TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**NGUYỄN MINH PHÚ - 52100920**

**DỰ ÁN 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**



**NGUYỄN MINH PHÚ - 52100920**

**DỰ ÁN CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

Người hướng dẫn

**PGS. Lê Anh Cường**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Chân thành cảm ơn thầy Cường đã tận tình giảng dạy bộ môn này trong suốt học kỳ vừa qua. Dù môn học này khó đối với sinh viên nhưng thầy vẫn luôn tìm cách để giảng dạy, liên hệ một cách trực quan nhất để phai mờ lớp áo trừu tượng và khô khan của môn học. Đây cũng là một môn học có tính ứng dụng nhiều trong lĩnh vực trí tuệ nhân tạo, và nhiều lĩnh vực mới trong công nghệ hiện nay nên em cũng rất vui khi được tạo điều kiện để được học một môn học có tính thực tiễn cao như thế này.

Chúc thầy nhiều sức khoẻ, và vẫn giữ nguyên vẹn một niềm tin với công việc giảng dạy. Đồng thời có nhiều đóng góp cho công nghệ.

*TP. Hồ Chí Minh, ngày 21 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

Nguyễn Minh Phú

**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. Nguyễn Văn A. 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 21 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

Nguyễn Minh Phú

**MỤC LỤC**

[DANH MỤC HÌNH VẼ iv](#_Toc154052085)

[DANH MỤC BẢNG BIỂU v](#_Toc154052086)

[DANH MỤC CÁC CHỮ VIẾT TẮT vi](#_Toc154052087)

[CHƯƠNG 1. BÀI LÀM CÁ NHÂN 1](#_Toc154052088)

[1.1 Optimizer: 1](#_Toc154052089)

[1.1.1 Gradient Descent (GD) 1](#_Toc154052090)

[1.1.2 Stochastic Gradient Descent (SGD) 2](#_Toc154052091)

[1.1.3 Momentum 3](#_Toc154052092)

[1.1.4 Adagrad 4](#_Toc154052093)

[1.1.5 RMSprop 5](#_Toc154052094)

[1.1.6 Adam 5](#_Toc154052095)

[1.1.7 Tổng quát 6](#_Toc154052096)

[1.2 Continual Learning và Test Production trong học máy: 8](#_Toc154052097)

[1.2.1 Continual Learning 8](#_Toc154052098)

[1.2.2 Test Production 10](#_Toc154052099)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 13](#_Toc154052100)

[2.1 Mạng neural hồi quy 13](#_Toc154052101)

[2.1.1 Recurrent Neural Network (RNN) 13](#_Toc154052102)

[TÀI LIỆU THAM KHẢO 14](#_Toc154052103)

# DANH MỤC HÌNH VẼ

[Hình 1.1 Gradient cho hàm một biến 1](file:///D:\University\ML\Final\52100920_NguyenMinhPhu.docx#_Toc154052104)

[Hình 1.2 SGD vs GD 2](#_Toc154052105)

[Hình 1.4 Gradient Descent dưới góc nhìn vật lý (Machine Learning Cơ Bản, n.d.) 4](#_Toc154052106)

[Hình 1.5 Quả cầu nặng với ma sát trong Adam 6](#_Toc154052107)

# DANH MỤC BẢNG BIỂU

[Bảng 1.1 So sánh các thuật toán optimizer thường dùng 7](#_Toc154052135)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| GD | Gradient Descent |
| SGD | Stochastic Gradient Descent |

# BÀI LÀM CÁ NHÂN

## Optimizer:

Trong phần này, tôi tập trung vào việc tìm hiểu, đánh giá và so sánh các phương pháp optimizer phổ biến trong huấn luyện mô hình học máy. Mục tiêu là xác định ưu, nhược điểm của từng phương pháp để có thể hiểu rõ, vận dụng hoặc kết hợp để cải thiện hiệu suất mô hình.

Đầu tiên, ta nhắc lại về optimizer hay còn gọi là thuật toán tối ưu. Có thể hiểu, optimizer là cơ sở để xây dựng mô hình mạng neural với mục đích học được các features của dữ liệu đầu vào, từ đó có thể tìm được 1 cặp weights và bias (còn có thể được gọi là các tham số của mô hình) phù hợp để tối ưu hóa model, giảm thiểu lỗi trong quá trình huấn luyện. Và việc điều chỉnh các tham số này, chúng ta cần một thuật toán để cải thiện theo từng bước, và đó là lý do các thuật toán optimizer ra đời. Có thể kể đến như Gradient Descent, Stochastic Gradient Descent, Momentum, Adagrad, RMSprop, Adam, v.v

Ta cùng tìm hiểu sâu hơn về các thuật toán được đề cập bên trên sau đây.

