hóa lại các số hàng. Tương tự, các biến trạng thái được chuẩn hóa như sau:

và

Với các ước lượng thích hợp, bây giờ chúng ta có thể viết ra các phương trình cập nhật. Đầu tiên, chúng ta điều chỉnh lại giá trị gradient, tương tự như ở RmsProp để có được: gt‘= . Không giống như Rmsprop, phương trình cập nhật sử dụng động lượng thay vì gradient. Hơn nữa, có một số khác biệt nhỏ ở đây: phép chuyển đổi được thực hiện bằng cách sử dụng + ϵ thay vì. Trong thực tế, cách đầu tiên hoạt động tốt hơn một chút, dẫn đến sự khác biệt này so với Rmsprop. Thông thường, ta chọn ϵ = 10-6 để cân bằng giữa tính ổn định số học và độ tin cậy. Bây giờ ta sẽ tổng hợp lại tất cả các điều trên để tính toán bước cập nhật. xt = xt-1 − gt′ .

#### Ưu điểm của Adam.

- Tốc độ hội tụ nhanh.

- Độ chính xác cao.

- Dễ dàng triển khai.

- Yêu cầu bộ nhớ ít.

#### Nhược điểm của Adam.

- Có thể bị quá khớp trong một số trường hợp

- Có thể bị ảnh hưởng bởi các gradient có phương sai lớn.

### Adadelta

- AdaDelta là một kỹ thuật tối ưu hóa ngẫu nhiên cho phép áp dụng phương pháp tốc độ học theo chiều cho SGD. Nó là một phần mở rộng của Adagrad nhằm tìm cách giảm tốc độ học tập giảm dần và đơn điệu. Thay vì tích lũy tất cả các gradient bình phương trong quá khứ, Adadelta giới hạn cửa sổ của các gradient đã tích lũy trong quá khứ ở một kích thước cố định w.

- Thay vì lưu trữ không hiệu quả gradient bình phương trước đó, tổng các gradient được xác định đệ quy là trung bình giảm dần của tất cả các gradient bình phương trước đó. Trung bình chạy E[g2]t  tại bước thời gian sau đó chỉ phụ thuộc vào độ dốc trung bình và hiện tại trước đó:

E[g2]t = βE[g2]t -1 + (1- β)g2t

β thường được đặt bằng 0.9. Cập nhật SGD theo vectơ cập nhật tham số:

Δft = -n\*gt,i .

ft+1 = ft +Δft

AdaDelta có dạng:

Δft = - gt

#### Ưu điểm của AdaDelta.

- Tốc độ hội tụ nhanh: Adadelta có thể đạt tốc độ hội tụ nhanh hơn RMSProp, đặc biệt là trong các bài toán có hàm mất mát phức tạp.

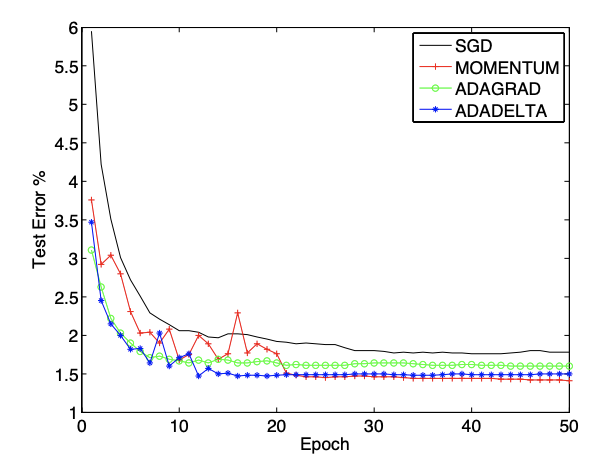
- Ít nhạy cảm với các gradient có phương sai lớn: Adadelta ít bị ảnh hưởng bởi các gradient có phương sai lớn, giúp hội tụ nhanh hơn.

- Dễ dàng triển khai: Adadelta có thể được triển khai một cách đơn giản và hiệu quả.

#### Nhược điểm của AdaDelta.

- Có thể bị quá khớp: Adadelta có thể bị quá khớp trong một số trường hợp, đặc biệt là khi sử dụng các tham số học tập lớn.

- Cần điều chỉnh tham số: Việc điều chỉnh các tham số của Adadelta, chẳng hạn như hệ số giảm dần, có thể cần thiết để đạt được kết quả tốt nhất.



# TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

## Continual Learning.

### Continual Learning là gì?

- Continual Learning (CL), còn được gọi là Học tập suốt đời hay Học tập gia tăng, là một lĩnh vực nghiên cứu trong trí tuệ nhân tạo (AI) tập trung vào khả năng của hệ thống trí tuệ học liên tục từ một chuỗi các nhiệm vụ, một sau một, mà không quên kiến thức đã học được từ các nhiệm vụ trước đó.

- Trong quy trình, mô hình tự cải thiện bằng cách học hỏi từ lần lặp mới nhất và cập nhật kiến thức của nó khi có dữ liệu mới. Vòng đời của mô hình học tập liên tục cho phép các mô hình duy trì tính phù hợp theo thời gian nhờ chất lượng năng động vốn có của chúng.

### Ưu điểm của việc học liên tục.

- Khả năng thích nghi: CL giúp các hệ thống AI có thể thích nghi với sự thay đổi của thế giới xung quanh. Ví dụ, một hệ thống phân tích hình ảnh y tế sử dụng CL có thể học cách phát hiện các loại bệnh mới mà không cần phải được huấn luyện lại hoàn toàn.

- Hiệu quả: CL có thể giúp các hệ thống AI học hiệu quả hơn bằng cách tận dụng kiến thức đã học được từ các nhiệm vụ trước đó. Ví dụ, một hệ thống dịch máy ngôn ngữ sử dụng CL có thể học cách dịch các ngôn ngữ mới nhanh hơn và hiệu quả hơn so với hệ thống chỉ được huấn luyện trên một bộ dữ liệu cố định.

- Linh hoạt: CL cho phép các hệ thống AI học các nhiệm vụ mới mà không cần phải thay đổi hoàn toàn kiến trúc của hệ thống. Điều này giúp các hệ thống AI trở nên linh hoạt hơn và có thể được sử dụng trong nhiều ứng dụng khác nhau.

- Giảm Giao Cắt (Interference): Continual Learning giúp giảm thiểu hiện tượng giao cắt, nơi kiến thức mới ảnh hưởng đến khả năng của mô hình trong việc nhớ và thực hiện các nhiệm vụ đã học trước đó.

### Các phương pháp học liên tục.

- Có 2 nhóm phương pháp học liên tục chủ yếu:

+ Các phương pháp tập trung vào việc giảm thiểu việc quên kiến thức cũ: Các phương pháp này cố gắng để ngăn chặn hiện tượng "quên thảm họa", hiện tượng hiệu suất giảm mạnh cho các nhiệm vụ cũ khi học các nhiệm vụ mới.

+ Các phương pháp tập trung vào việc tận dụng kiến thức đã học được từ các nhiệm vụ cũ: Các phương pháp này cố gắng để tận dụng kiến thức đã học được từ các nhiệm vụ cũ để học các nhiệm vụ mới một cách hiệu quả hơn.

- Một số phương pháp học liên tục phổ biến:

+ Học tập trọng số: Phương pháp này cập nhật các trọng số của mô hình theo cách mà các trọng số cho các nhiệm vụ cũ không bị thay đổi quá nhiều khi học các nhiệm vụ mới.

+ Học tập phân phối: Phương pháp này cập nhật phân phối của các trọng số của mô hình theo cách mà các trọng số cho các nhiệm vụ cũ vẫn có thể được truy cập.

+ Học tập tích hợp: Phương pháp này kết hợp các kiến thức đã học được từ các nhiệm vụ cũ với các kiến thức mới để học các nhiệm vụ mới một cách hiệu quả hơn.

+Học tập không gian: Phương pháp này cập nhật cấu trúc của mô hình theo cách mà các nhiệm vụ cũ vẫn có thể được thực hiện một cách hiệu quả.

+ Học tập dựa trên sự chú ý: Phương pháp này cho phép mô hình tập trung vào các thông tin quan trọng từ các nhiệm vụ mới, từ đó ngăn chặn việc quên kiến thức cũ.

+ Học tập dựa trên bản sao: Phương pháp này tạo ra một bản sao của mô hình cho mỗi nhiệm vụ, từ đó giúp mô hình học các nhiệm vụ mới mà không ảnh hưởng đến hiệu suất của các nhiệm vụ cũ.

### Các giai đoạn của continuous learning.

- Giai đoạn 1: Huấn luyện lại một cách thủ công. Trong giai đoạn này, các mô hình chỉ được huấn luyện lại khi đáp ứng hai điều kiện:

+ Hiệu suất của mô hình đã suy giảm đến mức hiện tại nó gây hại nhiều hơn là có lợi.

