TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

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

*Người thực hiện*: **THÁI TIẾN HOA – 52000046**

Khoá  **: K24**

**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**



**BÁO CÁO CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

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

*Người thực hiện*: **THÁI TIẾN HOA – 52000046**

Khoá  **: K24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến Thầy PGS TS. Lê Anh Cường đã hướng dẫn. Bài báo cáo này không thể hoàn thành được nếu không có sự hỗ trợ, giúp đỡ, và chia sẻ kiến thức từ thầy.

Em cũng muốn cảm ơn đến các nhà phát triển của các công nghệ, công cụ và thư viện mà tôi đã sử dụng trong quá trình nghiên cứu và thực hành. Các sản phẩm của họ đã giúp em tiết kiệm thời gian và công sức trong quá trình làm việc.

TÓM TẮT

Trong quá trình huấn luyện mô hình học máy, việc chọn lựa phương pháp tối ưu hóa là một yếu tố quan trọng đối với hiệu suất và tốc độ hội tụ của mô hình. Có nhiều phương pháp Optimizer như Stochastic Gradient Descent (SGD), Adam, RMSprop, và nhiều hơn nữa. Mỗi phương pháp có những ưu điểm và nhược điểm riêng, tùy thuộc vào đặc tính của dữ liệu và kiến trúc mô hình. Ví dụ, Adam thường được ưa chuộng do hiệu suất tốt trên nhiều loại bài toán, trong khi SGD đơn giản và hiệu quả trên dữ liệu lớn.

Hơn nữa, khi xây dựng một giải pháp học máy, việc áp dụng Continual Learning và Test Production trở nên quan trọng hơn bao giờ hết. Continual Learning giúp mô hình cập nhật thông tin liên tục từ dữ liệu mới mà không quên kiến thức cũ, giúp mô hình thích ứng và hiệu quả hơn theo thời gian. Trong khi đó, quá trình Test Production đảm bảo rằng mô hình được kiểm tra một cách toàn diện về hiệu suất, tích hợp, độ bền và an ninh, đảm bảo rằng nó đáp ứng đúng yêu cầu và có thể triển khai một cách an toàn và hiệu quả trong thực tế.

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc154315191)

[TÓM TẮT 2](#_Toc154315192)

[MỤC LỤC 3](#_Toc154315193)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 5](#_Toc154315194)

[CHƯƠNG 1 – TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY 6](#_Toc154315195)

[2.1 Gradient Descent (GD) 6](#_Toc154315196)

[2.1.2 Gradient descent cho hàm nhiều biến 10](#_Toc154315197)

[2.2 Stochastic Gradient Descent (SGD) 10](#_Toc154315198)

[2.3 Momentum 12](#_Toc154315199)

[2.4 Aagrad 15](#_Toc154315200)

[2.5 RMSprop 16](#_Toc154315201)

[2.6 Adam 16](#_Toc154315202)

[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 17](#_Toc154315203)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

**No table of figures entries found.**

**DANH MỤC BẢNG**

**No table of figures entries found.**

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

1. Optimizer là gì, tại sao phải dùng?

Trước khi đi sâu vào vấn đề thì chúng ta cần hiểu thế nào là thuật toán tối ưu (optimizers).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, từ đó có thể tìm 1 cặp weights và bias phù hợp để tối ưu hóa model. Nhưng vấn đề là "học" như thế nào? Cụ thể là weights và bias được tìm như thế nào! Đâu phải chỉ cần random (weights, bias) 1 số lần hữu hạn và hy vọng ở 1 bước nào đó ta có thể tìm được lời giải. Rõ ràng là không khả thi và lãng phí tài nguyên! Chúng ta phải tìm 1 thuật toán để cải thiện weight và bias theo từng bước, và đó là lý do các thuật toán optimizer ra đời.

