TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRƯƠNG THÀNH HUY - 52100628**

**BÁO CÁO CUỐI KỲ MÔN**

**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**



**TRƯƠNG THÀNH HUY - 52100628**

**BÁO CÁO CUỐI KỲ MÔN**

**NHẬP MÔN HỌC MÁY**

Người hướng dẫn

**PGS.TS. Lê Anh Cường**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Đầu tiên, em xin gửi lời cảm ơn đến Trường Đại học Tôn Đức Thắng vì đã đưa môn nhập môn Học máy vào chương trình giảng dạy. Đặc biệt,em xin gửi lời cám ơn sâu sắc đến giảng viên của bộ môn Thầy Lê Anh Cường đã truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập. Đây chắc chắn sẽ là những kiến thực quý báu trong hành trang để chúng em có thể đi xa hơn trong tương lại

Đây là một môn học vô cùng thú vị, bổ ích và có tính thực tế. Đảm bảo cung cấp đầy đủ kiến thức và gắn liền với nhu cầu thực tiễn của sinh viên Công Nghệ Thông Tin. Mặc dù em đã cố gắng hết sức những dự án sẽ có thể có những thiếu sót và những sai sót trong dự án, kính mong thầy cô xem xét và góp ý để dự án cuối kỳ của em được hoàn thiện hơn.

Em xin chân thành cảm ơn.

*TP. Hồ Chí Minh, ngày 19 tháng 12 năm 2023*

*Tác giả*

*Huy*

*Trương Thành Huy*

**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.TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 19 tháng 12 năm 2023*

*Tác giả*

*Huy  
Trương Thành Huy*

**TÓM TẮT**

Bài báo của em trình bày các kiến thức,lý thuyết về tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy và Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ v](#_Toc143173056)

[DANH MỤC BẢNG BIỂU vii](#_Toc143173057)

[DANH MỤC CÁC CHỮ VIẾT TẮT viii](#_Toc143173058)

[CHƯƠNG 1. Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy 1](#_Toc143173059)

[1. Gradient Descent 2](#_Toc143173060)

[1.1 Giới thiệu 2](#_Toc143173060)

[1.2 Các bước thuật toán 4](#_Toc143173060)

[2. Adam (Adaptive Moment Estimation) 5](#_Toc143173060)

[2.1 Giới thiệu 5](#_Toc143173060)

[2.2 Các bước thuật toán 6](#_Toc143173060)

[3. Adagrad 7](#_Toc143173061)

[3.1 Giới thiệu 7](#_Toc143173060)

[3.2 Công thức 7](#_Toc143173060)

[3.3 Thuật toán 8](#_Toc143173060)

[4. RMSProp - Root Mean Square Root 8](#_Toc143173060)

[4.1 Giới thiệu 8](#_Toc143173060)

[4.2 Công thức 9](#_Toc143173060)

[4.3 Thuật toán 9](#_Toc143173060)

[5. Momentum 9](#_Toc143173060)

[6. Stochastic Gradient Descent (SGD) 11](#_Toc143173060)

[6.1 Giới thiệu 11](#_Toc143173060)

[6.2 Công thức 1](#_Toc143173060)2

[6.3 Thuật toán 1](#_Toc143173060)2

[CHƯƠNG 2. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy. 13](#_Toc143173062)

[1. Continual Learning 1](#_Toc143173060)3

[1.1 Khái niệm 1](#_Toc143173060)3

[1.2 Thách thức và cách khắc phục 1](#_Toc143173060)4

[1.3 Xây dựng 1](#_Toc143173060)5

[2. Test Production 20](#_Toc143173060)

[2.1 Các cách kiểm tra 20](#_Toc143173060)

[2.2 Ý nghĩa 21](#_Toc143173060)

[2.3 Quá trình thử nghiệm 22](#_Toc143173060)

[2.4 Thách thức 22](#_Toc143173060)

[2.5 Kết Luận 22](#_Toc143173060)

[TÀI LIỆU THAM KHẢO 24](#_Toc143173076)

# DANH MỤC HÌNH VẼ

[Hình 1 Các mô hình nghiên cứu 1](#_Toc142677545)

[Hình 2 Cách thức hoạt động 2](#_Toc142677545)

