TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**BÁO CÁO CUỐI KÌ**

**MÔN HỌC: NHẬP MÔN HỌC MÁY**

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

*Người thực hiện*: **NGUYỄN TƯỜNG AN - 52100380**

Lớp **: 21050401**

Khoá  **: 25**

**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**

A red and blue logo

Description automatically generated

**BÁO CÁO CUỐI KÌ**

**MÔN HỌC: NHẬP MÔN HỌC MÁY**

**Mã môn học: 503044**

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

*Người thực hiện*: **NGUYỄN TƯỜNG AN - 52100380**

Lớp **: 21050401**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Môn học Nhập môn Học máy thực sự rất bổ ích, đã bổ trợ thêm cho em rất nhiều kiến thức liên quan đến chuyên ngành của mình. Em xin cảm ơn sự quan tâm của nhà trường và khoa CNTT khi đã đưa bộ môn vào giảng dạy. Đặc biệt em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn – thầy PGS.TS. Lê Anh Cường đã luôn ân cần giảng dạy, giải đáp tận tình những thắc mắc của chúng em, giúp chúng em có thể hiểu và làm bài một cách đơn giản nhất.

Bài báo cáo này là minh chứng rõ nhất cho những cố gắng, nỗ lực học tập của em. Tuy vậy sự tiếp thu kiến thức của em chỉ mới ở mức cơ bản và còn nhiều thiếu sót, kính mong thầy có thể cho em thêm nhận xét và góp ý để bài báo cáo có thể hoàn thiện hơn. Em xin chân thành cảm ơn!

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của thầy 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 đồ 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 đồ 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*

*(ký tên và ghi rõ họ tên)*

*An*

*Nguyễn Tường An*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Đối với nội dung tìm hiểu cá nhân trong bài một, em tiến hành tìm hiểu, nghiên cứu và đưa ra đánh giá, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy. Ngoài ra, cũng tiến hành tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

**MỤC LỤC**

[LỜI CẢM ƠN 3](#_Toc154259180)

[TÓM TẮT 6](#_Toc154259181)

[1.1. Tổng quan về optimizer: 8](#_Toc154259182)

[1.1.1 Định nghĩa Optimizer là gì: 8](#_Toc154259183)

[1.1.2 Tại sao phải dùng nó ? 8](#_Toc154259184)

[1.2. Các loại Optimizer phổ biến 9](#_Toc154259185)

[1.2.1 Gradient Descent 9](#_Toc154259186)

[1.2.2 Adaptive Gradient Algorithm (Adagrad): 13](#_Toc154259187)

[1.2.3 Root Mean Square Propagation (RMSprop) 14](#_Toc154259188)

[1.2.4 Adaptive Moment Estimation (Adam): 15](#_Toc154259189)

[1.2.5 Nesterov-accelerated Adaptive Moment Estimation (Nadam) 16](#_Toc154259190)

[1.2.6 Adadelta 17](#_Toc154259191)

[1.3. So sánh các phương pháp Optimizer 18](#_Toc154259192)

[2.1. Continual learning: 21](#_Toc154259193)

[2.1.1 Tổng quan 21](#_Toc154259194)

[2.1.2 Mục tiêu 22](#_Toc154259195)

[2.1.3 Các Loại mô hình Continual Learning: 22](#_Toc154259196)

[2.1.4 Thách thức của Continual Learning: 23](#_Toc154259197)

[2.1.5 Khi nào thì nên cập nhật lại mô hình? 24](#_Toc154259198)

[2.1.6 Ưu và nhược điểm Continual Learning 24](#_Toc154259199)

[2.2. Test Production: 25](#_Toc154259200)

[2.2.1 Tổng quan 25](#_Toc154259201)

[2.2.2 Mục đích 26](#_Toc154259202)

[2.2.3 Lợi ích chính của Machine Learning trong kiểm thử phần mềm 26](#_Toc154259203)

[2.2.4 Ứng dụng Machine Learning trong kiểm thử phần mềm 27](#_Toc154259204)

**DANH MỤC HÌNH ẢNH**

[Hình 1. 1 Optimizer 9](#_Toc154262495)

[Hình 1. 2 Đường cong fx 11](#_Toc154262496)

[Hình 1. 3 Tiếp tuyến 12](#_Toc154262497)

[Hình 1. 4 Number of iterations 19](#_Toc154262498)

[Hình 2. 1 Continual Learning Model 23](#_Toc154262503)

1. CÁC PHƯƠNG PHÁP OPTIMIZER

## Tổng quan về optimizer:

### Định nghĩa Optimizer là gì:

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.

Bài toán tối ưu dạng tổng quát:

Thoả mãn 2 điều kiện:

Trong đó:

là biến tối ưu (Optimizer variable)

là hàm mục tiêu (Objective/cost/lost function)

ràng buộc dạng bất đẳng thức

ràng buộc dạng đẳng thức

Theo quy ước,dạng tiêu chuẩn xác định một bài toán cực tiểu hoá. Bài toán cực đại hóa có thể được giải bằng cách phủ định hàm mục tiêu.

### Tại sao phải dùng nó ?

* Tìm kiếm hiệu quả: Thay vì tìm kiếm ngẫu nhiên trong không gian trọng số, thuật toán tối ưu hóa giúp chúng ta tìm kiếm một cách có hệ thống và hiệu quả.
* Hội tụ nhanh: Các thuật toán tối ưu hóa như Gradient Descent, Stochastic Gradient Descent (SGD), Adam, và nhiều thuật toán khác giúp mô hình hội tụ nhanh hơn.
* Khả năng tổng quát hóa cao: Một số thuật toán tối ưu hóa như SGD có khả năng tổng quát hóa cao, có nghĩa là chúng có thể nhận dạng được những mẫu chưa từng được huấn luyện.
* Độ chính xác cao: Các thuật toán tối ưu hóa giúp tăng độ chính xác của mô hình bằng cách điều chỉnh trọng số và sai số.
* Tránh rơi vào cực tiểu địa phương: Một số thuật toán tối ưu hóa như Adam có thể giúp mô hình tránh rơi vào các cực tiểu địa phương.