1. Các thuật toán tối ưu ?
   1. Gradient Descent (GD)

Trong các bài toán tối ưu, chúng ta thường tìm giá trị nhỏ nhất của 1 hàm số nào đó, mà hàm số đạt giá trị nhỏ nhất khi đạo hàm bằng 0. Nhưng đâu phải lúc nào đạo hàm hàm số cũng được, đối với các hàm số nhiều biến thì đạo hàm rất phức tạp, thậm chí là bất khả thi. Nên thay vào đó người ta tìm điểm gần với điểm cực tiểu nhất và xem đó là nghiệm bài toán. Gradient Descent dịch ra tiếng Việt là giảm dần độ dốc, nên hướng tiếp cận ở đây là chọn 1 nghiệm ngẫu nhiên cứ sau mỗi vòng lặp (hay epoch) thì cho nó tiến dần đến điểm cần tìm.

Công thức : xnew = xold - learningrate.gradient(x)

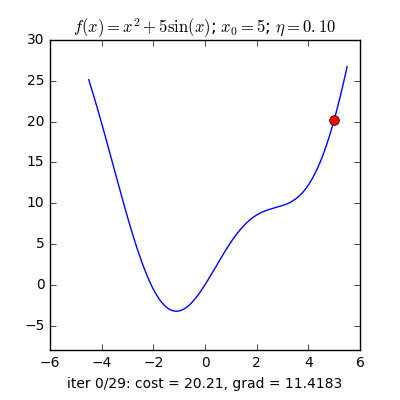
Đặt câu hỏi tại sao có công thức đó ? Công thức trên được xây dựng để cập nhật lại nghiệm sau mỗi vòng lặp . Dấu '-' trừ ở đây ám chỉ ngược hướng đạo hàm. Đặt tiếp câu hỏi tại sao lại ngược hướng đạo hàm ?

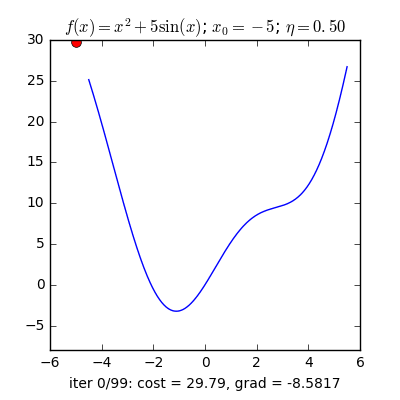
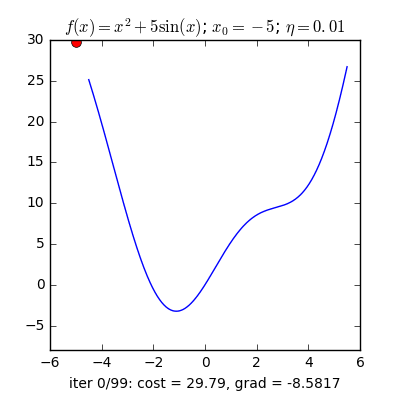
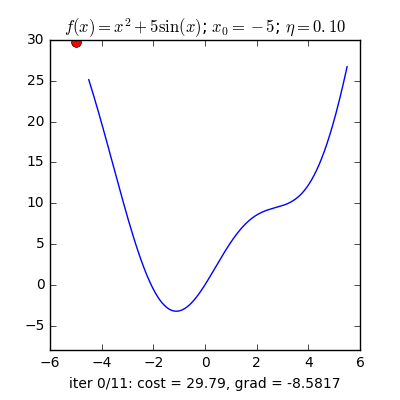
Ví dụ như đối với hàm f(x)= 2x +5sin(x) như hình dưới thì f'(x) =2x + 5cos(x)

với x\_old =-4 thì f'(-4) <0 => x\_new > x\_old nên nghiệm sẽ di chuyển về bên phải tiến gần tới điểm cực tiểu.