### Gradient Descent (GD)

Đây là một thuật toán tối ưu phổ biến trong học máy, nó thường được sử dụng để tìm giá trị nhỏ nhất của một hàm mục tiêu, hay có thể gọi là tìm cực tiểu toàn cục. Một cách phổ biến để tìm giá trị cực tiểu toàn cục này là thực hiện giải phương trình đạo hàm của hàm số này bằng không. Nhưng không phải lúc nào đạo hàm một hàm số cũng khả thi, nên thay vào đó, người ta tìm điểm gần với giá trị cực tiểu này nhất và xem đó là nghiệm của bài toán. Với công thức:

Hình 1.1 Gradient cho hàm một biến

(*Machine Learning Cơ Bản*, n.d.)

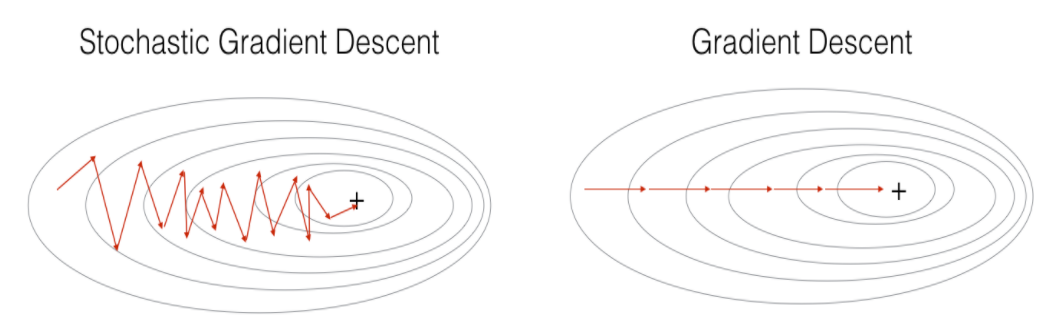
Trong công thức trên, là learning rate và là gradient của hàm mục tiêu tại điểm x và x là giá trị ta tìm được ở vòng lặp hiện tại. Một cách tổng quát, với thuật toán này, gradient descent sẽ cố gắng thực hiện tìm cực tiểu của hàm mất mát bằng cách cập nhật tham số theo hướng ngược lại với gradient của hàm mất mát. Tuy nhiên với một số hàm số, giá trị cực tiểu này chỉ có thể là cực tiểu địa phương, để giải quyết vấn đề này, ta có thể sử dụng các thuật toán khác được đề cập ở phần tiếp theo.

Qua đó, ta cũng có thể dễ dàng nhận thấy, Gradient descent phụ thuộc vào các yếu tố như vị trí chọn x ban đầu, learning rate nếu quá lớn thì có thể tiến nhanh tới đích sau vài vòng lặp tuy nhiên thuật toán không hội tụ, chỉ quanh quẩn quanh đích vì bước nhảy quá lớn, hoặc nếu quá nhỏ thì tốc độ hội tụ chậm sẽ ảnh hưởng đến quá trình training.

### Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent là một biến thể của Gradient Descent. Thay vì như thuật toán phía trên, sau mỗi epoch, ta thực hiện cập nhật trọng số một lần, thì với SGD tại một thời điểm, ta chỉ tính gradient dựa trên chỉ một điểm dữ liệu , tương ứng, trong mỗi epoch có N điểm dữ liệu thì sẽ là N lần cập nhật.

Như vậy, ở một khía cạnh nào đó, SGD có thể làm giảm tốc độ của một epoch, nhưng, nó sẽ giúp tốc độ hội tụ rất nhanh chỉ sau vài epoch bởi nó chỉ yêu cầu một lượng epoch rất nhỏ. Công thức của SGD cũng tương tự như GD nhưng thực hiện trên từng điểm dữ liệu của một mẫu dữ liệu:



Hình 1.2 SGD vs GD

Nhìn vào hình 1.2, đường đi của SGD sẽ không được mượt như GD. Điều này là dễ hiểu vì một điểm dữ liệu không đại diện cho toàn bộ dữ liệu. Bên cạnh đó, GD còn có hạn chế là, đối với cơ sở dữ liệu lớn, thì việc tính toán đạo hàm trên bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh; đồng thời, Gradient Descent không phù hợp với online learning (là khi dữ liệu cập nhật liên tục, thì mỗi lần thêm dữ liệu mới, ta lại phải tính đạo hàm trên toàn bộ dữ liệu) làm cho thời gian tính toán trở nên lâu hơn mà Stochastic Gradient Descent ra đời để khắc phục vấn đề đó (mỗi lần thêm một dữ liệu mới chỉ cập nhật một điểm đó). Tóm lại, đặc điểm nổi bậc của SGD là nó khắc phục việc cập nhật tham số trên toàn bộ tập dữ liệu.