## Các loại Optimizer phổ biến

A diagram of a diagram

Description automatically generated

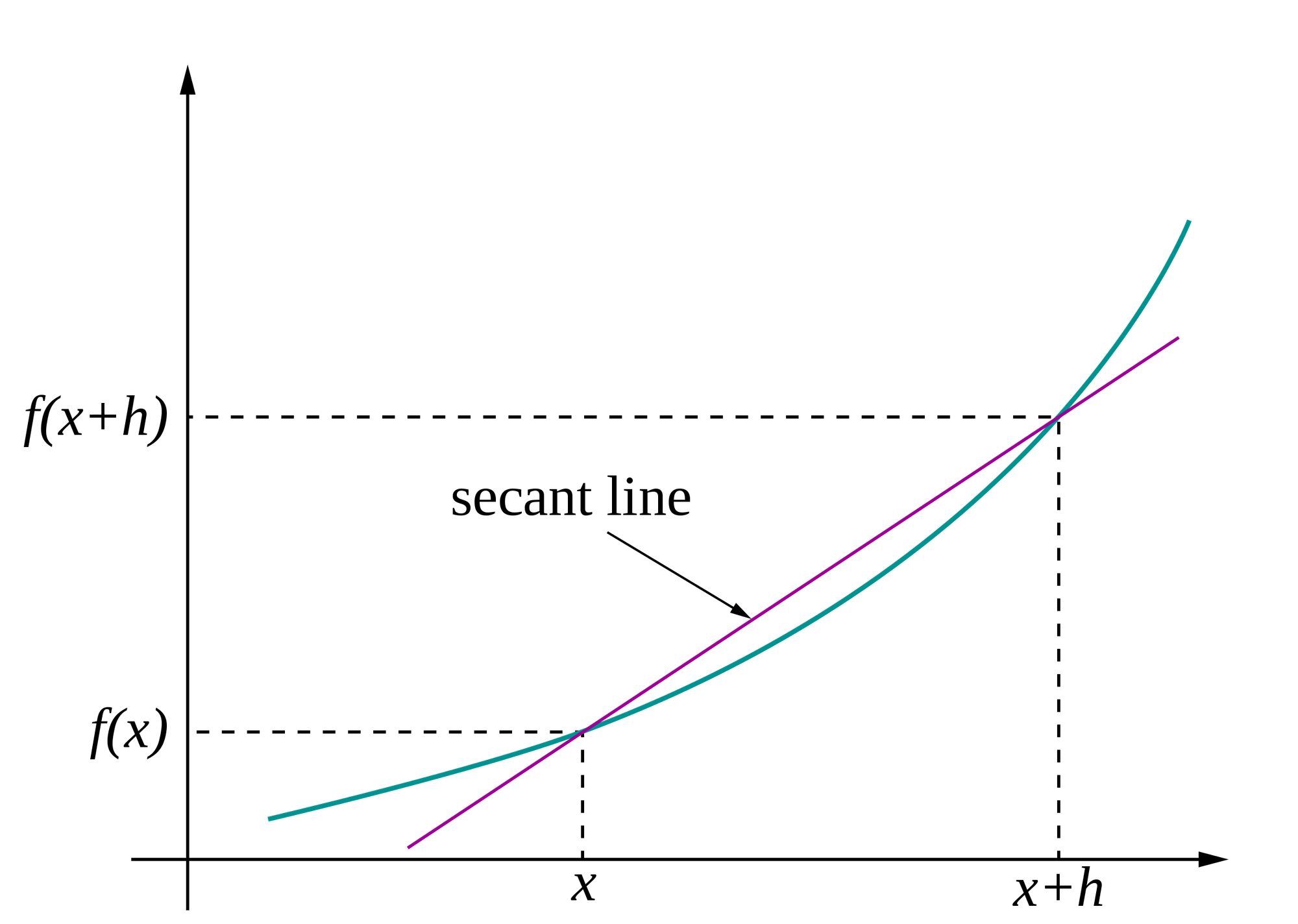
Hình 1. 1 Optimizer

* Gradient Descent
* Stochastic Gradient Descent
* Momentum
* Adam
* RMSprop
* **Adagrad**

