TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PHẠM NGUYỄN MINH QUANG - 52100926**

**TRÌNH BÀY NGHIÊN CỨU VỀ CÁC PHƯƠNG PHÁP OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION**

**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**



**PHẠM NGUYỄN MINH QUANG - 52100926**

**TRÌNH BÀY NGHIÊN CỨU VỀ CÁC PHƯƠNG PHÁP OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION**

**BÁO CÁO CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

Người hướng dẫn

**TS. Lê Anh Cường**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Chúng em xin gửi lời biết ơn sâu sắc đến các thầy cô trong khoa Công Nghệ Thông Tin của trường Đại học Tôn Đức Thắng; đặc biệt là thầy Lê Anh Cường, người đã hướng dẫn, hỗ trợ và tạo mọi điều kiện thuận lợi cho chúng em có thể tiếp cận đến những nguồn kiến thức, tài liệu cũng như là kinh nghiệm thực tiễn về cách áp dụng những kiến thức đấy vào một dự án nhỏ trong thực tế.

Đây là lần đầu tiên nhóm em tham gia vào một dự án có sử dụng những kiến thức mới, còn nhiều bỡ ngỡ nên chắc chắn sẽ có nhiều sai sót và khó khăn. Do đó chúng em rất mong nhận được những góp ý quý báu của thầy để có thể hoàn thiện hơn sự hiểu biết về lĩnh vực này. Đồng thời, có thể bổ sung, nâng cao kiến thức của mình, cũng như nâng cao chất lượng các bài báo cáo trong các học phần và môn học tiếp theo.

Chúng em xin chân thành cảm ơn thầy!

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

*Tác giả*

A close-up of a signature

Description automatically generated*(Ký tên và ghi rõ họ tên)*

Phạm Nguyễn Minh Quang

**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 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 23 tháng 12 năm 2023*

*Tác giả*

A close-up of a signature

Description automatically generated*(Ký tên và ghi rõ họ tên)*

Phạm Nguyễn Minh Quang

**TRÌNH BÀY NGHIÊN CỨU VỀ CÁC PHƯƠNG PHÁP OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION**

**TÓM TẮT**

Nghiên cứu này tập trung vào việc hiểu và so sánh các phương pháp tối ưu hóa phổ biến trong quá trình huấn luyện mô hình học máy. Các phương pháp phổ biến như Gradient Descent, Stochastic Gradient Descent, Momentum, Adagrad, RMSprop, Adam Optimizer và được xem xét chi tiết về hiệu suất, tốc độ học, ổn định và khả năng quản lý nhiễu.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vi](#_Toc154252192)

[DANH MỤC BẢNG BIỂU vii](#_Toc154252193)

[DANH MỤC CÁC CHỮ VIẾT TẮT viii](#_Toc154252194)

[CHƯƠNG 1. TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY 1](#_Toc154252195)

[1.1 Gradient Descent (GD) 1](#_Toc154252196)

[1.1.1 Hiệu suất 2](#_Toc154252197)

[1.1.2 Tốc độ học 2](#_Toc154252198)

[1.1.3 Ổn định 2](#_Toc154252199)

[1.1.4 Quản lý nhiễu 2](#_Toc154252200)

[1.2 Stochastic Gradient Descent (SGD) 2](#_Toc154252201)

[1.2.1 Hiệu suất 3](#_Toc154252202)

[1.2.2 Tốc độ học 3](#_Toc154252203)

[1.2.3 Ổn định 3](#_Toc154252204)

[1.2.4 Quản lý nhiễu 3](#_Toc154252205)

[1.3 Momentum 4](#_Toc154252206)

[1.3.1 Hiệu suất 5](#_Toc154252207)

[1.3.2 Tốc độ học 5](#_Toc154252208)

[1.3.3 Ổn định 5](#_Toc154252209)

[1.3.4 Quản lý nhiễu 5](#_Toc154252210)

[1.4 Adagrad 5](#_Toc154252211)

[1.4.1 Hiệu suất 6](#_Toc154252212)

[1.4.2 Tốc độ học 6](#_Toc154252213)

[1.4.3 Ổn định 7](#_Toc154252214)

[1.4.4 Quản lý nhiễu 7](#_Toc154252215)

[1.5 RMSprop 7](#_Toc154252216)

[1.5.1 Hiệu suất 8](#_Toc154252217)

[1.5.2 Tốc độ học 8](#_Toc154252218)