### Momentum

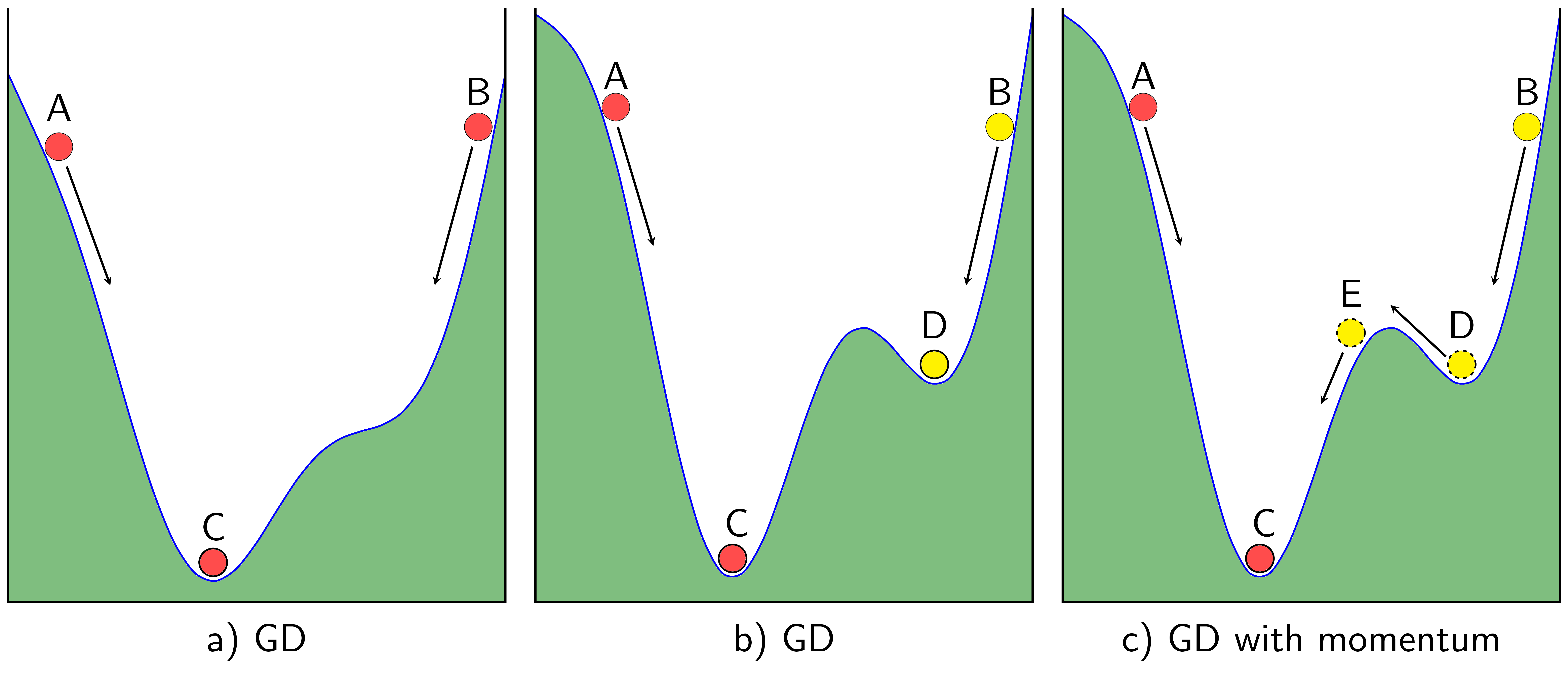
Gradient Descent with Momentum cũng là một biến thể của Gradient Descent, được sử dụng để tối ưu hóa các hàm mất mát trong học máy. Thuật toán này giúp tăng tốc độ hội tụ và giảm dao động của Gradient Descent bằng cách tính tích luỹ gradient từ các bước trước đó để tạo ra đà. Với công thức

Công thức cập nhật vận tốc:

Công thức cập nhật tham số:

Trong đó, η là tốc độ học, β là hệ số giảm thường là một giá trị khoảng 0.9, ∇f(x) là gradient của hàm mục tiêu tại x.