### ****Gradient Descent****

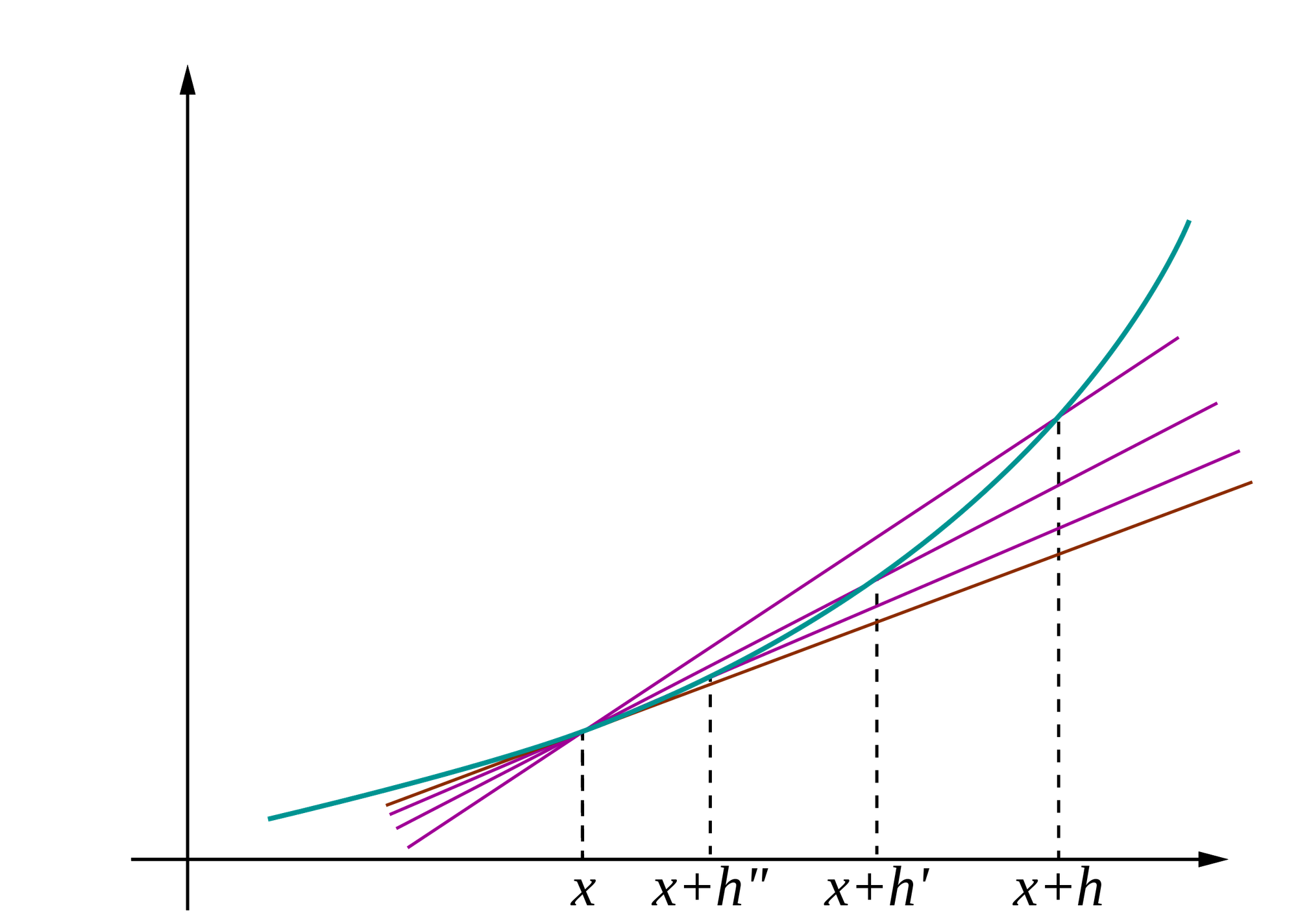
Xét một hàm số, ***đạo hàm của một hàm số*** là ***sự mô tả độ biến thiên*** (mức độ thay đổi giá trị) của hàm số đó.Ký hiệu đạo hàm của hàm số f(x) theo biến x là

Đạo hàm của một hàm số tại một điểm là sự mô tả độ biến thiên của hàm số tại điểm đó.

****

Hình 1. 2 Đường cong fx

Cát tuyến của đường cong y = f(x) tại hai điểm (x, f(x)) và (x+h, f(x+h))

****

Hình 1. 3 Tiếp tuyến

Khi h tiến dần về 0 thì cát tuyến sẽ tiến dần đến tiếp tuyến của đồ thị

**Stochastic GD**

Công thức cập nhật là:

Trong đó, α là tốc độ học, ∇f(x) là gradient của hàm mất mát tại điểm x. GD có hai biến thể là **Batch GD** và **Stochastic GD**. Batch GD dùng toàn bộ dữ liệu để cập nhật một lần, nên đảm bảo hội tụ chính xác nhưng tốn nhiều thời gian và bộ nhớ. [Stochastic GD dùng một điểm dữ liệu ngẫu nhiên để cập nhật mỗi lần, nên nhanh hơn nhưng không ổn định và có thể bị rơi vào điểm cực tiểu cục bộ](https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8).

**Momentum**

- Các biến thể của Gradient Descent giải quyết được vấn đề của chính bản thân nó mang lại như: có thể hoạt động tốt trên tập dữ liệu (bài toán) lớn.

- Dễ tối ưu hóa và hiệu chỉnh: Các biến thể của Gradient Descent có thể điều chỉnh siêu tham số (hyper parameter) để tối ưu hóa hiệu suất.

Nhược điểm:

- Phụ thuộc vào nghiệm khởi tạo và learning rate.

- Dễ rơi vào Local minimum, vì bài toán yêu cầu là tìm Global minimum.

- Bài toán càng lớn và phức tạp thì chi phí sẽ càng cao hơn so với các mô hình tối ưu khác.

- Không thực hiện được đối với bài toán không tính được đạo hàm.

Đặc biệt: **Momentum** là một kỹ thuật được sử dụng để giảm độ dao động (oscillation) và giúp quá trình hội tụ nhanh hơn trong thuật toán tối ưu hóa, đặc biệt là trong các phương pháp như Gradient Descent. Momentum giữ thông tin về hướng trước đó của các bước cập nhật và sử dụng nó để "giữ đà" cho quá trình cập nhật tiếp theo. Điều này giúp giảm bớt dao động và giúp thuật toán vượt qua các điểm nhỏ (local minima) trên đồ thị hàm mất mát. Hệ số momentum giúp giảm độ dao động của quá trình cập nhật, đặc biệt hiệu quả khi quá trình huấn luyện đi qua các vùng cong và lồi lõm. Các giá trị momentum từ các bước cập nhật trước giúp tăng tốc độ và ổn định cho bước cập nhật tiếp theo.

**Nguyên lí hoạt động**

Momentum hoạt động bằng cách tích lũy 1 vector vận tốc để giữ cho quá trình cập nhật trọng số được hướng về 1 hướng để có thể giải quyết vấn đề bị mắc kẹt ở cực tiểu local.

Công thức:



Trong đó:

* vt là vận tốc tại t
* là vector trọng số tại t
* là learning rate
* là hàm mất mát.