[1.5.3 Ổn định 8](#_Toc154252219)

[1.5.4 Quản lý nhiễu 8](#_Toc154252220)

[1.6 Adam 9](#_Toc154252221)

[1.6.1 Hiệu suất 10](#_Toc154252222)

[1.6.2 Tốc độ học 10](#_Toc154252223)

[1.6.3 Ổn định 10](#_Toc154252224)

[1.6.4 Quản lý nhiễu 10](#_Toc154252225)

[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 11](#_Toc154252226)

[2.1 Continual Learning 11](#_Toc154252227)

[2.2 Test Production 12](#_Toc154252228)

[2.3 Mối liên hệ giữa Continual Learning và Test Production 12](#_Toc154252229)

[2.4 TÀI LIỆU THAM KHẢO 14](#_Toc154252230)

# DANH MỤC HÌNH VẼ

**No table of figures entries found.**

# DANH MỤC BẢNG BIỂU

**No table of figures entries found.**

# DANH MỤC CÁC CHỮ VIẾT TẮT

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

Trong huấn luyện mô hình học máy, “optimizer” là một thuật toán tối ưu hóa quan trọng. Optimizer giúp tìm kiếm các giá trị trọng số phù hợp cho mô hình, từ đó tối ưu hóa hiệu suất của mô hình. Về cơ bản, thuật toán tối ưu là cơ sở để xây dựng mô hình neural network với mục đích “học” được các features (hay pattern) của dữ liệu đầu vào. Optimizer giúp cải thiện weight và bias theo từng bước, và đó là lý do các thuật toán optimizer ra đời.

## Gradient Descent (GD)

Gradient Descent là một thuật toán tối ưu lặp (iterative optimization algorithm) được sử dụng trong các bài toán Machine Learning và Deep Learning. Mục tiêu của thuật toán này là tìm một tập các biến nội tại (internal parameters) để tối ưu hóa mô hình.

Trong Gradient Descent, chúng ta thường tìm giá trị nhỏ nhất của một hàm số. Tuy nhiên, đối với các hàm số nhiều biến, việc tìm đạo hàm có thể rất phức tạp, thậm chí là bất khả thi. Do đó, người ta thường tìm điểm gần với điểm cực tiểu nhất và xem đó là nghiệm của bài toán.

Công thức cập nhật trong Gradient Descent là:

Trong đó, là gradient (đạo hàm) của hàm mục tiêu tại .

Gradient Descent phụ thuộc vào nhiều yếu tố:

Điểm khởi tạo: Nếu chọn điểm khởi tạo khác nhau, kết quả cuối cùng của Gradient Descent cũng sẽ khác nhau.

Tốc độ học (learning rate): Nếu tốc độ học quá nhỏ, tốc độ hội tụ sẽ rất chậm, ảnh hưởng đến quá trình huấn luyện. Ngược lại, nếu tốc độ học quá lớn, thuật toán có thể không hội tụ.

### Hiệu suất

Hiệu suất của Gradient Descent phụ thuộc vào nhiều yếu tố như điểm khởi tạo và tốc độ học. Nếu chọn điểm x ban đầu khác nhau sẽ ảnh hưởng đến quá trình hội tụ.

### Tốc độ học

Tốc độ học (learning rate) là một tham số quan trọng trong Gradient Descent. Nếu tốc độ học quá nhỏ thì tốc độ hội tụ rất chậm ảnh hưởng đến quá trình training, còn tốc độ học quá lớn thì tiến nhanh tới đích sau vài vòng lặp tuy nhiên thuật toán không hội tụ, quanh quẩn quanh đích vì bước nhảy quá lớn.

### Ổn định

Gradient Descent có thể không ổn định nếu bề mặt có nhiều đáy. Tuy nhiên, có các biến thể của Gradient Descent như Momentum, Nesterov accelerated gradient (NAG), … giúp khắc phục những hạn chế này.

### Quản lý nhiễu

Gradient Descent có thể bị ảnh hưởng bởi nhiễu. Điều này đặc biệt quan trọng khi làm việc với dữ liệu thực tế, có thể chứa nhiều nhiễu. Các biến thể của Gradient Descent như Stochastic Gradient Descent và Mini-batch Gradient Descent có thể giúp quản lý nhiễu.

## Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của Gradient Descent (GD), chúng ta sử dụng toàn bộ tập dữ liệu để tính toán gradient cho mỗi lần lặp. Nhưng thay vì cập nhật trọng số sau mỗi lô dữ liệu (batch), SGD cập nhật trọng số sau mỗi mẫu dữ liệu. Đầu tiên SGD **chọn ngẫu nhiên** một mẫu dữ liệu từ tập huấn luyện sau đó **tính độ dốc** (gradient) của hàm mất mát (loss function) đối với mẫu dữ liệu đó và sẽ **cập nhật trọng số** của mô hình dựa trên độ dốc tính được. Cuối cùng **lặp lại quá trình** trên cho tất cả các mẫu dữ liệu trong tập huấn luyện.

Công thức cập nhật trọng số của SGD:

Trong đó:

* là vector trọng số
* là learning rate
* là độ dốc của loss function theo trọng số

Công thức sử dụng mini-batch:

Trong đó là một mẫu dữ liệu trong mini-batch, và là độ dốc của loss function theo trọng số trên mẫu đó.

### Hiệu suất

SGD thường hoạt động tốt trên các tập dữ liệu lớn với nhiều mẫu và có thể giúp tránh việc rơi vào các điểm tối thiểu cục bộ. Nhưng SGD có thể tạo ra sự dao động trong quá trình học, đặc biệt là khi kích thước bước quá lớn hoặc quá nhỏ.

### Tốc độ học

Tốc độ học có thể được điều chỉnh một cách linh hoạt. Mỗi lần cập nhật chỉ sử dụng một mẫu, giúp mô hình có thể học nhanh chóng từ dữ liệu mới. Cần sự chăm sóc trong việc chọn kích thước bước để tránh sự dao động hoặc hội tụ quá chậm.

### Ổn định

SGD có khả năng tránh các điểm tối thiểu cục bộ nhờ tính ngẫu nhiên trong quá trình cập nhật trọng số. Có thể gặp sự dao động đặc biệt khi dữ liệu có nhiễu.

### Quản lý nhiễu

Sự ngẫu nhiên trong việc chọn mẫu giúp giảm ảnh hưởng của nhiễu trong gradient. Đôi khi nó có thể tạo ra bước cập nhật nhiễu, đặc biệt là nếu kích thước bước không được điều chỉnh đúng cách. Để cải thiện hiệu suất và ổn định ta có thể sử dụng một số biến thể hoặc kỹ thuật điều chỉnh như: Mini-batch SGD, Learning rate decay, Momentum, Nesterov Accelerated Gradient (NAG), …

## Momentum

Momentum optimizer là một biến thể của thuật toán tối ưu Gradient Descent, được sử dụng để mô tả khả năng của một quá trình tối ưu hóa theo dõi và giữ lại Momentum của quá trình tối ưu hóa trước đó. Mômentum giúp tăng tốc quá trình hội tụ của thuật toán tối ưu hóa và giảm độ dao động trong quá trình tìm kiếm trọng số tối ưu.

Một vấn đề của thuật toán gradient descent là quá trình tìm kiếm có thể dao động xung quanh không gian tìm kiếm dựa trên gradient. Ví dụ, tìm kiếm có thể tiến triển xuống phía dưới đến điểm tối thiểu, nhưng trong quá trình này, nó có thể di chuyển theo hướng khác, thậm chí đi lên, tùy thuộc vào gradient của các điểm cụ thể (bộ tham số) gặp phải trong quá trình tìm kiếm. Điều này có thể làm chậm tiến trình của tìm kiếm, đặc biệt là đối với những vấn đề tối ưu hóa nơi xu hướng tổng thể hoặc hình dạng của không gian tìm kiếm quan trọng hơn so với các gradient cụ thể trên đường đi. Điều này có thể làm chậm tiến trình của tìm kiếm, đặc biệt là đối với những vấn đề tối ưu hóa nơi xu hướng tổng thể hoặc hình dạng của không gian tìm kiếm quan trọng hơn so với các gradient cụ thể trên đường đi.

Dưới góc nhìn vật lý, SGD với momentum giúp cho việc hội tụ có gia tốc, làm nhanh quá trình hội tụ trên các đường cong có độ dốc lớn, nhưng cũng đồng thời làm giảm sự dao dộng khi gần hội tụ.

Ở đây:

* là giá trị momentum
* là hệ số ma sát, giả sử nó bằng 0.9
* là độ dốc của hàm mục tiêu tại vòng lặp
* là các tham số
* là tốc độ học

Ví dụ, một momentum lớn (0.9) có nghĩa là cập nhật bị ảnh hưởng mạnh mẽ bởi cập nhật trước đó, trong khi một momentum bé (0.2) có nghĩa là ảnh hưởng rất ít.