Để dễ hiểu hơn, ta cùng liên hệ với góc nhìn vật lý. Thuật toán Gradient Descent thường được ví như tác dụng của trọng lực lên một viên bi sắt đặt trên một bề mặt có dạng như thung lũng (hình a). Cho dù đặt ở vị trí A hay B, thì dưới tác dụng của trọng lực, nó vẫn sẽ lăn xuống vị trí C. Nhưng với bề mặt có hai đáy thung lũng (hình b) thì việc đặt bi ở vị trí A hay B thì nó sẽ cho ra hai vị trí cuối cùng khác nhau sẽ là C hoặc D, điểm D là một điểm cực tiểu cục bộ mà chúng ta không mong muốn.



Hình 1.4 Gradient Descent dưới góc nhìn vật lý (Machine Learning Cơ Bản, n.d.)

Ta cần điểm kết thúc khi đặt bi ở vị trí B là C chứ không phải D, thì nếu ta cho một vận tốc ban đầu đủ lớn thì khi bi lăn đến điểm D, theo *đà* () bi có thể tiếp tục và lên dốc phía bên trái của D, và có thể vượt dốc để tới điểm E (hình c) rồi lăn và kết thúc ở D như mong muốn nếu vận tốc ban đầu đủ lớn và thuật toán dựa trên hiện tượng này gọi là *Momentum*.

### Adagrad

Khác với các thuật toán trước đó, learning rate hầu như giống nhau trong quá trình huấn luyện (hằng số). Với Adagrad thì learning rate được coi là một giá trị, tức nó sẽ biến thiên và được điều chỉnh cho từng tham số riêng lẻ, giúp tối ưu hóa hàm mất mát.

Cụ thể, Adagrad sử dụng một ma trận đường chéo để lưu trữ tổng bình phương của các gradient đã tính toán trước đó, các tham số được cập nhật bằng cách chia learning rate cho căn bậc hai của tổng bình phương gradient. Điều này giúp giảm learning rate cho các tham số có gradient lớn và tăng learning rate cho các tham số có gradient nhỏ.

Công thức cập nhật tổng bình phương gradient:

Công thức cập nhật tham số:

Trong đó: η là tốc độ học, β là hệ số giảm, ∇f(x) là gradient của hàm mục tiêu tại x, và ϵ là một số dương nhỏ để tránh chia cho 0.

### RMSprop

RMSprop cũng tương tự như Adagrad qua việc sử dụng một learning rate được điều chỉnh cho từng tham số riêng lẻ. RMSprop sử dụng tham số beta để điều chỉnh giữa tổng bình phương gradient mới và cũ, giúp giảm độ dao động của hàm mất mát.

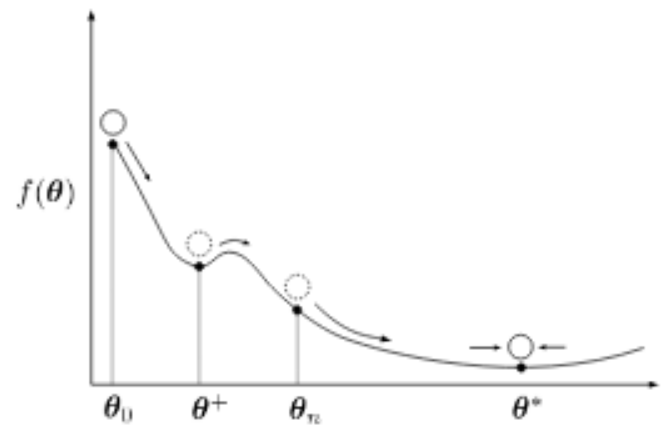
Công thức cập nhật tổng bình phương gradient:

Công thức cập nhật tham số:

### Adam

Vì thuật toán RMSprop có thể cho kết quả nghiệm là cực tiểu địa phương chứ không đạt được cục bộ như Momentum. Nên người ta kết hợp hai thuật toán trên để cho ra một thuật toán tối ưu là Adam.

Giải thích theo góc nhìn vật lý thì Momentum được ví như một quả cầu lao xuống dốc, còn Adam như quả cầu nặng có ma sát; vì vậy nó có thể dễ dàng vượt qua cực tiểu cục bộ để đạt được cực tiểu toàn cục và khi tới đích, nó cũng sẽ không tốn quá nhiều thời gian dao động, vì có ma sát nên nó dễ dừng lại hơn.