**Ưu điểm** của momentum: giải quyết được vấn đề gradient descen không tiến tới được điểm global minimum

**Nhược điểm** của momentum: Mất khá nhiều thời gian và tài nguyên để có thể tìm đến global minimum.

### Adaptive Gradient Algorithm (Adagrad):

Adaptive Gradient Algorithm là thuật toán cập nhật trọng số của mô hình dựa trên tần suất xuất hiện của các đặc trưng. Thuật toán này đặc trưng bởi việc sử dụng tốc độ học (learning rate) được điều chỉnh tự động cho từng tham số. Điều này là do nó sử dụng các tốc độ học khác nhau cho mỗi lần lặp. Sự thay đổi trong tốc độ học phụ thuộc vào sự khác biệt giữa các tham số trong quá trình huấn luyện. Càng nhiều tham số thay đổi, thì thay đổi tốc độ học càng nhỏ. Sửa đổi này rất hữu ích vì dữ liệu thực tế chứa các đặc trưng thưa thớt cũng như đặc trưng dày đặc. Do đó, việc có cùng một giá trị tốc độ học cho tất cả các đặc trưng là không công bằng.

Công thức:

Trong đó:

n là tốc độ học

là gradient tại thời điểm t

là hệ số tránh lỗi (do chia cho mẫu bằng 0)

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: loại bỏ nhu cầu điều chỉnh tốc độ học thủ công. Nó đáng tin cậy hơn so với các thuật toán gradient descent và các biến thể của chúng, và đạt được sự hội tụ ở tốc độ cao.

Nhược điểm: nó giảm tốc độ học mạnh mẽ và liên tục. Có thể có một điểm mà tốc độ học trở nên cực kỳ nhỏ. Điều này là do đạo hàm bình phương trong mẫu số tiếp tục tích tụ, và do đó, phần mẫu tiếp tục tăng lên. Do tốc độ học nhỏ, mô hình cuối cùng trở nên không thể tiếp thu thêm kiến thức, và do đó, độ chính xác của mô hình bị đánh giá thấp.

### Root Mean Square Propagation (RMSprop)

RMSprop là một phương pháp tối ưu trong học máy, được dùng để giải quyết vấn đề tổng bình phương của Adagard bằng cách tính toán tổng bình phương theo thời gian. RMSprop giúp tối ưu và thích ứng tốt hơn với độ lệch và tỷ lệ khác nhau giữa các tham số.

Công thức

* Công thức cập nhật tổng bình phương:



Trong đó:

*  là tổng bình phương của các gradient trước đó cho tham số thứ i
* là gradient của tham số thứ i tại t
* là hệ số giảm ảnh hưởng (decay rate)
* Công thức cập nhật trọng số:



Trong đó:

*  là giá trị của tham số thứ i tại t
*  là learning rate
*  là gradient của tham số thứ i tại t
*  là tổng bình phương của các gradient trước đó cho tham số thứ i
*  là tham số chống chia cho 0
* Ưu điểm: khả năng tự động điều chỉnh tốc độ học và giảm vấn đề của tốc độ học biến đổi đột ngột.
* Nhược điểm: có thể dẫn đến việc giảm tốc độ học quá nhanh, và mô hình có thể hội tụ chưa đủ khi đến gần điểm tối ưu.

### Adaptive Moment Estimation (Adam):

Adaptive Moment Estimation (Adam) là một thuật toán tối ưu phổ biến nhất trong các thuật toán tối ưu trong deep learning và machine learning. Thuật toán được xây dựng dựa trên ý tưởng kết hợp 2 thuật toán optimizer khác là: momentum và RMSprop để cùng một lúc “giữ đà” để vượt qua các local minimum (momentum) và thay đổi tốc độ học (learning rate) dựa trên độ lớn của gradient để tiến về global minimum nhanh hơn (RMSprop).

Công thức:

Trong đó:

là tập hợp các tham số (weights)

n là tỷ lệ học (learning rate)

là trung bình động của gradient theo bậc hai

là trung bình động của gradient

là hệ số tránh lỗi (do chia cho mẫu bằng 0)

Ưu điểm:

- Adam là một thuật toán tối ưu hóa hiệu quả và linh hoạt nhờ cơ chế tự động điều chỉnh learning rate dựa trên trung bình động của gradient, giúp nó hoạt động tốt trong các trường hợp gradient biến động.

- Adam hoạt động tốt với lượng dữ liệu lớn và không yêu cầu quá nhiều bộ nhớ để lưu trữ toàn bộ tập dữ liệu.

Nhược điểm:

- Trong một số trường hợp, Adam có thể dẫn đến over-optimization, đặc biệt với dữ liệu nhỏ hoặc khi mô hình quá phức tạp.

- Hiệu suất của Adam có thể phụ thuộc nhiều vào việc lựa chọn siêu tham số (hyper-parameter) như learning rate và beta.

- Một vài trường hợp cần nhiều bộ nhớ cho việc lưu trữ so với các thuật toán khác.

### Nesterov-accelerated Adaptive Moment Estimation (Nadam)

Nadam là một thuật toán tối ưu hóa được kết hợp bởi 2 thuật toán tối ưu hóa là: Nesterov Accelerated Gradient (NAG) và Adaptive Moment Estimation (Adam). Tên “Nadam” là sự kết hợp của “Nesterov” và “Adam”. Nadam có tất cả các ưu điểm mà 2 thuật toán cấu thành mang lại, thậm chí có phần cải tiến hơn.

Công thức:

Trong đó:

là gradient tại thời điểm t

và là các trung bình động tuyến tính và không gian tốc độ của gradient.

và là các hệ số giảm dần gradient.

là tỷ lệ học

là hệ số tránh lỗi (do chia cho mẫu bằng 0)

Ưu điểm:

- Hiệu suất tốt, hay vượt trội theo từng loại bài toán.