### Hiệu suất

Momentum giúp tăng tốc quá trình hội tụ và đạt được kết quả tốt hơn bằng cách giữ lại thông tin về động lượng của quá trình tối ưu hóa trước đó hoặc khi hàm mục tiêu có độ cong cao, gradient nhỏ nhưng nhất quán. Trong một số trường hợp, nếu learning rate quá lớn, mô hình có thể vượt qua điểm tối ưu hoặc gặp phải vấn đề của bước cập nhật quá nhanh.

### Tốc độ học

Momentum có thể sử dụng một tốc độ học lớn hơn mà không bị phân kỳ, do đó giảm thời gian học, giúp giảm ảnh hưởng của các bước cập nhật nhanh và tăng tốc hội tụ. Cần thiết lập kích thước bước cẩn thận để tránh vấn đề của bước cập nhật quá nhanh.

### Ổn định

Momentum giúp ổn định quá trình tối ưu hóa, giảm độ dao động và tránh rơi vào các điểm tối thiểu cục bộ. Cần lựa chọn kích thước bước và hệ số mômentum thích hợp để đảm bảo sự ổn định.

### Quản lý nhiễu

Momentum có thể giúp giảm ảnh hưởng của nhiễu trong gradient, do đó giảm thiểu sự biến động của các cập nhật tham số, đặc biệt là khi sử dụng mini-batch.

## Adagrad

Gradient descent (GD) có một hạn chế đó là là nó sử dụng cùng một learning rate cho mỗi biến đầu vào. Điều này có thể là một vấn đề trên các hàm mục tiêu có độ cong khác nhau theo các chiều khác nhau, và do đó, có thể yêu cầu một kích thước bước khác nhau để đến điểm mới.

**Adaptive Gradients**, hay gọi tắt là **AdaGrad**, là biến thể của thuật toán tối ưu hóa gradient descent cho phép step size (learning rate) trong mỗi chiều sử dụng bởi thuật toán tối ưu hóa tự động thích ứng dựa trên độ dốc được quan sát cho biến (đạo hàm riêng) thấy qua quá trình tìm kiếm.

Sử dụng AdaGrad nhằm cải thiện quá trình tối ưu hóa bằng cách tự động điều chỉnh kích thước bước cho mỗi biến trong không gian tìm kiếm. Khác với phương pháp gradient descent truyền thống, AdaGrad giải quyết vấn đề của step size đồng đều cho tất cả các biến, cho phép di chuyển lớn hơn trong các chiều có độ dốc cao và di chuyển nhỏ hơn trong các chiều có độ dốc thấp hơn. Thuật toán điều chỉnh tốc độ học cho mỗi biến dựa trên đạo hàm riêng, giúp tối ưu hóa hiệu quả trong các chiều đa dạng. AdaGrad đặc biệt hữu ích trong các hàm mục tiêu với độ cong thay đổi qua các chiều. Nó yêu cầu một step size ban đầu, nhưng ưu điểm của nó là ít nhạy cảm hơn đối với tốc độ học ban đầu so với phương pháp gradient descent truyền thống.

Ở đây, θ là tham số được cập nhật sau mỗi vòng lặp, η là tốc độ học, G là tổng bình phương của các độ dốc trước đó cho tham số đó, và g là độ dốc hiện tại.

### Hiệu suất

AdaGrad giúp cải thiện hiệu suất của thuật toán tối ưu hóa bằng cách điều chỉnh tự động kích thước bước cho mỗi biến đầu vào. Điều này giúp thuật toán chủ động hơn trong việc di chuyển qua các chiều có độ dốc lớn hơn và giảm di chuyển trong các chiều có độ dốc thấp hơn, tối ưu hóa quá trình hội tụ.

### Tốc độ học

Cung cấp một cách tự động để điều chỉnh tốc độ học của mỗi biến, giúp thuật toán tự động thích ứng với độ khó của bài toán tối ưu hóa. Điều này làm cho tốc độ học trở nên linh hoạt hơn và giúp ngăn chặn vấn đề của tốc độ học cố định trong các thuật toán truyền thống.