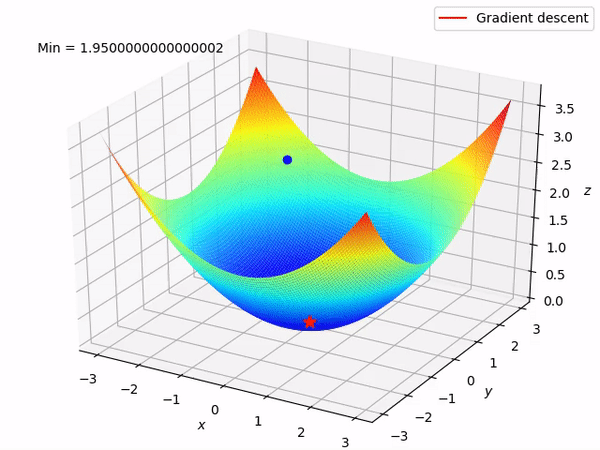
ngược lại với x\_old =4 thì f'(4) >0 => x\_new <x\_old nên nghiệm sẽ di chuyển về bên trái tiến gần tới điểm cực tiểu.

* + 1. Gradient cho hàm 1 biến





Qua các hình trên ta thấy Gradient descent phụ thuộc vào nhiều yếu tố : như nếu chọn điểm x ban đầu khác nhau sẽ ảnh hưởng đến quá trình hội tụ; hoặc tốc độ học (learning rate) quá lớn hoặc quá nhỏ cũng ảnh hưởng: 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.

* + 1. Gradient descent cho hàm nhiều biến

Ưu điểm :

 Thuật toán gradient descent cơ bản, dễ hiểu. Thuật toán đã giải quyết được vấn đề tối ưu model neural network bằng cách cập nhật trọng số sau mỗi vòng lặp.

Nhược điểm :

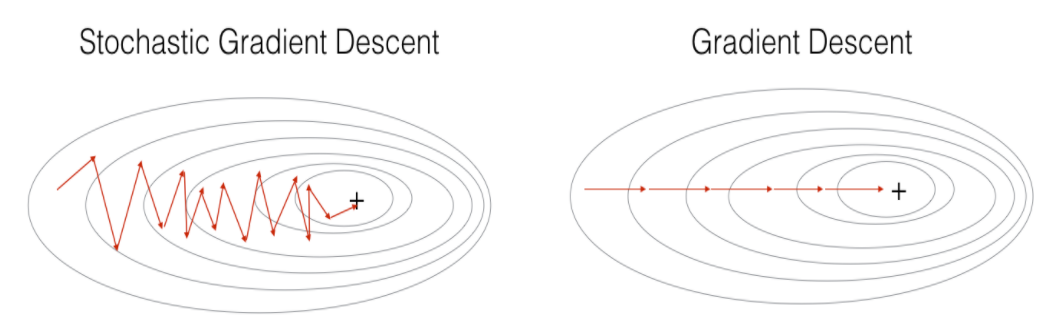
 Vì đơn giản nên thuật toán Gradient Descent còn nhiều hạn chế như phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.

 Ví dụ 1 hàm số có 2 global minimum thì tùy thuộc vào 2 điểm khởi tạo ban đầu sẽ cho ra 2 nghiệm cuối cùng khác nhau.

 Tốc độ học quá lớn sẽ khiến cho thuật toán không hội tụ, quanh quẩn bên đích vì bước nhảy quá lớn; hoặc tốc độ học nhỏ ảnh hưởng đến tốc độ training

2.2 Stochastic Gradient Descent (SGD)

Stochastic là 1 biến thể của Gradient Descent . Thay vì sau mỗi epoch chúng ta sẽ cập nhật trọng số (Weight) 1 lần thì trong mỗi epoch có N điểm dữ liệu chúng ta sẽ cập nhật trọng số N lần. Nhìn vào 1 mặt , SGD sẽ làm giảm đi tốc độ của 1 epoch. Tuy nhiên nhìn theo 1 hướng khác,SGD sẽ hội tụ rất nhanh chỉ sau vài epoch. Công thức SGD cũng tương tự như GD nhưng thực hiện trên từng điểm dữ liệu.