+ Nhóm của bạn có thời gian để cập nhật mô hình.

- Giai đoạn 2: Huấn luyện lại trạng thái tự động theo lịch trình không cố định Trong giai đoạn này, các mô hình được huấn luyện lại theo lịch trình không cố định, thường dựa trên "trực giác". Điểm uốn giữa giai đoạn 1 và giai đoạn 2 thường là một tập lệnh do ai đó viết để chạy quá trình huấn luyện lại không trạng thái theo định kỳ. Việc viết tập lệnh này có thể rất dễ hoặc rất khó tùy thuộc vào số lượng phần phụ thuộc cần được phối hợp để huấn luyện lại một mô hình. Các bước cấp cao của tập lệnh này là:

+ Kéo dữ liệu

+ Xuống mẫu hoặc lấy mẫu dữ liệu nếu cần thiết

+ Trích xuất các tính năng

+ Xử lý và/hoặc chú thích nhãn để tạo dữ liệu huấn luyện

+ Bắt đầu quá trình huấn luyện

+ Đánh giá mô hình mới

+ Triển khai sử dụng

- Giai đoạn 3: Trong giai đoạn này, các mô hình được huấn luyện lại theo lịch trình định kỳ, thường là theo tuần, tháng, hoặc quý. Để đạt được điều này, cần thiết phải thiết lập một hệ thống tự động theo dõi hiệu suất của mô hình và kích hoạt quá trình huấn luyện lại khi cần thiết. Một ví dụ về cách theo dõi hiệu suất của mô hình là sử dụng chỉ số đánh giá (metric) như độ chính xác, độ chính xác, hoặc thời gian đáp ứng. Khi chỉ số đánh giá giảm xuống dưới một ngưỡng nhất định, hệ thống sẽ kích hoạt quá trình huấn luyện lại. Trong ví dụ trên, V1 và V2 là hai phiên bản của cùng một mô hình, với V1 là phiên bản cũ hơn và V2 là phiên bản mới hơn. Số phiên bản được ghi theo sau bởi một dấu chấm, chẳng hạn như V1.2 và V2.3. Số sau dấu chấm cho biết số lần mô hình đã được huấn luyện lại.

- Giai đoạn 4: Huấn luyện lại linh hoạt Trong giai đoạn này, các mô hình được huấn luyện lại một cách linh hoạt, không theo lịch trình cố định. Thay vào đó, các mô hình được huấn luyện lại khi cần thiết, dựa trên một số cơ chế kích hoạt. Các cơ chế kích hoạt phổ biến bao gồm:

+ Kích hoạt dựa trên thời gian: Các mô hình được huấn luyện lại sau một khoảng thời gian nhất định, chẳng hạn như mỗi tuần, mỗi tháng, hoặc mỗi quý.

+ Kích hoạt dựa trên hiệu suất: Các mô hình được huấn luyện lại khi hiệu suất của chúng suy giảm xuống dưới một ngưỡng nhất định.

+ Kích hoạt dựa trên khối lượng: Các mô hình được huấn luyện lại khi có thêm dữ liệu mới được thu thập.

+ Kích hoạt dựa trên sự trôi dạt: Các mô hình được huấn luyện lại khi có sự thay đổi trong phân phối dữ liệu của tập huấn luyện.

+ Kích hoạt dựa trên sự trôi dạt có thể là một cách hiệu quả để ngăn chặn mô hình suy giảm hiệu suất khi phân phối dữ liệu thay đổi. Tuy nhiên, cũng có thể khó xác định khi nào sự thay đổi phân phối dữ liệu là đáng kể và cần phải huấn luyện lại mô hình.

### Các thách thức của học liên tục.

- Tính tốn kém: Huấn luyện lại hoàn toàn hoặc huấn luyện theo dòng đòi hỏi nhiều khả năng tính toán hơn so với huấn luyện lại một phần. Điều này có thể là một thách thức đối với các hệ thống có khả năng tính toán hạn chế, chẳng hạn như thiết bị di động hoặc thiết bị Internet vạn vật (IoT).

- Tính phức tạp: Học liên tục có thể phức tạp hơn so với học máy truyền thống, đặc biệt là đối với các hệ thống phân phối và thực hiện. Điều này có thể là một thách thức đối với các nhà phát triển và kỹ sư hệ thống, những người cần hiểu rõ về các nguyên tắc học máy và các hệ thống phân tán.