### Ổn định

Tăng tính ổn định của quá trình tối ưu hóa bằng cách giảm ảnh hưởng của các biến có độ dốc cao và tăng ảnh hưởng của các biến có độ dốc thấp. Điều này giúp kiểm soát sự dao động của quá trình tối ưu hóa và giảm khả năng bị ảnh hưởng mạnh bởi outliers.

### Quản lý nhiễu

Tự động thích ứng với độ dốc của mỗi biến, giúp giảm ảnh hưởng của nhiễu trong quá trình tối ưu hóa. Bằng cách giảm độ dốc của các biến có độ dốc lớn, AdaGrad có thể giúp tránh tình trạng quá tương ứng với nhiễu và cải thiện khả năng tổng quát hóa của mô hình.

## RMSprop

Như đã nói ở trên, AdaGrad cho phép tự động điều chỉnh kích thước bước cho mỗi biến dựa trên đạo hàm riêng (partial derivatives) quan sát được trong quá trình tối ưu hóa. Tuy nhiên, một hạn chế của AdaGrad là nó có thể dẫn đến kích thước bước rất nhỏ cho mỗi tham số vào cuối quá trình tìm kiếm, có thể làm chậm quá trình tối ưu hóa và gây khó khăn trong việc tìm ra giá trị tối ưu.

Root Mean Squared Propagation, hay còn được gọi tắt là RMSProp là một mở rộng của thuật toán gradient descent và còn là một phiên bản AdaGrad của gradient descent sử dụng trung bình trượt giảm dần của đạo hàm riêng trong việc điều chỉnh kích thước bước cho mỗi tham số. Việc sử dụng trung bình trượt giảm dần cho phép thuật toán quên đạo hàm sớm và tập trung vào đạo hàm riêng gần đây nhất được quan sát trong quá trình tìm kiếm, vượt qua hạn chế của AdaGrad.

RMSProp duy trì trung bình di động của các độ dốc bình phương cho mỗi trọng số. Sau đó, chúng ta chia độ dốc cho căn bậc hai của trung bình bình phương.

Ở đây là trung bình động của bình phương các độ dốc, là độ dốc của hàm chi phí liên quan đến trọng số, là tốc độ học, và là tham số trung bình động (giá trị mặc định là 0.9, để làm cho tổng giá trị độ dốc mặc định là 0.1 ở mini-batches thứ chín và -0.9 ở mini-batches thứ mười gần như bằng không, và giá trị mặc định là 0.001).

### Hiệu suất

RMSProp thường hoạt động hiệu quả trên nhiều loại bài toán và có khả năng chống lại các vấn đề của các phương pháp tối ưu hóa khác như gradient descent thông thường. Thuật toán giúp giảm tốc độ học ở những hướng có gradient lớn và tăng tốc độ học ở những hướng có gradient nhỏ.

### Tốc độ học

Tự động điều chỉnh tốc độ học cho từng tham số riêng lẻ của mô hình. Điều này giúp làm giảm nguy cơ quá mức thay đổi (exploding) hoặc biến mất (vanishing) gradient. Các giá trị của tốc độ học được lưu trữ theo dạng bình phương của gradient trung bình, giúp giảm độ nhạy của thuật toán với tốc độ học.

### Ổn định

Tăng tính ổn định của quá trình đào tạo bằng cách giảm tốc độ học theo thời gian, đặc biệt là khi tiếp cận điểm cực tiểu. Việc sử dụng bình phương của gradient trung bình giúp giảm độ biến động của tốc độ học, làm cho quá trình đào tạo mô hình trở nên ổn định hơn.

### Quản lý nhiễu

Quản lý nhiễu bằng cách lấy bình phương của gradient trung bình, giúp giảm nhiễu do sự biến động mạnh của gradient. Thuật toán có khả năng thích ứng với sự biến động của gradient, giúp làm giảm nhiễu đặc biệt hiệu quả trên dữ liệu có độ biến động lớn.

## Adam

Adam là thuật toán tối ưu hóa được sử dụng rộng rãi trong học máy, đặc biệt là trong huấn luyện mạng nơron. Adam kết hợp các ưu điểm của Adagrad và RMSprop, giúp cho quá trình huấn luyện nhanh hơn, ổn định hơn và có kết quả tốt hơn.

Adam hoạt động bằng cách cập nhật các tham số của mô hình theo hướng giảm hàm mất mát. Hàm mất mát là một hàm đo lường mức độ chính xác của mô hình trên tập dữ liệu huấn luyện.

