VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYỄN NGỌC MINH – 521H0505**

**LEARN ABOUT OPTIMIZER, CONTINUAL LEARNING AND TEST PRODUCTION**

**FINAL PROJECT**

**INTRODUCTION TO MACHINE LEARNING**

**HO CHI MINH CITY, 2023**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYỄN NGỌC MINH – 521H0505**

**LEARN ABOUT OPTIMIZER, CONTINUAL LEARNING AND TEST PRODUCTION**

**FINAL PROJECT**

**INTRODUCTION TO MACHINE LEARNING**

Instructor

**Mr. Lê Anh Cường**

**HO CHI MINH CITY, 2023**

**ACKNOWLEDGEMENTS**

Hello Mr. Le Anh Cuong, we would like to send our sincere thanks to you throughout the study process. We truly appreciate you taking the time to ensure we understood each concept fully before moving on. Thank you also for the additional resources and study materials you provided. They have helped reinforce what we have learned and allowed us to go deeper into topics we found most interesting. We are grateful not just for your knowledge and expertise in teaching, but also for your kindness and encouragement. Thank you again for everything you have done to help us in our education.

We sincerely thank you.

*Ho Chi Minh City, December 23 năm 2023.*

*Author*

*(Sign and write full name)*

*Nguyễn Ngọc Minh*

**THE WORK IS COMPLETED AT TON DUC THANG UNIVERSITY**

I hereby declare that this is my own research project and is under the scientific guidance of Mr. Lê Anh Cường. The research content and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluation were collected by the author from different sources and clearly stated in the reference section.

Project also uses a number of comments, assessments as well as data from other authors and other organizations, all with citations and source notes.

**If any fraud is detected, I will take full responsibility for the content of my Project**. Ton Duc Thang University is not involved in copyright violations caused by me during the implementation process (if any).

*Hồ Chí Minh City, December 23, 2023.*

*Author*

*(Sign and write full name)*

*Nguyễn Ngọc Minh*

**MỤC LỤC**

[CHƯƠNG 1. TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER 1](#_Toc154266335)

[1.1 Gradient Descent (GD) 1](#_Toc154266336)

[1.1.1 Gradient Descent là gì? 1](#_Toc154266337)

[1.1.2 Hiểu rõ về công thức của Gradient Descent 1](#_Toc154266338)

[1.1.3 Tính chất của Gradient Descent 2](#_Toc154266339)

[1.2 Stochastic Gradient Descent (SGD) 4](#_Toc154266340)

[1.3 Momentum 5](#_Toc154266341)

[1.3.1 Tổng quan 5](#_Toc154266342)

[1.3.2 Nguyên tắc hoạt động 5](#_Toc154266343)

[1.4 Adagrad (Adaptive Gradient Descent) 6](#_Toc154266344)

[1.4.1 Tại sao cần điều chỉnh tốc độ học? 6](#_Toc154266345)

[1.4.2 Cơ chế hoạt động của Adagrad 6](#_Toc154266346)

[1.4.3 Ưu và nhược điểm của Adagrad 7](#_Toc154266347)

[1.5 RMS Prop (Root Mean Square) 7](#_Toc154266348)

[1.6 AdaDelta 8](#_Toc154266349)

[1.6.1 Tính chất chung 8](#_Toc154266350)

[1.6.2 Vấn đề chính với AdaGrad và RMSprop 8](#_Toc154266351)

[1.6.3 Giải pháp của AdaDelta 8](#_Toc154266352)

[1.6.4 Công thức cập nhật trong AdaDelta 8](#_Toc154266353)

[1.6.5 Ưu và nhược điểm của AdaDelta 9](#_Toc154266354)

[1.7 Adam 9](#_Toc154266355)

[1.8 So sánh các thuật toán qua adult dataset 11](#_Toc154266356)

[CHƯƠNG 2. CONTINUAL LEARNING VÀ TEST PRODUCT 12](#_Toc154266357)

[2.1 Continual learning 12](#_Toc154266358)

[2.1.1 Nguyên tắc cơ bản 12](#_Toc154266359)

[2.1.2 Thách thức chính: Hiện tượng "Quên" và giải pháp 13](#_Toc154266360)

[2.2 Test production 14](#_Toc154266361)

[2.2.1 Ý Nghĩa và Vai Trò của Test Production 14](#_Toc154266362)