Hình 1.5 Quả cầu nặng với ma sát trong Adam

Quay trở lại thuật toán, Adam sử dụng hai tham số là momentum và learning rate; momentum giúp tính toán trung bình trọng số của các gradient trước đó, trong khi learning rate giúp xác định tốc độ học của thuật toán. Adam cũng sử dụng một tham số epsilon để tránh việc chia cho 0 khi tính toán.

Công thức cập nhật vận tốc:

Công thức cập nhật tổng bình phương gradient:

Công thức cập nhật tham số:

Trong đó: η là tốc độ học, β là hệ số giảm, ∇f(x) là gradient của hàm mục tiêu tại x, và ϵ là một số dương nhỏ để tránh chia cho 0.

### Tổng quát

Bên trên là một số các thuật toán tối ưu thường được sử dụng. Ta có thể lập một bảng so sánh và nêu lên ưu-nhược điểm như sau:

Bảng 1.1 So sánh các thuật toán optimizer thường dùng

|  |  |  |  |
| --- | --- | --- | --- |
| **Phương pháp** | **Tổng quát** | **Ưu điểm** | **Nhược điểm** |
| Gradient Descent | Cơ bản, thực hiện gradient toàn bộ dữ liệu. | Cơ bản, dễ hiểu, dễ dùng, ổn định | Phụ thuộc vào nghiệm ban đầu và learning rate.  Khó xử lý trên dữ liệu lớn, có thể cho tốc độ hội tụ chậm hoặc tốc độ huấn luyện chậm. |
| Stochastic Gradient Descent | Sử dụng một mẫu ngẫu nhiên từ dữ liệu (hoặc mini-batch nhỏ) để tính toán gradient và cập nhật tham số. | Tăng tốc độ học, thích nghi với dữ liệu không đồng đều, khắc phục việc cập nhật tham số trên toàn bộ tập dữ liệu, và đối với cơ sở dữ liệu lớn mà GD kém. | Dao động, khó kiểm soát.  Chưa giải quyết được 2 nhược điểm của GD learning rate và x ban đầu. |
| Momentum | Sử dụng đà để giúp tối ưu quá trình tối ưu hóa, kết hợp các giá trị gradient của các bước trước đó. | Hạn chế dao động, tăng tốc học.  Giải quyết được vấn đề dừng lại ở local minimum. | Phụ thuộc vào hệ số momentum, đôi khi bị chậm.  Vẫn mất khá nhiều thời gian dao động quanh đích. |
| Adagrad | Điều chỉnh learning rate riêng lẻ cho từng tham số | Hiệu quả trên dữ liệu thưa; không cần phải điều chỉnh learning rate bằng tay, chỉ cần thiết lập tốc độ học mặt định là 0.01 và thuật toán sẽ tự điều chỉnh. | Tốc độ hội tụ chậm, có thể gặp vấn đề với dữ liệu nhiễu.  Tổng bình phương biến thiên sẽ lớn theo thời gian và làm cho tốc độ học cực kỳ nhỏ, làm việc training trở nên đóng băng. |
| RMSprop | Cải tiến Adagrad, điều chỉnh trọng số | Hiệu quả trên dữ liệu nhanh; giải quyết được vấn đề tốc độ học giảm dần của Adagrad (có thể dẫn tới đóng băng) | Vẫn có thể gặp vấn đề với dữ liệu nhiễu. Có thể cho kết quả chỉ là cực tiểu địa phương chứ không phải cực tiểu toàn cục. |
| Adam | Sự kết hợp giữa Momentum và RMSprop, tính toán trung bình di động của gradient (như Momentum) và trung bình di động của gradient bình phương (như RMSprop) để điều chỉnh tốc độ học. | Hiệu suất tốt trên nhiều loại dữ liệu, ổn định. | Cần cấu hình nhiều tham số, chi phí tính toán cao. Có thể không hoạt động tốt với dữ liệu nhiễu hoặc khi cần cấu hình nhiều tham số. |

Các thuật toán trên đều có ưu nhược điểm riêng. Trong hầu hết các trường hợp Adam hoạt động tốt hơn các thuật toán khác (nên nó cũng được sử dụng rộng rãi). Tuy vậy, việc lựa chọn thuật toán optimizer còn phụ thuộc vào bài toán cụ thể và dữ liệu đang xử lý.