Nhìn vào 2 hình trên, ta thấy SGD có đường đi khá là zig zắc , không mượt như GD. Dễ hiểu điều đó vì 1 điểm dữ liệu không thể đại diện cho toàn bộ dữ liệu. Đặt câu hỏi tại sao phải dùng SGD thay cho GD mặt dù đường đi của nó khá zig zắc ? Ở đây, GD có hạn chế đối với cơ sở dữ liệu lớn ( vài triệu dữ liệu ) thì việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh. Bên cạnh đó GD không phù hợp với online learning. Vậy online learning là gì? online learning là khi dữ liệu cập nhật liên tục (ví dụ như thêm người dùng đăng kí ) thì mỗi lần thêm dữ liệu ta phải tính lại đạo hàm trên toàn bộ dữ liệu => thời gian tính toán lâu, thuật toán không online nữa. Vì thế SGD ra đời để giải quyết vấn đề đó, vì mỗi lần thêm dữ liệu mới vào chỉ cần cập nhật trên 1 điểm dữ liệu đó thôi, phù hợp với online learning.

Một ví dụ minh hoạ : có 10.000 điểm dữ liệu thì chỉ sau 3 epoch ta đã có được nghiệm tốt, còn với GD ta phải dùng tới 90 epoch để đạt được kết quả đó.

Ưu điểm :

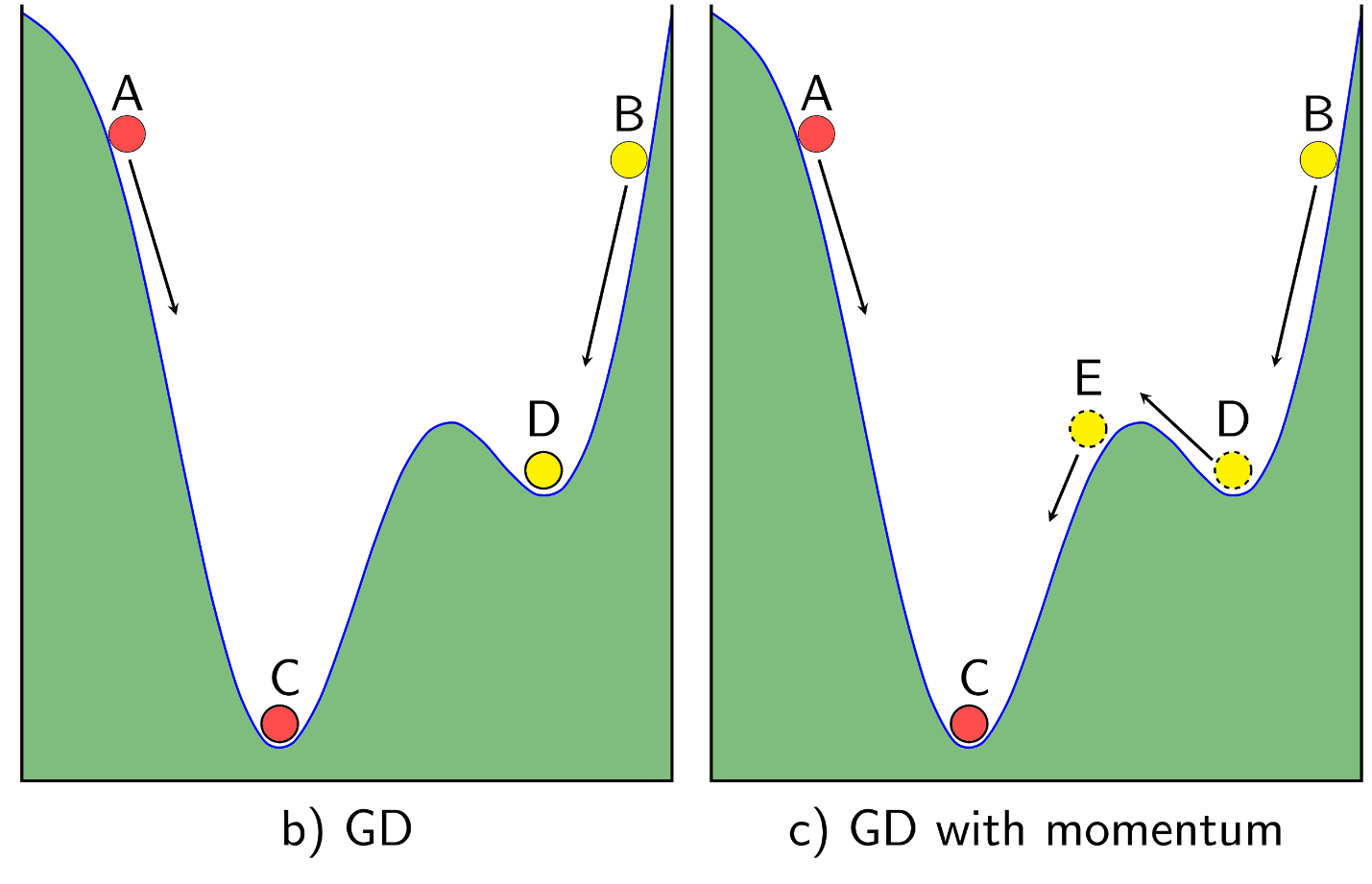
 Thuật toán giải quyết được đối với cơ sở dữ liệu lớn mà GD không làm được. Thuật toán tối ưu này hiên nay vẫn hay được sử dụng.