[2.2.2 Quy trình xây dựng tập kiểm thử 14](#_Toc154266363)

[2.2.3 Cân Nhắc Khi Xây Dựng Tập Kiểm Thử Cân Đối 14](#_Toc154266364)

[TÀI LIỆU THAM KHẢO 16](#_Toc154266365)

# TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER

## Gradient Descent (GD)

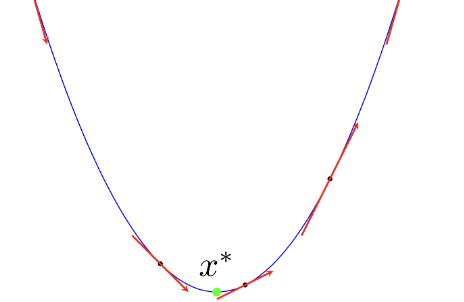
### Gradient Descent là gì?

Cho hàm số y = mx + b với m là trọng số (weight / ), b là bias ()

Gradient Descent là thuật toán sau n vòng lặp (epoch) sẽ cố gắng cập nhật các tham số của mô hình (weight và bias) sao cho giảm thiểu giá trị của hàm mất mát (loss function). Mỗi lần lặp, thuật toán sẽ tính gradient của hàm mất mát đối với các tham số, sau đó điều chỉnh các tham số theo hướng ngược của gradient để giảm giá trị của hàm mất mát (để đạo hàm gần bằng 0). Quá trình này lặp lại cho đến khi đạt được điều kiện dừng hoặc đạt đến một số vòng lặp cố định. Điều này giúp mô hình học từ dữ liệu và điều chỉnh các tham số để tối ưu hóa hiệu suất trên tập dữ liệu đào tạo

### Hiểu rõ về công thức của Gradient Descent

Để hiểu rõ hơn thì ta có hình vẽ sau:



Cách Gradient Descent hoạt động là chúng ta chọn ngẫu nhiên 1 điểm trên trục và sau n vòng lặp nó sẽ càng gần về cực tiểu. Để làm được điều đó thì ta có công thức của Gradient Descent như sau: hay x\_new=x – alpha\*

Vì chúng ta muốn sau mỗi lần cập nhật thì x của chúng ta càng gần cực tiểu nên cần trừ đi 1 đại lượng ngược dấu với đạo hàm và nhân với tốc độ học . Khi \* thì chúng ta sẽ có bước nhảy, rất quan trọng trong việc mô hình học nhanh hay chậm vì khi x đang ở xa cực tiểu thì chúng ta sẽ muốn bước nhảy xa và khi lại gần cực tiểu rồi thì bước nhảy bắt đầu nhỏ dần. Nếu chỉ có mà không có thì bước nhảy sẽ rất chậm. Thay vì chỉ 20 bước là ta tiến tới cực tiểu thì sẽ gấp nhiều lần ta mới đến được, như thế sẽ không tối ưu về tốc độ học

### Tính chất của Gradient DescentA graph of a function Description automatically generatedA graph of a function Description automatically generatedA graph of a function Description automatically generated

Qua hình trên ta có thể thấy rằng Gradient Descent phụ thuộc vào nhiều yếu tố như chọn điểm x ban đầu hoặc learning rate khác nhau sẽ ảnh hưởng đến quá trình hội tụ.

## Stochastic Gradient Descent (SGD)

Việc tính toán gradients sẽ tốn kém nếu ta có kích thước dữ liệu lớn vì cứ mỗi vòng lặp lại chạy trên tất cả điểm dữ liệu. Để giải quyết vấn đề này thì chúng ta có Stochastic Gradient Descent, thuật toán này sẽ chọn số điểm dữ liệu ngẫu nhiên từ tập dữ liệu đào tạo.

w := w - n(w)

Cũng giống như GD thông thương, chúng ta sẽ khởi tạo tham số w và learning rate ban đầu. Nhưng cái khác ở chỗ là sẽ xáo trộn ngẫu nhiên dữ liệu ở mỗi lần lặp để đạt mức tối thiểu gần đúng (approximate minimum) -> đường đi sẽ không được mịn như GD -> SGD sử dụng số lần lặp cao hơn để đạt đến cục bộ. Nhưng ngay cả sau khi tăng số lần lặp, chi phí tính toán vẫn thấp hơn so với GD.