Trong đó:

* ​ là moment bậc nhất (trung bình động của gradient) tại thời điểm t.
* là moment bậc hai (trung bình động của bình phương gradient) tại thời điểm t.
* ​ và ​ là các ước lượng hiệu chỉnh của ​ ​ và để tránh trường hợp chúng bị thiên về 0 khi t nhỏ.
* ​ là gradient của hàm mục tiêu tại thời điểm t.
* là tham số cần tối ưu hóa tại thời điểm t.
* α là tốc độ học.
* β1​ và β2​ là các hệ số quyết định mức độ giảm của moment bậc nhất và bậc hai. Thông thường, β1 ​= 0.9 và β2​ = 0.999.
* ϵ là một hằng số nhỏ để tránh chia cho 0. Thông thường, ϵ = 10−8.

### Hiệu suất

Adam thường cho hiệu suất tốt trên nhiều loại bài toán khác nhau và được sử dụng phổ biến trong cộng đồng học máy. Thuật toán này thích ứng tốt với độ chệch của hàm mục tiêu, giúp tăng tốc quá trình học so với các phương pháp tối ưu hóa khác.

### Tốc độ học

Adam tự động điều chỉnh tốc độ học cho từng tham số riêng lẻ của mô hình. Cơ chế thích ứng này giúp giảm độ nhạy của thuật toán với tốc độ học, giảm nguy cơ quá mức thay đổi (exploding) hoặc biến mất (vanishing) gradient.

### Ổn định

Adam thường khá ổn định và thích ứng tốt với các hình dạng khác nhau của hàm mục tiêu. Sự kết hợp của hai moment (momentum và bình phương động) giúp giảm độ dao động của tốc độ học và làm tăng ổn định của quá trình đào tạo.

### Quản lý nhiễu

Adam quản lý nhiễu bằng cách tính toán và sử dụng đồng thời hai moment: momentum và bình phương động. Moment bình phương động giúp giảm nhiễu bằng cách lấy bình phương của gradient, giúp làm giảm độ nhạy của tốc độ học đối với các biến động nhỏ của gradient.

# TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY ĐỂ GIẢI QUYẾT MỘT BÀI TOÁN NÀO ĐÓ

## Continual Learning

Continual Learning, hay còn được biết đến là Lifelong Learning, được xây dựng trên nền tảng của ý tưởng về việc không ngừng học hỏi về thế giới xung quanh để tự động mở rộng khả năng và liên tục phát triển các kỹ năng và kiến thức phức tạp.

Một hệ thống Continual Learning có thể được định nghĩa như một thuật toán thích ứng có khả năng học từ một luồng thông tin liên tục, với thông tin đó dần dần trở nên có sẵn theo thời gian và số lượng nhiệm vụ cần học (ví dụ: các lớp thành viên trong một nhiệm vụ phân loại) không được định trước. Điều quan trọng nhất là việc tích hợp thông tin mới này phải diễn ra mà không làm mất thông tin cũ hoặc tạo ra sự can thiệp.

Continual learning đối mặt với một số thách thức lớn, chẳng hạn như:

* **Catastrophic forgetting:** hiện tượng một mô hình học máy bị mất đi hoặc làm giảm hiệu suất của các nhiệm vụ đã học trước đó khi học các nhiệm vụ mới.
* **Task boundary detection:** việc xác định khi nào một nhiệm vụ mới bắt đầu và khi nào nó kết thúc, đặc biệt khi dữ liệu được cung cấp theo dạng luồng liên tục.
* **Data imbalance:** hiện tượng một số nhiệm vụ có nhiều dữ liệu hơn các nhiệm vụ khác, dẫn đến sự thiên vị trong quá trình học.

Continual learning là một lĩnh vực nghiên cứu đầy tiềm năng và thách thức, có thể ứng dụng vào nhiều lĩnh vực như:

* Công nghệ trợ lý ảo
* Xử lý ngôn ngữ tự nhiên
* Hệ thống vận tải thông minh
* Nhận dạng khuôn mặt
* Phân loại hình ảnh
* Tăng cường trí tuệ nhân tạo

## Test Production

Test Production là một thuật ngữ được sử dụng để chỉ quá trình kiểm tra và đánh giá hiệu suất của một mô hình học máy hoặc học sâu trước khi đưa vào sử dụng thực tế. Chúng ta quen thuộc với việc đánh giá khi chúng ta huấn luyện một mô hình và đánh giá hiệu suất của nó trên một tập kiểm tra không nhìn thấy trước, điều này được thực hiện thông qua các chỉ số.