## Continual Learning và Test Production trong học máy:

### Continual Learning

Continual Learning là quá trình cập nhật liên tục mô hình học máy để thích ứng với sự thay đổi của dữ liệu. Điều này quan trọng khi mô hình được triển khai trong môi trường sản xuất, nơi dữ liệu có thể thay đổi theo thời gian. Continual Learning giúp mô hình luôn cập nhật và phản ánh chính xác nhất xu hướng hiện tại của dữ liệu.

Continual Learning giúp giải quyết vấn đề “quên”, một hiện tượng phổ biến trong học máy khi mô hình học một cái mới và quên đi kiến thức đã học từ trước đó. Dựa trên việc cập nhật mô hình học máy theo thời gian khi tiếp xúc với dữ liệu mới thay vì chỉ huấn luyện một lần trên một tập dữ liệu cố định. Mô hình sẽ tiếp tục học từ dữ liệu mới mà vẫn “nhớ” kiến thức đã học từ dữ liệu cũ, giúp mô hình có thể thích ứng với các tác vụ mới mà không quên kiến thức đã học.

Các kỹ thuật cụ thể được sử dụng để giải quyết catastrophic forgetting:

* Regularization-based methods: Các phương pháp này sử dụng các kỹ thuật regularization để ngăn chặn các trọng số của mô hình thay đổi quá nhiều khi học dữ liệu mới. Thêm hạn chế vào hàm mất mát trong quá trình huấn luyện. Có thể kể đến như EWC, LwF và dropout.
  + Elastic Weight Consolidation (EWC): EWC giả định rằng mỗi tham số trong mạng nơ-ron có một mức độ quan trọng đối với các tác vụ đã học. Khi học một tác vụ mới, EWC sẽ cố gắng giữ nguyên các tham số quan trọng đối với các tác vụ cũ và chỉ cập nhật những tham số ít quan trọng.
  + Learning Without Forgetting (LwF): LwF cho phép mô hình học tác vụ mới mà không quên kiến thức đã học từ các tác vụ cũ. LwF hoạt động bằng cách huấn luyện mô hình trên dữ liệu của tác vụ mới, trong khi vẫn giữ nguyên đầu ra của mô hình trên dữ liệu của các tác vụ cũ.
  + Dropout: Dropout là một kỹ thuật điều chuẩn đơn giản nhưng mạnh mẽ cho các mạng nơ-ron và các mô hình học sâu. Trong dropout, một số lượng phần trăm nhất định các nơ-ron ngẫu nhiên được chọn để loại bỏ trong quá trình huấn luyện. Điều này buộc các neural phải hoạt động độc lập và không quá phụ thuộc vào một số kết nối nhất định, giảm sự phụ thuộc lẫn nhau giữa các nơ-ron và giúp mô hình học các đặc trưng tổng quát hơn.
* Rehearsal-based methods: Các phương pháp này lưu trữ một số mẫu dữ liệu cũ và sử dụng các mẫu này để cập nhật mô hình. Các phương pháp này bao gồm memory replay và data distillation.
  + Memory Replay: Memory replay hoạt động bằng cách lưu trữ một số mẫu dữ liệu từ các tác vụ cũ (lưu trữ một tập hợp nhỏ các mẫu dữ liệu cũ – memory buffer) và sử dụng chúng trong quá trình huấn luyện cho các tác vụ mới như để nhắc nhở mô hình về các kiến thức đã học trước đó.
  + Data Distillation: Data Distillation là một phương pháp sử dụng một mô hình để tạo ra các mẫu giả (mô hình nhỏ gọn – student model) từ các tác vụ cũ và sau đó, huấn luyện mô hình trên các mẫu giả này cùng với dữ liệu của tác vụ mới.
* Dynamic architectures methods: Các phương pháp này thay đổi kiến trúc của mô hình để phù hợp với dữ liệu mới. Các phương pháp này bao gồm progressive neural networks và forgetron.
  + Progressive Neural Networks (PNN): PNN hoạt động bằng cách thêm các lớp mới vào mô hình khi học các đặc trưng tác vụ mới, thay đổi kiến trúc của mô hình (ví dụ: thêm hoặc xóa các nút hoặc lớp) để thích ứng với các tác vụ mới, trong khi giữ nguyên các lớp đã học từ các tác vụ cũ.
  + Forgetron: Forgetron hoạt động bằng cách thêm các nơ-ron vào mô hình khi có dữ liệu mới; đồng thời, loại bỏ các nơ-ron ít quan trọng nhất để giải phóng không gian cho các nơ-ron mới. Quá trình này giúp mô hình thích nghi với dữ liệu mới mà vẫn nhớ kiến thức cũ.