* Nếu dữ liệu rất lớn và thời gian tính toán là một yếu tố thiết yếu, thì việc SGD nên được ưu tiên hơn thuật toán GD

A diagram of a solar system

Description automatically generated

## Momentum

### 1.3.1 Tổng quan

Vì SGD có số lần lặp nhiều hơn nên thời gian chạy sẽ khá lâu, để giải quyết vấn đề đó thì chúng ta có thuật toán Momentum.

Momentum là một kỹ thuật giúp thuật toán hội tụ nhanh hơn đối với hàm mất mát (loss function).

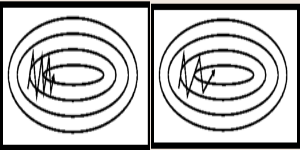
Trong SGD, đôi khi thuật toán dao động giữa các hướng của độ dốc và cập nhật trọng số tương ứng. Momentum được thiết kế để giảm thiểu hiện tượng này.

### 1.3.2 Nguyên tắc hoạt động

Momentum hoạt động bằng cách thêm một phần của cập nhật trọng số trước đó vào cập nhật trọng số hiện tại

Điều này tạo ra một "đà" cho quá trình cập nhật, giúp thuật toán vượt qua những điểm low points (đáy) và tăng tốc khi tiến gần tới global minimum.

\*Lưu ý: Khi training bằng thuật toán này thì learning rate phải được điều chỉnh learning rate một cách thích hợp. Thông thường, learning rate có thể được giảm xuống khi momentum được sử dụng để tránh việc đi quá xa và bất thường.



Từ hình trên, ở bên trái là quá trình hội tụ của SGD, bên phải là Momentum. Ta có thể thấy rằng Momentum giúp ta hội tụ nhanh hơn.

## Adagrad (Adaptive Gradient Descent)

Đây là thuật toán giúp điều chỉnh tốc độ học cho từng tham số

### Tại sao cần điều chỉnh tốc độ học?

Dữ liệu thực tế thường chứa các đặc trưng với mức độ ảnh hưởng khác nhau đối với kết quả dự đoán. Có những đặc trưng quan trọng hơn và có ảnh hưởng lớn hơn đối với kết quả cuối cùng.

Tối ưu hóa tỷ lệ học cho từng tham số có thể giúp tận dụng tốt nhất khả năng học của mỗi tham số, giúp tối ưu hóa mô hình một cách hiệu quả hơn.

* Việc điều chỉnh tỷ lệ học cho từng tham số giúp mô hình học một cách linh hoạt và có thể dẫn đến sự hội tụ nhanh chóng và hiệu quả hơn.

### Cơ chế hoạt động của Adagrad

Adagrad điều chỉnh tỷ lệ học (learning rate) của từng tham số dựa trên lịch sử của gradient của tham số đó. Cụ thể, mỗi tham số có một "accumulator" (bộ tích lũy) cho tổng bình phương của các gradient đã thấy trước đó. Tỷ lệ học của mỗi tham số được chia cho căn bậc hai của accumulator, giúp tỷ lệ học giảm đi khi gradient lớn và tăng lên khi gradient nhỏ.

Công thức: A square root of a mathematical equation

Description automatically generated

Trong đó:

* là giá trị mới của tham số sau bước cập nhật
* là giá trị hiện tại của tham số
* là tốc độ học (learning rate)
* *J*() là gradient của hàm mất mát theo tham số tại điểm
* là ma trận đường chéo của các bình phương của gradient tích lũy cho mỗi tham số từ bước đầu tiên đến bước hiện tại.
* là một giá trị nhỏ (ví dụ: 1x10^-8) được thêm vào dưới mẫu để tránh trường hợp chia cho 0.

Adagrad tỏ ra hiệu quả trong việc xử lý dữ liệu có các đặc trưng với mức độ quan trọng khác nhau, nhưng có thể bị giới hạn bởi vấn đề "vanishing learning rate" khi accumulator ngày càng tăng lớn.

### Ưu và nhược điểm của Adagrad

Ưu điểm:

* Loại bỏ nhu cầu phải điều chỉnh tỷ lệ học một cách thủ công, giúp giảm bớt công đoạn cần phải tinh chỉnh siêu tham số.
* Adagrad được coi là đáng tin cậy hơn so với các thuật toán Gradient Descent và các biến thể của chúng. Nó có khả năng đạt đến điểm hội tụ nhanh hơn trong một số trường hợp.