Nhược điểm :

 Thuật toán vẫn chưa giải quyết được 2 nhược điểm lớn của gradient descent ( learning rate, điểm dữ liệu ban đầu ). Vì vậy ta phải kết hợp SGD với 1 số thuật toán khác như: Momentum, AdaGrad,..

* 1. Momentum

Để khắc phục các hạn chế trên của thuật toán Gradient Descent người ta dùng gradient descent with momentum. Vậy gradient with momentum là gì ?

 Để giải thích được Gradient with Momentum thì trước tiên ta nên nhìn dưới góc độ vật lí: Như hình b phía trên, nếu ta thả 2 viên bi tại 2 điểm khác nhau A và B thì viên bị A sẽ trượt xuống điểm C còn viên bi B sẽ trượt xuống điểm D, nhưng ta lại không mong muốn viên bi B sẽ dừng ở điểm D (local minimum) mà sẽ tiếp tục lăn tới điểm C (global minimum). Để thực hiện được điều đó ta phải cấp cho viên bi B 1 vận tốc ban đầu đủ lớn để nó có thể vượt qua điểm E tới điểm C. Dựa vào ý tưởng này người ta xây dựng nên thuật toán Momentum ( tức là theo đà tiến tới ).

Nhìn dưới góc độ toán học, ta có công thức Momentum:

xnew = xold -(gama.v + learningrate.gradient)

Trong đó :

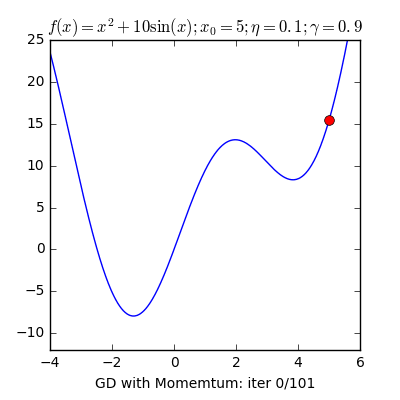
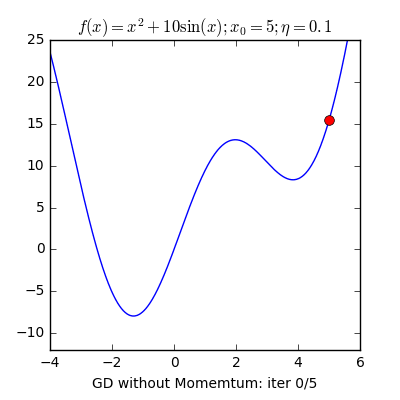
xnew: tọa độ mới

xod : tọa độ cũ

gama: parameter , thường =0.9

learningrate : tốc độ học

gradient : đạo hàm của hàm f



Qua 2 ví dụ minh họa trên của hàm f(x) = x.2 + 10sin(x), ta thấy GD without momentum sẽ hội tụ sau 5 vòng lặp nhưng không phải là global minimum. Nhưng GD with momentum dù mất nhiều vòng lặp nhưng nghiệm tiến tới global minimum, qua hình ta thấy nó sẽ vượt tốc tiến tới điểm global minimum và dao động qua lại quanh điểm đó trước khai dừng lại.