[Hình 3 Gradient 3](#_Toc142677545)

[Hình 4 So sánh các mô hình 8](#_Toc142677545)

[Hình 5 GD-GD with Momentum 11](#_Toc142677545)

[Hình 6 GD -SGD 11](#_Toc142677545)

[Hình 7 Update model 1](#_Toc142677545)8

[Hình 8 A/B Testing 20](#_Toc142677545)

# DANH MỤC CÁC CHỮ VIẾT TẮT

RMSProp - Root Mean Square Root

GD -Gradient Descent

SGD -Stochastic Gradient Descent

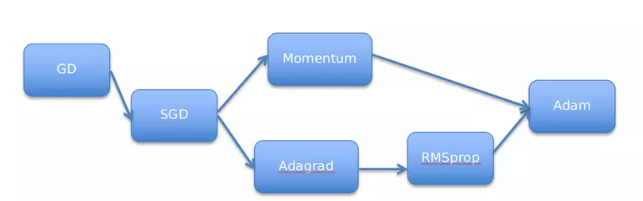
# Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máyLý do chọn đề tài

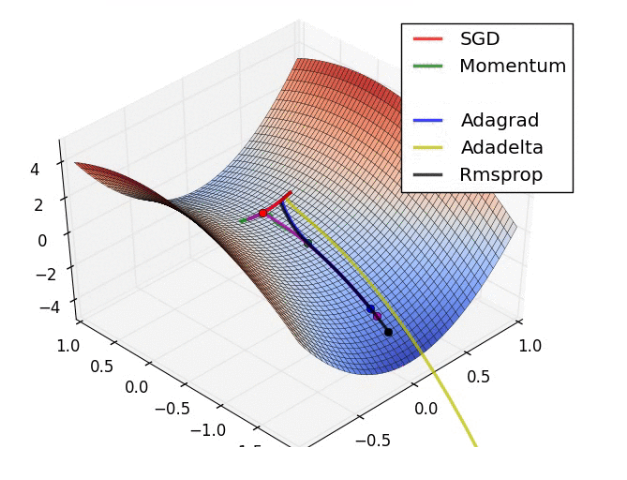
**Sơ lược về optimizers:**

"thuật toán tối ưu" (optimizers). Thuật toán tối ưu là nền tảng giúp xây dựng mô hình neural network để học các đặc trưng (hoặc mẫu) từ dữ liệu đầu vào. Mục tiêu là tìm ra một bộ trọng số (weights) và độ lệch (bias) phù hợp nhằm tối ưu hóa mô hình. Tuy nhiên, quan trọng là chúng ta cần hiểu quá trình "học" này diễn ra như thế nào.

Việc tìm kiếm trọng số và độ lệch không thể chỉ dựa vào việc ngẫu nhiên chọn (weights, bias) một số lần và hy vọng rằng ở một điểm nào đó chúng ta sẽ tìm được lời giải. Điều này rõ ràng không khả thi và lãng phí tài nguyên. Chúng ta cần một phương pháp để cải thiện trọng số và độ lệch từng bước một. Đó là lý do tại sao các thuật toán tối ưu ra đời.

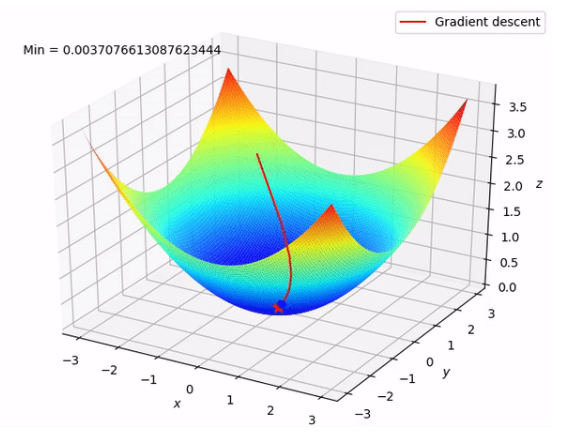
Những thuật toán này giúp mô hình học bằng cách điều chỉnh trọng số và độ lệch dựa trên độ chệch giữa dự đoán của mô hình và giá trị thực tế của dữ liệu huấn luyện. Chúng ta có thể điều chỉnh các tham số này sao cho độ chệch giảm dần theo thời gian, đồng thời cải thiện khả năng dự đoán của mô hình.