Nhược điểm:

* Nó giảm tỷ lệ học một cách mạnh mẽ và liên tục. Điều này có thể dẫn đến việc tỷ lệ học trở nên cực kỳ nhỏ tới mức không thể đáng kể do việc bình phương gradient tích lũy trong mẫu ngày càng tăng lớn, làm cho phần mẫu trở nên rất lớn
* Với tỷ lệ học quá nhỏ, có thể dẫn đến mô hình không còn có khả năng học thêm nhiều kiến thức mới.

## RMS Prop (Root Mean Square)

Đây là thuật toán điều chỉnh độ lớn của bước học dựa trên một trung bình có trọng số của bình phương của gradient, được thiết kế để giảm độ chậm trễ của Adagrad bằng cách giữ cho độ lớn của gradient không thay đổi quá nhanh dựa trên thông tin từ toàn bộ lịch sử gradient.

Công thức: A black square with a white background

Description automatically generated with medium confidence

Trong đó E[ là trung bình có trọng số của bình phương của gradient

Ưu điểm:

+ AdaGrad giảm nhu cầu phải điều chỉnh tỷ lệ học một cách thủ công. Tính tự điều chỉnh của nó giúp loại bỏ bước này, giảm bớt công sức của người lập trình.

+ Được coi là đáng tin cậy hơn so với các thuật toán gradient descent thông thường và các biến thể của chúng.

+ Đạt được sự hội tụ nhanh chóng hơn, đặc biệt là ở các điểm thực tế với dữ liệu có đặc tính đa dạng.

Nhược điểm:  
+ Một vấn đề của của RMSProp là nó không hoạt động tốt trên dữ liệu lớn mà cần cập nhật theo từng mini-batch thay vì toàn bộ tập dữ liệu.

+ AdaGrad có xu hướng giảm tỷ lệ học một cách quá mạnh và đơn điệu. Điều này dẫn đến việc tỷ lệ học trở nên cực kỳ nhỏ ở một số điểm, khiến mô hình không thể học thêm kiến thức mới, ảnh hưởng đến độ chính xác của mô hình.

## AdaDelta

### Tính chất chung

AdaDelta được coi là một phiên bản cải tiến và mạnh mẽ hơn so với AdaGrad optimizer.

Được thiết kế để giải quyết nhược điểm quan trọng của AdaGrad và RMSprop optimizer.

### Vấn đề chính với AdaGrad và RMSprop

Tỷ lệ học khởi tạo: Cả AdaGrad và RMSprop đều đòi hỏi người lập trình phải xác định tỷ lệ học ban đầu.

Tình trạng suy giảm tỷ lệ học: Tính chất này khiến tỷ lệ học giảm đến gần 0 ở một số điểm sau một số lượng lớn các iterations, làm cho mô hình không thể học thêm kiến thức mới.

### Giải pháp của AdaDelta

AdaDelta sử dụng hai biến trạng thái để lưu trữ trung bình rò rỉ (leaky average) của độ lớn gradient bậc hai và trung bình rò rỉ của độ lớn của sự thay đổi của các tham số trong mô hình.

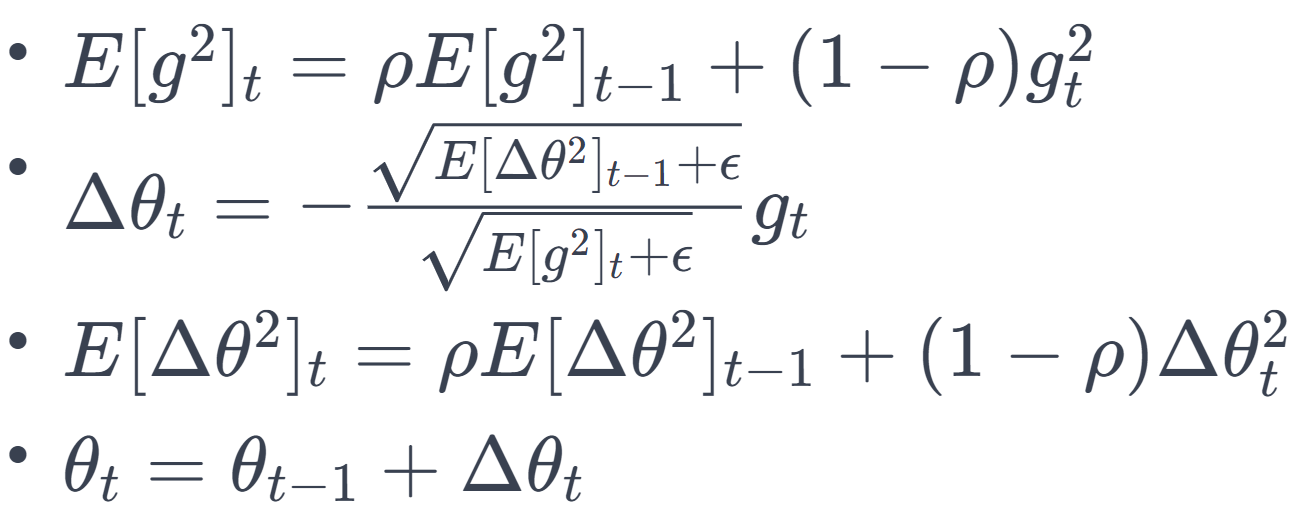
Thay vì yêu cầu tỷ lệ học khởi tạo, AdaDelta tự động điều chỉnh tỷ lệ học dựa trên các thay đổi trong các tham số. Điều này làm cho quá trình huấn luyện trở nên linh hoạt và không phụ thuộc vào việc chọn lựa tỷ lệ học.