- Khả năng linh động trong tốc độ học

- Tiết kiệm tài nguyên, không yêu cầu nhiều về bộ nhớ

Nhược điểm: Công thức và nguyên tắc hoạt động của Nadam có thể rất phức tạp và khó hiểu so với người vừa bắt đầu học về các thuật toán optimizer.

### Adadelta

Adadelta là một biến thể khác của AdaGrad. Điểm khác biệt chính là Adadelta giảm mức độ mà tốc độ học sẽ thay đổi. Adadelta thường được biết đến là thuật toán không sử dụng tốc độ học vì nó dựa trên chính lượng thay đổi hiện tại để tính toán lượng thay đổi trong tương lai.

Các bước cập nhật của mô hình dựa trên các công thức:

- Tính gradient trung bình:

- Tính delta x trung bình:

- Tính tỷ lệ học mới:

- Cập nhật tham số:

Trong đó:

là gradient tại thời điểm t

là thay đổi của tham số tại thời điểm t

p là hệ số giảm dần gradient

là hệ số tránh lỗi (do chia cho mẫu bằng 0)

Ưu điểm:

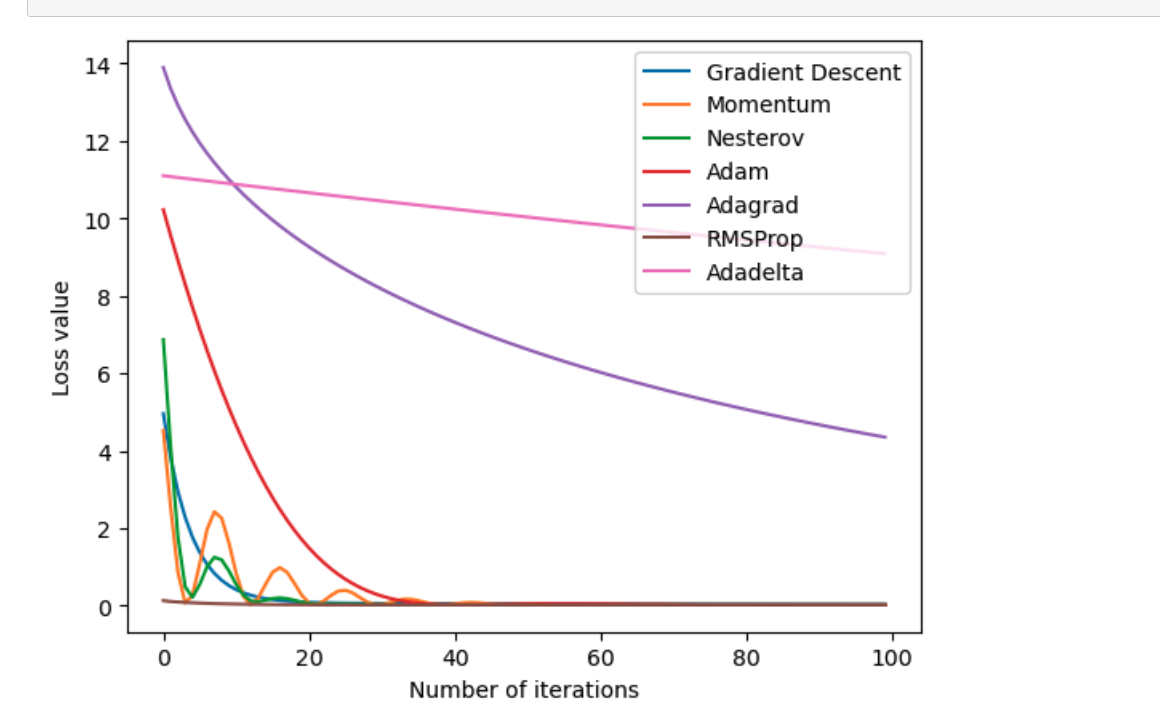
- Linh hoạt trong tốc độ học, giảm áp lực cho việc chọn siêu tham số

- Hiệu quả trên dữ liệu lớn, ít tiêu tốn tài nguyên hơn so với các thuật toán khác.

Nhược điểm:

- Khó hiểu và tính phức tạp cao hơn so với các thuật toán optimizer khác.

## So sánh các phương pháp Optimizer



Hình 1. 4 Number of iterations

So sánh

|  |  |  |
| --- | --- | --- |
| Thuật Toán | Ưu Điểm | Nhược Điểm |
| Nadam | - Kết hợp ưu điểm của Adam và Nesterov Accelerated Gradient (NAG). - Cải thiện khả năng hội tụ và chống overshooting so với Adam. | - Cần hiểu rõ các tham số để tránh hiện tượng quá mức hoặc mất khả năng hội tụ. |
| Adadelta | - Không yêu cầu sự điều chỉnh thủ công của tỷ lệ học (learning rate). - Hiệu quả khi xử lý các vấn đề với dữ liệu thưa và không yêu cầu việc lưu trữ toàn bộ lịch sử gradient. | - Cần điều chỉnh một số tham số như rho và epsilon, và không phù hợp cho mọi loại bài toán. |
| Adam | - Hiệu quả với dữ liệu lớn, không yêu cầu thực hiện các điều chỉnh thủ công của learning rate và thường là một lựa chọn an toàn. | - Cần hiểu rõ các tham số và có thể đôi khi không hội tụ đúng cách nếu không được cấu hình đúng. |
| RMSprop | - Hiệu quả với dữ liệu không đồng nhất, giảm ảnh hưởng của gradient nhỏ hoặc lớn. - Tích hợp khả năng tự giảm learning rate dựa trên gradient lịch sử. | - Cần điều chỉnh tham số learning rate và không phải lựa chọn tốt cho mọi bài toán. |
| Adagrad | - Tích hợp khả năng tự điều chỉnh learning rate dựa trên lịch sử gradient. - Hiệu quả với các bài toán có đặc điểm dữ liệu thưa. | - Tăng learning rate theo thời gian có thể làm giảm hiệu suất và không phù hợp với mọi bài toán. |

1. CONTINUAL LEARNING VÀ TEST PRODUCTION

## Continual learning:

### Tổng quan

Continual Learning (Học Liên Tục) là một phương pháp trong lĩnh vực học máy mà mô hình được liên tục cập nhật và mở rộng để học từ dữ liệu mới mà không quên kiến thức đã học trước đó. Điều này là quan trọng trong các hệ thống thời gian thực hoặc trong các ngữ cảnh mà dữ liệu thay đổi theo thời gian. Mục tiêu là giữ cho mô hình linh hoạt, có khả năng học từ những thay đổi liên tục mà không làm giảm hiệu suất trên dữ liệu cũ.