Một ví dụ thực tế về continual learning như các hệ thống đề xuất, gợi ý quảng cáo, video của các trang web rất nổi bậc hiện nay (shopee, youtube, facebook). Với khả năng thích ứng với sở thích thay đổi theo thời gian của người dùng để đưa ra các đề xuất, gợi ý gần nhất với điều mà người dùng đang quan tâm trong thời điểm đó.

Tóm lại, continual learning cũng là một lĩnh vực quan trọng có tìm năng trong nhiều lĩnh vực và vẫn đang được nghiên cứu để giải quyết các vấn đề liên quan đến tối ưu tài nguyên và hiệu suất.

### Test Production

Trong quá trình xây dựng một giải pháp học máy, sau khi mô hình đã được huấn luyện và tinh chỉnh, ta cần kiểm thử mô hình trên môi trường sản phẩm (production environment) trước khi triển khai. Mục tiêu là đảm bảo rằng mô hình hoạt động đúng như mong đợi, đảm bảo hiệu suất của mô hình khi tiếp xúc với dữ liệu thực tế. Kiểm thử sản phẩm giúp phát hiện các lỗi mà không thể tìm thấy khi kiểm thử trong các môi trường khác đê không gây ra hậu quả không mong muốn.

Quá trình thực hiện test production trong học máy có thể bao gồm các bước sau:

1. Chuẩn bị dữ liệu: Dữ liệu trong sản phẩm thường khác so với dữ liệu được đào tạo. Do đó, cần chuẩn bị dữ liệu sản xuất để đảm bảo mô hình hoạt động tốt trong môi trường thực tế. Gồm các bước nhỏ: Thu thập dữ liệu, chuẩn hóa dữ liệu, chia dữ liệu.
2. Huấn luyện mô hình: Mô hình học máy được huấn luyện trên tập huấn luyện.
3. Kiểm tra mô hình: mô hình được kiểm tra trên tập kiểm thử để đánh giá hiệu suất của mô hình trên dữ liệu mới.
4. Đánh giá mô hình: mô hình được đánh giá bằng các chỉ số hiệu suất phù hợp và các metric liên quan (accuracy, precision, recall, F1-score,…) để đảm bảo mô hình hoạt động tốt, và có tính tổng quát đối với dữ liệu mới.

Khi thực hiện test production trong xây dựng một mô hình cần dựa trên các nguyên tắc sau: Tính toàn vẹn của dữ liệu (tức dữ liệu phải được thu thập và chuẩn hóa cẩn thận), tính đại diện của dữ liệu (dữ liệu sản xuất phải đại diện cho dữ liệu thực tế mà mô hình sẽ gặp phải), tính thời gian thực của dữ liệu (tức dữ liệu phải được cập nhật thường xuyên để đảm bảo mô hình phản ánh chính xác các thay đổi trong thế giới thực).

Việc triển khai test production hiệu quả, có thể kể đến các phương pháp:

* A/B Testing: Là một phương pháp so sánh hai phiên bản của một mô hình để xác định phiên bản nào hoạt động tốt hơn giúp đánh giá hiệu quả của một thay đổi trên một nhóm người dùng nhỏ trước khi triển khai chính thức. Phương pháp này dễ triển khai, cho phép so sánh trực tiếp giữ hai phiên bản của mô hình; nhưng nó yêu cầu một lượng người dùng thử đủ lớn để đảm bảo kết quả thống kê là tin cậy.
* Canary Releases: Là một phương pháp triển khai dần dần một thay đổi đối với một mô hình cho một nhóm người dùng nhỏ trước khi triển khai cho toàn bộ người dùng giúp giảm rủi ro khi triển khai một thay đổi lớn trên một mô hình đang hoạt động. Phương pháp này cho phép giám sát hiệu suất trước khi triển khai rộng rãi, nhưng cũng nên cẩn thận để tránh ảnh hưởng lớn đến trải nghiệm người dùng.
* Blue-green deployment: Là phương pháp triển khai gồm hai môi trường hoàn toàn độc lập song song: môi trường green và môi trường blue. Trong đó, green version là môi trường đang hoạt động và blue version là phiên bản mới; khi blue version đã sẵn sàng người dùng sẽ được điều hướng và chuyển sang môi trường blue, nếu hoạt động tốt green version sẽ được loại bỏ, ngược lại lượng truy cập này sẽ được chuyển trở lại cho green version (rollback).