### Công thức cập nhật trong AdaDelta

Các biến trạng thái chính:

1. E[: Trung bình rò rỉ (leaky average) của bình phương gradient.
2. E[]: Trung bình rò rỉ của bình phương của sự thay đổi của tham số.

Công thức cập nhật các tham số:



Trong đó:

* là gradient tại thời điểm t
* là thay đổi của tham số tại thời điểm t
* là hệ số rò rỉ (leakage factor) quyết định mức độ giữ lại thông tin từ các bước trước đó.

Công thức này giúp điều chỉnh tỷ lệ học tự động dựa trên các biến trạng thái của gradient và sự thay đổi của tham số

### Ưu và nhược điểm của AdaDelta

Ưu điểm:

+ Khắc phục vấn đề của AdaGrad và RMSprop bằng cách tự động điều chỉnh tỷ lệ học.

+ Không yêu cầu xác định tỷ lệ học khởi tạo.

Nhược điểm:

+ Cũng có thể bị vấn đề của việc giảm tỷ lệ học quá mạnh, nhưng nó được giảm thiểu hơn so với các phiên bản trước đó.

## Adam

Viết tắt của Adaptive Moment Estimation, là 1 phần mở rộng của thuật toán stochastic gradient descent (SGD). Nó còn kết hợp những đặc điểm tích cực của hai thuật toán khác là AdaGrad và RMSProp để cung cấp cập nhật hiệu quả và điều chỉnh tốt đối với trọng số trong quá trình training.

Các bước cập nhật trọng số trong thuật toán Adam:  
+ Bước 1: Tính gradient và bình phương gradient:

* Gradient cho từng trọng số
* Bình phương gradient cho từng trọng số

+ Bước 2: Cập nhật moment bậc nhất (First Moment Estimate):

* Momentum bậc nhất (momentum giống với RMSProp):

= .+(1 - )

+ Bước 3: Cập nhật moment bậc hai (Second Moment Estimate):

= .+(1 - )

+ Bước 4: Điều chỉnh moment bậc nhất và moment bậc hai:

* Moment bậc nhất được điều chỉnh để giảm tác động của mất mát lớn ban đầu:

=

* Moment bậc hai được điều chỉnh để giảm tác động của mất mát lớn ban đầu:

=

+ Bước 5: Cập nhật trọng số:

= - .

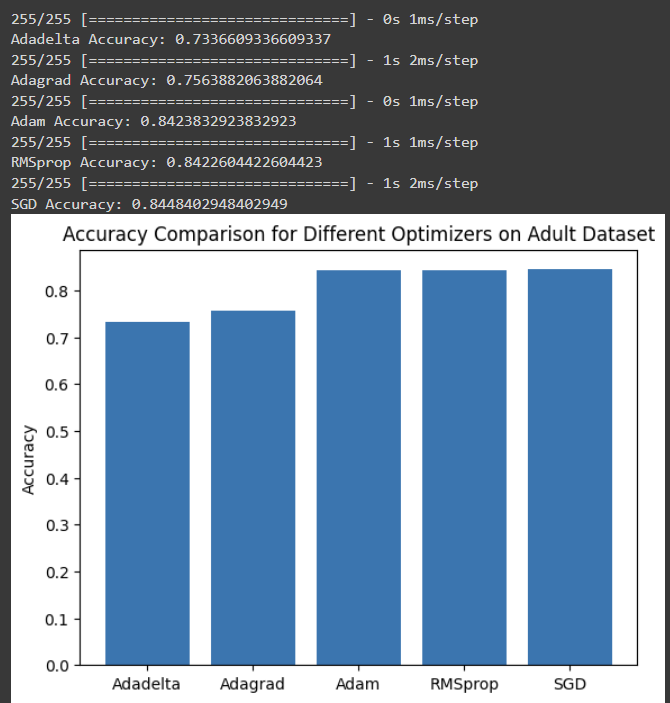
Trong đó:

* là trọng số tại bước thời gian t
* là gradient tại bước thời gian t
* là moment bậc nhất (momentum) tại bước thời gian t
* là moment bậc hai tại bước thời gian t
* là learning rate.
* và là các hệ số đánh giá tầm quan trọng của moment bậc nhất và moment bậc hai, thường được đặt là 0.9 và 0.999.

Adam là thuật toán được sử dụng phổ biến nhất hiện nay, nó có xu hướng tập trung vào thời gian tính toán nhanh. Nhưng đôi khi các thuật toán như SGD tổng quát hóa dữ liệu một cách tốt hơn với chi phí là tốc độ tính toán thấp.

* Các thuật toán tối ưu hóa có thể được chọn tùy thuộc vào yêu cầu và loại dữ liệu cụ thể.

## So sánh các thuật toán qua adult dataset



# CONTINUAL LEARNING VÀ TEST PRODUCT

## Continual learning

Hay còn gọi là Lifelong Learning, là một mô hình học máy mà mô hình được liên tục cập nhật và học thêm từ dữ liệu mới mà nó gặp phải, mà không cần phải huấn luyện lại từ đầu.

### Nguyên tắc cơ bản

Quên có ý thức (Conscious Forgetting):

* Mô hình cần có khả năng quên một số kiến thức cũ khi học kiến thức mới.
* Điều này giúp mô hình duy trì khả năng học tốt hơn trên dữ liệu mới và giảm thiểu hiện tượng "interference" (can thiệp) giữa kiến thức mới và cũ.

Ứng phó với dữ liệu không đồng nhất:

* Continual learning phải đối mặt với việc học từ các tập dữ liệu không đồng nhất, thay đổi theo thời gian.
* Mô hình cần thích ứng và học từ dữ liệu mới mà không làm suy giảm khả năng dự đoán trên dữ liệu cũ.

Phân đoạn (Segmentation):

* Dữ liệu được chia thành các phân đoạn (hoặc nhiệm vụ) khác nhau và mô hình học từng phân đoạn một cách riêng lẻ.
* Điều này giúp giảm đối mặt với vấn đề "cái gì học sau đè lên cái gì học trước" bằng cách tạo ra một cơ chế chia sẻ thông tin linh hoạt giữa các phân đoạn.

Regularization và Stabilization:

* Sử dụng các phương pháp regularization như Elastic Weight Consolidation (EWC) để bảo vệ trọng số của các phần quan trọng của mô hình, ngăn chúng bị thay đổi quá mức khi học từ dữ liệu mới.
* Stabilization giúp duy trì ổn định khi áp dụng continual learning, ngăn mô hình bị biến động quá mức.

Kiểm soát hiệu suất (Performance Monitoring):

* Liên tục đánh giá hiệu suất của mô hình trên dữ liệu mới và cũ.
* Nếu có sự suy giảm độ chính xác trên dữ liệu cũ, cần áp dụng các biện pháp để đảm bảo rằng mô hình vẫn giữ được khả năng dự đoán tốt.

### Thách thức chính: Hiện tượng "Quên" và giải pháp

Hiện tượng "quên" là mô hình học kiến thức mới nhưng đồng thời làm suy giảm hiệu suất trên kiến thức cũ. Dưới đây là một số giải pháp để đối mặt với vấn đề này:

Regularization:

* Sử dụng các kỹ thuật regularization như Elastic Weight Consolidation (EWC) để bảo vệ trọng số của các phần quan trọng của mô hình.
* Regularization giúp giảm sự thay đổi của trọng số liên quan đến kiến thức cũ khi học từ dữ liệu mới.