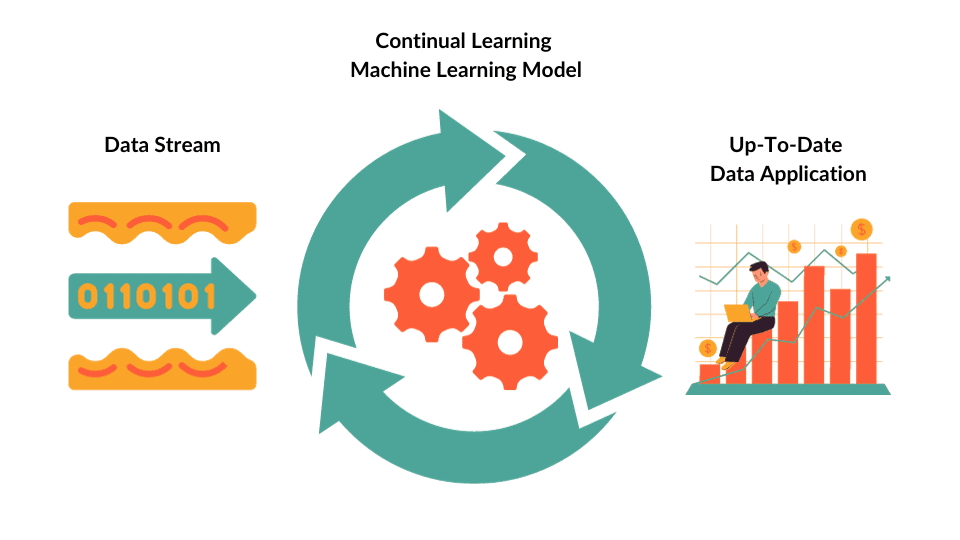
Continual learning (hay học tập liên tục) là ý tưởng cập nhật model (mô hình) của bạn khi có dữ liệu mới; điều này giúp mô hình của bạn theo kịp sự phân bổ dữ liệu hiện tại.

Sau khi mô hình của bạn được cập nhật, mô hình đó không thể được deploy (phát hành) một cách mù quáng vào sản xuất. Nó cần phải được thử nghiệm để đảm bảo rằng nó an toàn và tốt hơn so với mô hình hiện tại đang được sản xuất.

Continual learning thường bị hiểu sai:

- Continual learning KHÔNG đề cập đến một lớp thuật toán Machine Learning (ML) đặc biệt cho phép cập nhật mô hình dần dần khi có mọi điểm dữ liệu mới.

- Continual learning KHÔNG có nghĩa là bắt đầu công việc đào tạo lại mỗi khi có mẫu dữ liệu mới. Trên thực tế, điều này rất nguy hiểm vì nó khiến mạng lưới thần kinh dễ bị forgetting (lãng quên) một cách thảm họa.



Hình 2. 1 Continual Learning Model

### Mục tiêu

Lý do cơ bản là để giúp mô hình của bạn theo kịp sự thay đổi phân phối dữ liệu. Có một số trường hợp sử dụng trong đó việc thích ứng nhanh chóng với việc thay đổi phân phối là rất quan trọng.

Ví dụ: có thể có một buổi hòa nhạc ở một khu vực ngẫu nhiên vào Thứ Hai ngẫu nhiên và "mô hình Machine Learning định giá buổi hòa nhạc vào Thứ Hai" có thể không được trang bị tốt để xử lý nó.

### Các Loại mô hình Continual Learning:

Có nhiều loại Continual Learning để áp dụng cho mô hình. Các loại phổ biến bao gồm incremental learning, transfer learning và lifelong learning.

* Incremental learning: là một hình thức của continual learning trong đó mô hình được huấn luyện dần dần trên các dữ liệu mới mà không quên đi dữ liệu cũ. Các đặc điểm của phương pháp này:
* Khả năng mở rộng: Mô hình có thể mở rộng để train các dữ liệu mới mà không cần học lại từ đầu.
* Quản lý dữ liệu cũ: Đảm bảo việc không mất đi thông tin từ dữ liệu đã học bằng cách thông qua việc kết hợp dữ liệu mới và cũ.
* Tối ưu hóa: Cập nhật mô hình nhanh chóng khi có sự thay đổi giúp tối ưu hóa hiệu suất
* Transfer learning: Là một hình thức của continual learning mà mô hình được huấn luyện trước trên 1 tập dữ liệu lớn, sau đó dữ liệu đó được dùng để giải quyết 1 bài toán cụ thể hoặc 1 tập dữ liệu mới. Các đặc điểm của phương pháp này:
* Pre-training và Fine-tuning: Mô hình sẽ được huấn luyện trên 1 tập dữ liệu lớn trước khi giải quyết 1 bài toán cụ thể.
* Chuyển giao kiến thức: Kiến thức học từ một tập dữ liệu lớn có thể được sử dụng để giúp quá trình học trên dữ liệu mới.
* Hiệu suất cao: Khi được học trên dữ liệu mới có ít dữ liệu mô hình sẽ có hiệu quả cao hơn.
* Lifelong Learning: là một hình thức của Continual learning mà mô hình được huấn luyện trên nhiều tập dữ liệu khác nhau theo thời gian và phải giải quyết nhiều nhiệm vụ khác nhau. Các đặc điểm chính bao gồm:
* Học liên tục: Mô hình không chỉ học từ các dữ liệu mới mà còn giải quyết nhiều nhiệm vụ theo thời gian.
* Quản lý đa nhiệm: Có khả năng quản lý và kết từ các nhiệm vụ khác nhau.
* Chuyển giao thông tin: Thông tin từ mỗi nhiệm vụ có thể được sử dụng để hỗ trợ cho các nhiệm vụ mới.