**Ưu điểm:**

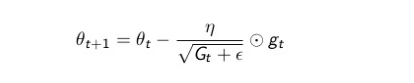
 Thuật toán tối ưu giải quyết được vấn đề: Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.

**Nhược điểm:**

 Tuy momentum giúp hòn bi vượt dốc tiến tới điểm đích, tuy nhiên khi tới gần đích, nó vẫn mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn, điều này được giải thích vì viên bi có đà.

* 1. Aagrad

Không giống như các thuật toán trước đó thì learning rate hầu như giống nhau trong quá trình training (learning rate là hằng số), Adagrad coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên sau mỗi thời điểm t.



Trong đó :

n : hằng số

gt : gradient tại thời điểm t

ϵ : hệ số tránh lỗi ( chia cho mẫu bằng 0)

G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

Ưu điểm :

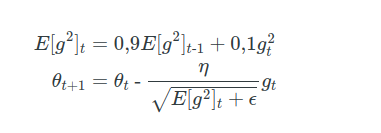
 Một lơi ích dễ thấy của Adagrad là tránh việc điều chỉnh learning rate bằng tay, chỉ cần để tốc độ học default là 0.01 thì thuật toán sẽ tự động điều chỉnh.

Nhược điểm :

 Yếu điểm của Adagrad là tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

* 1. 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.



Ưu điểm :

 Ưu điểm rõ nhất của RMSprop là giải quyết được vấn đề tốc độ học giảm dần của Adagrad ( vấn đề tốc độ học giảm dần theo thời gian sẽ khiến việc training chậm dần, có thể dẫn tới bị đóng băng )

Nhược điểm :

 Thuật toán RMSprop có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum như Momentum. Vì vậy người ta sẽ kết hợp cả 2 thuật toán Momentum với RMSprop cho ra 1 thuật toán tối ưu Adam. Chúng ta sẽ trình bày nó trong phần sau.

* 1. Adam

Như đã nói ở trên Adam là sự kết hợp của Momentum và RMSprop . Nếu giải thích theo hiện tượng vật lí thì Momentum giống như 1 quả cầu lao xuống dốc, còn Adam như 1 quả cầu rất nặng có ma sát, vì vậy nó dễ dàng vượt qua local minimum tới global minimum và khi tới global minimum nó không mất nhiều thời gian dao động qua lại quanh đích vì nó có ma sát nên dễ dừng lại hơn.

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

1. Continual Learning

* Continual Learning (Học Liên Tục): Continual Learning (CL) là một lĩnh vực trong học máy tập trung vào khả năng của một mô hình học máy để liên tục học từ dữ liệu mới mà không quên đi kiến thức đã học từ dữ liệu cũ. Điều này là quan trọng trong các ứng dụng yêu cầu mô hình thích ứng với sự thay đổi liên tục của dữ liệu mà không cần phải huấn luyện lại từ đầu. Một số thách thức trong học liên tục bao gồm:
* Quên (Catastrophic Forgetting): Mô hình có thể quên thông tin từ quá khứ khi được huấn luyện trên dữ liệu mới.
* Interference (Nhiễu): Học từ dữ liệu mới có thể làm ảnh hưởng đến hiệu suất của mô hình trên dữ liệu cũ.
* Scalability (Khả năng Mở Rộng): Khả năng mở rộng để xử lý nhiều nhiệm vụ một cách hiệu quả.

Các kỹ thuật trong học liên tục bao gồm Elastic Weight Consolidation (EWC), Synaptic Intelligence, và Progressive Neural Networks. Đối với mô hình học liên tục, quản lý một bộ nhớ ngoài (external memory) để lưu trữ thông tin quan trọng từ quá khứ và sử dụng kỹ thuật như rehearsal (luyện tập lại) để tái sử dụng dữ liệu cũ là những chiến lược phổ biến.