Rehearsal (Tập luyện lại):

* Làm cho mô hình "nhớ" lại một số mẫu từ dữ liệu cũ khi học dữ liệu mới.
* Rehearsal giúp duy trì thông tin quan trọng từ dữ liệu cũ và giảm khả năng quên.

Ephemeral Memorization:

* Thiết kế mô hình để tạm thời ghi nhớ thông tin mới trước khi nó quên thông tin cũ.
* Điều này có thể bao gồm việc sử dụng bộ nhớ ngắn hạn hoặc cơ chế giữa thời kỳ "ghi nhớ" và "quên."

Phân đoạn và Chia sẻ thông tin:

* Chia dữ liệu thành các phân đoạn và cho phép mô hình học mỗi phân đoạn một cách riêng lẻ.
* Chia sẻ thông tin linh hoạt giữa các phân đoạn để duy trì sự liên kết giữa các kiến thức cũ và mới.

Transfer Learning:

* Sử dụng transfer learning để chuyển giao kiến thức từ một mô hình đã được huấn luyện trước đó.
* Mô hình sẽ bắt đầu với kiến thức cũ và sau đó cập nhật thông tin khi học từ dữ liệu mới.

Kích thích Môi trường Sống (Lifelong Learning Scenario):

* Tạo ra môi trường mà mô hình liên tục học và gặp phải dữ liệu mới.
* Sử dụng các chiến lược như online learning để mô hình có thể học liên tục mà không cần quá nhiều giữa các chu kỳ học.

## Test production

### Ý Nghĩa và Vai Trò của Test Production

Đánh Giá Hiệu Suất:

* Mục đích chính của Test Production là đánh giá hiệu suất của mô hình học máy trên dữ liệu mới và cũ.
* Cung cấp thông tin về khả năng tổng quan của mô hình trong môi trường thực tế.

Đảm Bảo Độ Tin Cậy:

* Hỗ trợ xác nhận rằng mô hình không bị suy giảm hiệu suất theo thời gian hoặc trên các tập dữ liệu mới.
* Duy trì độ tin cậy và khả năng dự đoán của mô hình trong các tình huống thực tế.

### Quy trình xây dựng tập kiểm thử

Lựa Chọn Mẫu Kiểm Thử Đại Diện:

* Chọn mẫu kiểm thử có tính đại diện cao, phản ánh đa dạng của dữ liệu mà mô hình có thể gặp phải trong thực tế.
* Bao gồm các trường hợp biên, tình huống khó khăn và đặc điểm đặc biệt của dữ liệu.

Xây Dựng Tập Kiểm Thử Đa Dạng:

* Đảm bảo tập kiểm thử bao gồm nhiều loại dữ liệu và tình huống khác nhau để kiểm tra khả năng tổng quan của mô hình.
* Bao gồm cả dữ liệu mới và dữ liệu cũ để đánh giá hiệu suất trên cả hai mặt.

### Cân Nhắc Khi Xây Dựng Tập Kiểm Thử Cân Đối

Đối Với Dữ Liệu Imbalanced:

* Giải quyết vấn đề chênh lệch về số lượng mẫu giữa các lớp bằng cách sử dụng kỹ thuật lựa chọn mẫu thông minh.
* Đảm bảo rằng mô hình được đánh giá công bằng trên cả các lớp thiểu số.

Sự Đa Dạng và Đối Ngẫu:

* Kết hợp cả dữ liệu tích cực và tiêu cực để tạo ra một tập kiểm thử cân đối về tính đối ngẫu.
* Bảo đảm rằng mô hình không chỉ hiệu quả trên trường hợp phổ biến mà còn trên các trường hợp đặc biệt.

Kiểm Thử Định Kỳ:

* Thực hiện kiểm thử định kỳ để đánh giá hiệu suất của mô hình qua thời gian và trên các tập dữ liệu thay đổi.
* Cập nhật tập kiểm thử theo cách định kỳ để đảm bảo sự đại diện liên tục.

# TÀI LIỆU THAM KHẢO

Tiếng Việt

<https://machinelearningcoban.com/2017/01/12/gradientdescent/>

<https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8>

https://www.youtube.com/watch?v=pxQX5f\_iHIA&ab\_channel=Ph%E1%BA%A1mThanhScience%26Tech

Tiếng Anh

<https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#Gradient_Descent_Deep_Learning_Optimizer>

<https://link.springer.com/article/10.1007/s10845-021-01793-0\>