Lợi ích của việc thực hiện test production có thể kể đến như:

* Đảm bảo chất lượng của mô hình trong điều kiện thực tế: Kiểm tra mô hình trên tập dữ liệu mới giúp đảm bảo mô hình hoạt động tốt trên dữ liệu mới và đáp ứng được yêu cầu bài toán.
* Tăng độ chính xác của mô hình: Sử dụng các độ đo phù hợp để đánh giá mô hình trên tập dữ liệu mới, giúp cải thiện độ chính xác của mô hình.
* Phát hiện các vấn đề tiềm ẩn: Việc kiểm tra kết quả dự đoán của mô hình với các tập kiểm thử và huấn luyện, có thể giúp nhận biết mô hình có đang bị overfitting hay underfitting hoặc có vấn đề về hiệu suất hay không. Điều này giúp đảm bảo an toàn và độ tin cậy của mô hình khi đối mặt với các tình huống mới.

Bên cạnh đó, test production cũng đối mặt với một số thách thức, bao gồm:

* Chi phí thực hiện: Việc triển khai test production có thể đòi hỏi chi phí lớn, đặc biệt khi mô hình đặc biệt phức tạp và yêu cầu nguồn nhân lực, vật lực và thời gian đáng kể để thực hiện.
* Khó khăn về dữ liệu: Việc thu thập dữ liệu cũng là một vấn đề khó khăn đối với một số mô hình đặc biệt, nếu dữ liệu không đủ hoặc dữ liệu không đồng đều, nhiễu nhiều hoặc có biến động lớn có thể ảnh hưởng đến hiệu suất, tính tổng quá của mô hình và tính tin cậy của quá trình kiểm thử.
* Rủ ro lỗi: Test production cũng có thể gây ra một số lỗi trong hệ thống sản xuất, nếu mô hình hoạt động không mong muốn. Có thể là một vấn đề nghiêm trọng nếu mô hình gây ảnh hưởng đến một mắt xích quan trọng trong hệ thống.

Test production là một kỹ thuật quan trọng trong việc đánh giá hiệu năng của một mô hình học máy trong môi trường sản xuất. Ta cần lựa chọn một cách thức phù hợp để ứng với đều kiện thực tế. Bên cạnh đó, cần có sự tối ưu, giám sát chặt chẽ tương ứng để giảm thiểu và cảnh báo các rủi ro có thể xảy đến.

# CƠ SỞ LÝ THUYẾT

## Mạng neural hồi quy

### Recurrent Neural Network (RNN)

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

*Machine Learning cơ bản*. (n.d.). Retrieved December 20, 2023, from <https://machinelearningcoban.com/2017/01/16/gradientdescent2/>

*Optimizer- Hiểu sâu về các thuật toán tối ưu ( GD,SGD,Adam,..)*. (2020, October 17). <https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8>

*Học máy—Machine learning: Giải thích đơn giản dễ hiểu*. (n.d.). Retrieved December 20, 2023, from <https://itguru.vn/blog/machine-learning-hoc-may-don-gian-de-hieu/>

*Tại sao bạn nên kiểm thử trên môi trường Production?* (2020, March 20). <https://viblo.asia/p/tai-sao-ban-nen-kiem-thu-tren-moi-truong-production-WAyK8oOm5xX>

**Tiếng Anh**

*AdaGrad—Cornell University Computational Optimization Open Textbook—Optimization Wiki*. (n.d.). Retrieved December 20, 2023, from <https://optimization.cbe.cornell.edu/index.php?title=AdaGrad>

Li, Z., & Hoiem, D. (2016). Learning Without Forgetting. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision – ECCV 2016* (pp. 614–629). Springer International Publishing. <https://doi.org/10.1007/978-3-319-46493-0_37>

Brownlee, J. (2022, July 5). Dropout Regularization in Deep Learning Models with Keras. *MachineLearningMastery.Com*. <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>

Brownlee, J. (2018, December 2). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. *MachineLearningMastery.Com*. <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>

*9. Continual Learning and Test in Production—Designing Machine Learning Systems [Book]*. (n.d.). Retrieved December 20, 2023, from <https://www.oreilly.com/library/view/designing-machine-learning/9781098107956/ch09.html>

*Blue/green deployment—Definition & overview*. (n.d.). Sumo Logic. Retrieved December 21, 2023, from <https://www.sumologic.com/glossary/blue-green-deployment/>