### Thách thức của Continual Learning:

- Catastrophic Forgetting: Khi học một nhiệm vụ mới, mô hình có thể quên mất kiến thức đã học từ các nhiệm vụ trước đó.

- Fresh data access challenge(Thách thức truy cập dữ liệu mới): Tốc độ lưu trữ dữ liệu vào kho dữ liệu, Tốc độ ghi nhãn.

- Knowledge Transfer: Làm thế nào để chia sẻ và sử dụng lại kiến thức đã học từ một nhiệm vụ cho những nhiệm vụ mới.

- Representation Management: Làm thế nào để biểu diễn mô hình sao cho nó có thể linh hoạt và thích ứng được với nhiều loại dữ liệu.

- Evaluation Challenge (Thách thức đánh giá): việc học hỏi liên tục sẽ mở ra cơ hội cho các cuộc tấn công đối nghịch phối hợp nhằm đầu độc các mô hình.

- Data scaling challenge (Thách thức mở rộng quy mô dữ liệu): Việc mở rộng quy mô yêu cầu quyền truy cập vào số liệu thống kê dữ liệu toàn cầu như tối thiểu, tối đa, trung bình và phương sai. (min, max, average and variance.)

- Algorithm challenge (Thách thức thuật toán): Thách thức này xuất hiện khi bạn sử dụng một số loại thuật toán nhất định và muốn cập nhật chúng thật nhanh (ví dụ: mỗi giờ).

### Khi nào thì nên cập nhật lại mô hình?

Hầu hết các kiến thức tôi đã đề cập đều là liên quan đến việc cập nhật mô hình với dữ liệu mới (tức là lặp lại dữ liệu). Tuy nhiên, trong thực tế, đôi khi bạn cũng có thể cần thay đổi kiến trúc mô hình của mình (tức là lặp lại mô hình).Dưới đây là một số gợi ý về khi bạn nên và không nên xem xét việc lặp lại mô hình:

- Nếu bạn liên tục giảm ngưỡng kích hoạt tái đào tạo dữ liệu và bạn không đạt được nhiều hiệu suất, bạn có thể nên tìm kiếm một mô hình tốt hơn (nếu doanh nghiệp của bạn cần).

- Nếu việc chuyển sang một kiến trúc mô hình lớn hơn và yêu cầu 100 lần công suất tính toán mang lại 1% cải thiện hiệu suất, nhưng giảm ngưỡng kích hoạt tái đào tạo xuống 3 giờ cũng mang lại tăng hiệu suất 1% với 1 lần công suất tính toán, hãy ưu tiên lặp lại dữ liệu hơn là lặp lại mô hình.

- Câu hỏi "khi nào nên lặp lại mô hình so với lặp lại dữ liệu" vẫn chưa có câu trả lời tốt cho tất cả các nhiệm vụ. Bạn sẽ cần thực hiện các thử nghiệm trên nhiệm vụ cụ thể của mình để tìm ra lúc nào nên thực hiện lựa chọn nào

### Ưu và nhược điểm Continual Learning

+Ưu điểm của continual learning:

Continual learning có thể hữu ích với nhiều loại mô hình đặc biệt là trong các trường hợp liên quan tới dữ liệu thay đổi nhanh chóng. Sau đây là các ưu điểm so với phương pháp truyền thống:

* Tổng quát hóa: Mô hình dựa vào cả dữ liệu cũ và dữ liệu mới giúp tăng khả năng tổng quát hóa.
* Giữ lại thông tin: Mô hình sử dụng dữ liệu đã học trước đó giúp tích lũy thông tin theo thời gian. Việc này giúp mô hình không quên thông tin đã học và giúp huấn luyện tốt hơn với dữ liệu mới.
* Khả năng thích ứng: Mô hình sử dụng khả năng continual learning có khả năng thích ứng với dữ liệu mới, điều này giúp tăng khả năng dự đoán và linh hoạt.

Continual learning đem đến sự linh hoạt và khả năng thích ứng với mô hình học máy, đặc biệt với những bộ dữ liệu thay đổi liên tục.

+Nhược điểm của continual learning

Phương pháp continual learning mặc dù có rất nhiều ưu điểm nhưng cũng đi kèm 1 số nhược điểm như sau:

* Chi phí: Phương pháp continual learning phức tạp hơn về mặt tính toán với những phương pháp truyền thống vì mô hình cần liên tục điều chỉnh dựa trên dữ liệu mới.
* Quản lý mô hình: Mỗi khi dữ liệu được cập nhật thì mô hình mới được tạo ra. Vậy nên cần quản lý các mô hình 1 cách hiệu quả để tránh ảnh hưởng đến hiệu suất.
* Biến động trong dữ liệu: Đối với continual learning phải xử lý 1 lượng lớn dữ liệu, nên phương pháp này có rủi ro mất khả năng dự đoán khi các đặc trưng của dữ liệu thay đổi đột ngột.

## Test Production:

### Tổng quan

Test Production (Kiểm Thử Sản Xuất) là một giai đoạn quan trọng trong vòng đời của một mô hình học máy, chuyển từ giai đoạn phát triển và đào tạo sang triển khai trong các kịch bản thực tế. Trong quá trình Kiểm Thử Sản Xuất, mục tiêu chính là đánh giá hiệu suất của mô hình khi đối mặt với dữ liệu thực tế, thường là dữ liệu động từ môi trường mà nó được thiết kế để hoạt động.