- Tính tin cậy: Học liên tục có thể làm giảm độ tin cậy của mô hình, đặc biệt là khi dữ liệu mới không đại diện chính xác cho phân phối dữ liệu thực tế. Điều này có thể là một thách thức đối với các ứng dụng quan trọng, chẳng hạn như chẩn đoán bệnh hoặc lái xe tự động.

- Việc áp dụng việc học tập liên tục như một phương pháp thực hành có nguy cơ dẫn đến những thất bại của mô hình. Việc cập nhật mô hình càng thường xuyên thì càng có nhiều cơ hội để mô hình thất bại.

### Các ứng dụng của việc học liên tục.

- Phân loại ngôn ngữ: Học liên tục có thể được sử dụng để cải thiện hiệu quả của các mô hình phân loại ngôn ngữ khi có thêm dữ liệu văn bản mới được tạo ra. Ví dụ, một mô hình phân loại ngôn ngữ có thể được sử dụng để phân loại các tin nhắn văn bản thành các loại khác nhau, chẳng hạn như tin nhắn spam, tin nhắn rác, hoặc tin nhắn quan trọng. Học liên tục có thể giúp mô hình học cách phân loại các tin nhắn mới hơn một cách chính xác hơn.

- Chẩn đoán bệnh: Học liên tục có thể được sử dụng để cải thiện độ chính xác của các mô hình chẩn đoán bệnh khi có thêm dữ liệu y tế mới được thu thập. Ví dụ, một mô hình chẩn đoán bệnh có thể được sử dụng để chẩn đoán các bệnh dựa trên các triệu chứng của bệnh nhân. Học liên tục có thể giúp mô hình học cách chẩn đoán các bệnh mới hơn một cách chính xác hơn.

- Bảo mật: Học liên tục có thể được sử dụng để cải thiện hiệu quả của các hệ thống bảo mật khi có thêm dữ liệu về các cuộc tấn công mạng mới.

- Công nghệ sản xuất: Học liên tục có thể được sử dụng để cải thiện hiệu quả của các quy trình sản xuất khi có thêm dữ liệu về các sản phẩm hoặc dịch vụ mới.

- Lái xe tự động: Học liên tục có thể được sử dụng để cải thiện hiệu quả của các hệ thống lái xe tự động khi có thêm dữ liệu từ môi trường xung quanh. Ví dụ, một hệ thống lái xe tự động có thể sử dụng học liên tục để học cách tránh các chướng ngại vật hoặc cách vượt qua các tình huống giao thông phức tạp.

## Test Production

- Testing Production là quá trình kiểm thử và đảm bảo rằng mô hình máy học và hệ thống xử lý dữ liệu của bạn hoạt động hiệu quả khi triển khai vào môi trường sản xuất. Điều này bao gồm việc kiểm tra cả mô hình máy học và toàn bộ quy trình xử lý dữ liệu từ đầu đến cuối để đảm bảo rằng chúng đáp ứng đúng yêu cầu và không gặp vấn đề không mong muốn khi sử dụng trong môi trường thực tế.

- Để đảm bảo chất lượng của các mô hình học máy, mỗi nhóm cần có một quy trình đánh giá rõ ràng, bao gồm các thử nghiệm cần thực hiện, người thực hiện và các ngưỡng cần đạt được để mô hình được triển khai. Tốt nhất, các quy trình này nên được tự động hóa để có thể chạy liên tục và kịp thời phát hiện các vấn đề tiềm ẩn.

### Các phương pháp kiểm tra.

- Kiểm thử tính năng (Functional Testing): Đảm bảo mô hình thực hiện chính xác các chức năng dự kiến cho các đầu vào khác nhau. Kiểm tra xem mô hình có đưa ra dự đoán hoặc đánh giá chính xác hay không. Có thể sử dụng các kỹ thuật kiểm thử phần mềm truyền thống như kiểm thử hộp đen và hộp trắng.

- Kiểm thử hiệu suất (Performance Testing): Đánh giá tốc độ, độ trễ và khả năng mở rộng của mô hình dưới tải trọng thực. Đo các chỉ số như thời gian phản hồi, throughput và tài nguyên sử dụng. Xác định xem mô hình có thể xử lý khối lượng giao dịch và dữ liệu dự kiến trong môi trường sản xuất hay không.