1. Test Production (Sản Xuất Kỳ Thị):

Test Production là quá trình tạo ra dữ liệu kiểm thử để đánh giá hiệu suất của mô hình học máy trên một bài toán cụ thể. Khi xây dựng giải pháp học máy, quá trình này rất quan trọng để đảm bảo rằng mô hình đang hoạt động đúng đắn và có thể tự tin được triển khai trong môi trường thực tế. Các bước quan trọng trong quá trình sản xuất kỳ thị bao gồm:

* Chuẩn bị Dữ Liệu Kiểm Thử: Tạo ra một tập dữ liệu kiểm thử đrepresentative và đa dạng, phản ánh mọi khía cạnh của dữ liệu mà mô hình có thể gặp trong thực tế.
* Kiểm Thử Hiệu Suất (Performance Testing): Đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử để đảm bảo nó đáp ứng yêu cầu về độ chính xác, tốc độ, và các yếu tố khác.
* Kiểm Thử Tích Hợp (Integration Testing): Đảm bảo rằng mô hình tích hợp một cách đúng đắn với các thành phần khác của hệ thống.
* Kiểm Thử Robustness (Kiểm Thử Độ Bền): Đánh giá khả năng của mô hình chống lại nhiễu, dữ liệu ngoại lai, và điều kiện khác ngoài dự đoán bình thường.
* Kiểm Thử An Toàn và Bảo Mật (Security Testing): Đảm bảo rằng mô hình không bị tấn công và đáp ứng các yêu cầu an toàn và bảo mật.

Quá trình sản xuất kỳ thị giúp đảm bảo rằng mô hình có thể hoạt động đúng đắn trong mọi tình huống và là một phần quan trọng của quy trình triển khai giải pháp học máy.

1. Áp dụng xây dựng một giải pháp học máy để giải quyết một bài toán nào đó

Nhận diện đối tượng trong ảnh (Object Detection)

Continual Learning:

Giả sử bạn đang phát triển một mô hình nhận diện đối tượng trong ảnh cho một hệ thống an ninh. Mỗi ngày, hệ thống nhận được dữ liệu mới từ nhiều nguồn khác nhau, bao gồm cả đối tượng mới và biến đổi về điều kiện ánh sáng, thời tiết, và góc chụp.

* Quản lý Quên (Catastrophic Forgetting): Sử dụng kỹ thuật như Elastic Weight Consolidation (EWC) để giữ lại trọng số quan trọng từ quá khứ và tránh quên thông tin quan trọng khi học từ dữ liệu mới.
* Bộ Nhớ Ngoài (External Memory): Lưu trữ một bộ nhớ ngoài để lưu trữ đặc điểm của đối tượng đã học, giúp mô hình nhớ các đối tượng cũ khi đối mặt với đối tượng mới.

Test Production:

* Chuẩn Bị Dữ Liệu Kiểm Thử: Tạo ra một tập dữ liệu kiểm thử đa dạng với nhiều điều kiện khác nhau, bao gồm đối tượng mới, ánh sáng yếu, và ảnh ngoại lai.
* Kiểm Thử Hiệu Suất: Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm thử, đảm bảo rằng nó có độ chính xác cao và có thể đối mặt với các tình huống đặc biệt.
* Kiểm Thử Tích Hợp: Đảm bảo rằng mô hình tích hợp một cách chính xác với hệ thống an ninh tổng thể và có thể tương tác với các thành phần khác như camera, hệ thống cảnh báo, v.v.
* Kiểm Thử Độ Bền: Mô hình nên được kiểm thử với các điều kiện khác nhau, như ánh sáng yếu, thời tiết xấu, và đối tượng nhỏ để đảm bảo rằng nó làm việc đáng tin cậy.
* Kiểm Thử An Toàn và Bảo Mật: Đảm bảo rằng mô hình không bị tấn công, và kiểm tra khả năng chống lại các dạng tấn công thông thường như đầu vào độc hại

Áp dụng những nguyên tắc của Continual Learning và Test Production giúp đảm bảo rằng mô hình không chỉ đáp ứng hiệu quả với dữ liệu mới mà còn là một phần đáng tin cậy của hệ thống tổng thể trong ứng dụng nhận diện đối tượng trong ảnh.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1] Slides bài giảng do giảng viên cung cấp

(Thực, 2020)

**Tiếng Anh**