### Mục đích

[**Tự tin trong các bản phát hành**: Bằng việc kiểm thử trong môi trường sản xuất, bạn sẽ tạo nên một mức độ tự tin khác trong các bản phát hành sau khi thực hiện các kiểm tra khác nhau trong một môi trường sản xuất](https://viblo.asia/p/tai-sao-ban-nen-kiem-thu-tren-moi-truong-production-WAyK8oOm5xX).

[**Hiểu ứng dụng trong thế giới thực**: Kiểm thử trong môi trường sản xuất cho phép công ty thấy được một ứng dụng phản ứng thế nào với khi mà có code mới được đẩy vào trong thế giới người dùng thực](https://viblo.asia/p/tai-sao-ban-nen-kiem-thu-tren-moi-truong-production-WAyK8oOm5xX).

[**Phát hiện lỗi không thể tìm thấy trong môi trường kiểm thử khác**: Bạn sẽ tìm thấy một tập hợp các lỗi mà bạn không tìm thấy khi kiểm thử trong các môi trường kiểm thử khác (dev, staging, hoặc pre-prod)](https://viblo.asia/p/tai-sao-ban-nen-kiem-thu-tren-moi-truong-production-WAyK8oOm5xX).

[**Cải thiện chất lượng ứng dụng**: Các lỗi được thu thập ở trong môi trường sản xuất sẽ giúp nhóm phát triển cô lập các lỗi để cải thiện chất lượng ứng dụng, từ đó cung cấp một trải nghiệm khách hàng tốt hơn](https://viblo.asia/p/tai-sao-ban-nen-kiem-thu-tren-moi-truong-production-WAyK8oOm5xX).

### ****Lợi ích chính của Machine Learning trong kiểm thử phần mềm****

Học máy mang lại nhiều lợi ích trong kiểm thử phần mềm, bao gồm:

● Tăng hiệu quả

● Cải thiện độ chính xác

● Khả năng thích ứng với các yêu cầu thay đổi

● Nâng cao khả năng ra quyết định

Hãy thảo luận chi tiết hơn về những lợi ích này.

**Tăng hiệu quả**

Thuật toán ML có thể tự động hóa các tác vụ lặp đi lặp lại và tốn thời gian, giải phóng thời gian quý báu cho người thử nghiệm để tập trung vào các khía cạnh phức tạp và quan trọng hơn của quá trình thử nghiệm. Kết quả là hiệu quả tổng thể của quá trình thử nghiệm được cải thiện đáng kể.

**Cải thiện độ chính xác**

Các thuật toán học máy có thể phân tích lượng lớn dữ liệu một cách nhanh chóng và chính xác, giảm khả năng xảy ra lỗi của con người. Điều này dẫn đến kết quả kiểm tra đáng tin cậy hơn và phần mềm chất lượng cao hơn.

**Khả năng thích ứng**

Các thuật toán ML có thể thích ứng với các yêu cầu thay đổi và học hỏi từ dữ liệu mới, khiến chúng trở nên linh hoạt và linh hoạt hơn. Khả năng thích ứng này là cần thiết trong bối cảnh công nghệ phát triển nhanh chóng ngày nay.

**Nâng cao khả năng ra quyết định**

Bằng cách cung cấp thông tin chi tiết chuyên sâu và phân tích dự đoán, thuật toán ML có thể hỗ trợ người thử nghiệm đưa ra quyết định dựa trên dữ liệu tốt hơn trong suốt quá trình thử nghiệm.

### ****Ứng dụng Machine Learning trong kiểm thử phần mềm****

Dưới đây là một số ứng dụng hứa hẹn nhất của machine learning trong kiểm thử phần mềm:

**Ưu tiên trường hợp kiểm thử:** Thuật toán ML có thể phân tích dữ liệu kiểm thử lịch sử để xác định các trường hợp kiểm thử có mức độ ưu tiên cao, đảm bảo rằng chức năng quan trọng nhất được kiểm tra trước.

**Dự đoán lỗi:** Bằng cách phân tích dữ liệu lỗi lịch sử, thuật toán ML có thể dự đoán khả năng xảy ra lỗi ở các khu vực cụ thể của phần mềm, cho phép người kiểm tra tập trung nỗ lực vào các khu vực có rủi ro cao.

**Tối ưu hóa bộ kiểm thử:** Học máy có thể giúp xác định các trường hợp kiểm thử dư thừa hoặc lỗi thời, cho phép người kiểm tra tối ưu hóa bộ kiểm thử của họ để đạt hiệu quả và hiệu quả tối đa.

**Tạo thử nghiệm tự động:** Thuật toán ML có thể tạo ra các trường hợp thử nghiệm dựa trên yêu cầu của phần mềm, giảm nỗ lực thủ công và cải thiện phạm vi thử nghiệm.

**Kiểm tra hiệu suất:** Học máy có thể được sử dụng để phân tích dữ liệu hiệu suất và dự đoán các tắc nghẽn hoặc sự cố tiềm ẩn, cho phép người kiểm tra chủ động giải quyết các mối lo ngại về hiệu suất

TÀI LIỆU THAM KHẢO

1. [What is Continuous Learning? Revolutionizing Machine Learning & Adaptability | DataCamp](https://www.datacamp.com/blog/what-is-continuous-learning)
2. [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)
3. [Thuật toán Gradient Descent cho Linear Regression | How Kteam](https://howkteam.vn/course/machine-learning-co-ban-voi-numpy/thuat-toan-gradient-descent-cho-linear-regression-3997)
4. <https://machinelearningcoban.com/>
5. <https://phamdinhkhanh.github.io/>
6. <https://viblo.asia/>