- Kiểm thử độ trôi dạt (Drift Testing): Theo dõi hiệu suất của mô hình theo thời gian để phát hiện sự suy giảm do dữ liệu thay đổi hoặc các yếu tố khác. Sử dụng các kỹ thuật như tính toán các chỉ số thống kê về đầu ra của mô hình hoặc so sánh dự đoán của mô hình với các giá trị thực tế. Ngăn chặn mô hình đưa ra kết quả sai lệch do dữ liệu hoặc môi trường thay đổi.

- Kiểm thử chéo (Cross-validation Testing): Kiểm tra mô hình trên các hệ thống và môi trường khác nhau để đảm bảo tính tương thích và khả năng tổng quát. Xác định xem mô hình có thể hoạt động hiệu quả trên các cấu hình phần cứng và phần mềm khác nhau hay không. Phòng tránh các vấn đề về triển khai do sự khác biệt giữa môi trường phát triển và sản xuất.

- Kiểm thử adversarial (Adversarial Testing): Kiểm tra mô hình với dữ liệu được thiết kế để đánh lừa hoặc tấn công nó, nhằm tăng cường khả năng chống chịu lỗi. Sử dụng các kỹ thuật như tấn công lừa đảo (adversarial attacks) để tìm lỗ hổng trong mô hình. Cải thiện khả năng của mô hình trong việc xử lý dữ liệu nhiễu và các trường hợp cạnh.

- Theo dõi mô hình: Sử dụng các bảng điều khiển và cảnh báo để theo dõi hiệu suất của mô hình và phát hiện các vấn đề tiềm ẩn.

- Kiểm thử liên tục: Tích hợp kiểm thử vào quy trình CI/CD để tự động kiểm tra mô hình khi có thay đổi mới.

### Các loại kiểm tra.

#### Kiểm thử A/B:

So sánh phiên bản mới của mô hình với phiên bản hiện tại bằng cách phân tách lưu lượng truy cập.

* Ưu điểm:

+ Đơn giản và dễ thực hiện: Kiểm thử A/B chỉ yêu cầu phân tách lưu lượng truy cập của người dùng thành hai nhóm, sau đó phân phối ngẫu nhiên mỗi nhóm đến một phiên bản của sản phẩm hoặc dịch vụ.

+ Tính khách quan: Kiểm thử A/B cung cấp kết quả chính xác và khách quan, dựa trên dữ liệu thực tế từ người dùng.

+ Tính hiệu quả: Kiểm thử A/B có thể được sử dụng để đánh giá hiệu suất của các mô hình ML trên quy mô lớn

* Nhược điểm:

+ Yêu cầu đủ dữ liệu: Kiểm thử A/B cần đủ dữ liệu để có thể thu được kết quả đáng tin cậy.

+ Yêu cầu thời gian: Kiểm thử A/B có thể mất thời gian để thu thập đủ dữ liệu, đặc biệt là đối với các sản phẩm hoặc dịch vụ có lượng người dùng lớn.

+ Yêu cầu phân tích dữ liệu: Kiểm thử A/B yêu cầu có kỹ năng phân tích dữ liệu để có thể đánh giá kết quả thử nghiệm một cách chính xác.

#### Kiểm thử canary.

Triển khai phiên bản mới của mô hình cho một nhóm nhỏ người dùng trước khi triển khai rộng rãi.

* Ưu điểm của kiểm thử canary:

+ Giảm thiểu rủi ro: Kiểm thử canary giúp giảm thiểu rủi ro khi triển khai phiên bản mới của một sản phẩm hoặc dịch vụ, bằng cách phát hiện sớm các vấn đề tiềm ẩn trước khi triển khai rộng rãi.

+ Tăng cường trải nghiệm người dùng: Kiểm thử canary giúp cải thiện trải nghiệm người dùng, bằng cách phát hiện và khắc phục các vấn đề trước khi ảnh hưởng đến nhiều người dùng.

+ Tiết kiệm thời gian và chi phí: Kiểm thử canary có thể giúp tiết kiệm thời gian và chi phí, bằng cách giảm thiểu nhu cầu triển khai lại phiên bản mới của sản phẩm hoặc dịch vụ.

* Nhược điểm của kiểm thử canary

+ Yêu cầu kỹ năng và kinh nghiệm: Kiểm thử canary yêu cầu có kỹ năng và kinh nghiệm để có thể triển khai và quản lý một cách hiệu quả.

+ Có thể gây ra sự gián đoạn: Kiểm thử canary có thể gây ra sự gián đoạn cho một số người dùng, nếu phiên bản mới của sản phẩm hoặc dịch vụ gặp phải vấn đề.

# **TÀI LIỆU THAM KHẢO**