Ngược lại, Test Production liên quan đến việc kiểm tra hành vi của mô hình. Các kiểm thử trước khi huấn luyện - có thể chạy mà không cần các tham số đã được huấn luyện - kiểm tra xem logic đã được viết của chúng ta có đúng hay không. Ví dụ, xác suất phân loại có nằm trong khoảng từ 0 đến 1 không? Các kiểm thử sau khi huấn luyện kiểm tra xem logic đã học được có đúng như mong đợi hay không.

Test production có thể bao gồm các bước như:

* Chia dữ liệu thành tập huấn luyện, tập kiểm tra và tập xác thực.
* Huấn luyện mô hình trên tập huấn luyện và tinh chỉnh các tham số trên tập xác thực.
* Đánh giá mô hình trên tập kiểm tra bằng các phép đo như độ chính xác, sai số, độ nhạy, độ đặc trưng, v.v.
* So sánh mô hình với các mô hình cơ sở hoặc các mô hình khác trên cùng một bài toán.
* Kiểm tra mô hình trên các tình huống thực tế, chẳng hạn như dữ liệu mới, dữ liệu nhiễu, dữ liệu không cân bằng, v.v.
* Xác định các điểm mạnh và điểm yếu của mô hình, cũng như các hướng cải thiện.

## Mối liên hệ giữa Continual Learning và Test Production

* Continual Learning giúp mô hình học từ dữ liệu mới, trong khi Test Production đảm bảo rằng mô hình hoạt động đúng đắn khi triển khai trong môi trường sản xuất.
* Khi triển khai Continual Learning, quá trình kiểm thử cần được thực hiện một cách kỹ lưỡng để đảm bảo rằng các cập nhật mô hình không làm suy giảm hiệu suất trên dữ liệu cũ và mới.

## TÀI LIỆU THAM KHẢO

Tiếng Việt

1. [Optimizer- Hiểu sâu về các thuật toán tối ưu ( GD,SGD,Adam,..) (viblo.asia)](https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8)
2. [Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/01/16/gradientdescent2/)
3. [Thuật toán tối ưu adam (viblo.asia)](https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m)
4. [Machine Learning và Deep Learning: Sự khác biệt là gì? (funix.edu.vn)](https://funix.edu.vn/hoi-dap-cntt/machine-learning-va-deep-learning-su-khac-biet-la-gi/)

Tiếng Anh

1. [Gradient Descent For Machine Learning - MachineLearningMastery.com](https://machinelearningmastery.com/gradient-descent-for-machine-learning/)
2. [Stochastic gradient descent - Wikipedia](https://en.wikipedia.org/wiki/Stochastic_gradient_descent)
3. [Gradient Descent With Momentum from Scratch - MachineLearningMastery.com](https://machinelearningmastery.com/gradient-descent-with-momentum-from-scratch/)
4. [Gradient Descent With AdaGrad From Scratch - MachineLearningMastery.com](https://machinelearningmastery.com/gradient-descent-with-adagrad-from-scratch/)
5. [Adagrad: An Adaptive Gradient Algorithm for Optimization (opengenus.org)](https://iq.opengenus.org/adagrad/#:~:text=The%20key%20idea%20behind%20Adagrad%20is%20to%20accumulate,%3D%20%CE%B8%20-%20%28%CE%B7%20%2F%20%E2%88%9AG%29%20%2A%20g)
6. [Gradient Descent With RMSProp from Scratch - MachineLearningMastery.com](https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/)
7. <https://keras.io/api/optimizers/>
8. [RMSProp - Cornell University Computational Optimization Open Textbook - Optimization Wiki](https://optimization.cbe.cornell.edu/index.php?title=RMSProp#RMSProp)
9. [Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - MachineLearningMastery.com](https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/)
10. [Introduction to Continual Learning - Wiki (continualai.org)](https://wiki.continualai.org/the-continualai-wiki/introduction-to-continual-learning)
11. [Machine Learning in Production - Testing (applyingml.com)](https://applyingml.com/resources/testing-ml/)

Tiếng Trung

1. [「连续学习Continual learning, CL」最新2022研究综述 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/534219613)
2. [什么是持续学习/终身学习（Continual Learning/Life-Long Learning） - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/553122619)
3. [Arxiv 8.23 持续学习/增量学习/终身学习 continual learning/incremental learning/lifelong learning - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/651849307)