**1 Gradient Descent**

* 1. **Giới thiệu :**



Gradient Descent là một thuật toán tối ưu lặp (iterative optimization algorithm) được sử dụng trong các bài toán Machine Learning và Deep Learning (thường là các bài toán tối ưu lồi — Convex Optimization) với mục tiêu là tìm một tập các biến nội tại (internal parameters) cho việc tối ưu models. Trong đó:

* Gradient: là tỷ lệ độ nghiêng của đường dốc (rate of inclination or declination of a slope). Về mặt toán học, Gradient của một hàm số là đạo hàm của hàm số đó tương ứng với mỗi biến của hàm. Đối với hàm số đơn biến, chúng ta sử dụng khái niệm Derivative thay cho Gradient.
* Descent: là từ viết tắt của descending, nghĩa là giảm dần.

Gradient Descent có nhiều dạng khác nhau như Stochastic Gradient Descent (SGD), Mini-batch SDG

Trong lĩnh vực Machine Learning và Toán Tối Ưu, chúng ta thường phải tìm giá trị nhỏ nhất hoặc lớn nhất của một hàm số, như các hàm mất mát trong Linear Regression và K-means Clustering. Tuy nhiên, việc tìm ra giá trị tối thiểu toàn cục của các hàm mất mát trong Machine Learning thường là phức tạp, thậm chí là không khả thi. Thay vào đó, chúng ta thường tìm các điểm tối thiểu cục bộ và coi chúng là nghiệm của bài toán.

Các điểm tối thiểu cục bộ là nghiệm của phương trình đạo hàm bằng 0. Nếu chúng ta có thể tìm được toàn bộ các điểm tối thiểu cục bộ, chúng ta có thể thay từng điểm này vào hàm số và tìm ra điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, việc giải phương trình đạo hàm bằng 0 thường là không khả thi do sự phức tạp của dạng đạo hàm, số chiều lớn của dữ liệu, hoặc số lượng lớn các điểm dữ liệu.

Một phương pháp phổ biến để giải quyết vấn đề này là bắt đầu từ một điểm gần với nghiệm và sử dụng phép toán lặp để tiến dần đến điểm cần tìm. Gradient Descent (GD) và các biến thể của nó là một trong những phương pháp được sử dụng nhiều nhất.

**1.2.Các bước của thuật toán:**

* Khởi tạo biến nội tại.
* Đánh giá model dựa vào biến nội tại và hàm mất mát (Loss function).
* Cập nhật các biến nội tại theo hướng tối ưu hàm mất mát (finding optimal points).
* Lặp lại bước 2, 3 cho tới khi thỏa điều kiện dừng.

Công thức cập nhật :

1 số vấn đề cần quan tâm đặt ra khi sử dụng thuật toán :

\* Việc chọn tham số learning rate

Viêc chọn tham số trên anh hưởng rất nhiều đến viêc có tìm được tham số tối ưu không. Ảnh dưới đây thể hiện khái quát vấn đề của tham số:

* Nếu learning rate quá nhỏ, mỗi lần cập nhật hàm số giảm rất chậm, phải mất rất nhiều thời gian để có thể đến được điểm tối ưu.
* Nếu learning rate quá lớn, mỗi lần cập nhật quá lớn khiến cho rất lâu mới có thể hội tụ tại điểm tối ưu, thường được gọi là hiện tượng overshoot.

Cách khắc phục:

* Điều tra tham khảo các tham số learning rate của các model đã được nghiên cứu trước để chọn sao cho phù hợp
* Tuning hyper parametter
* Visualize hàm số để hiểu về hàm số cần tối ưu

\*Nếu hàm số có nhiều điểm cực tiểu

Nếu learning rate không đủ lớn rất có thể điểm cực tiểu chúng ta tìm thấy không phải điểm cực tiểu tối ưu nhất

Cách khắc phục: sử dụng thuật toán momentum.

\* Việc tính toán đạo hàm của hàm nhiều biến

Đối với việc tính đạo hàm của các hàm nhiều biến thường rất khó khăn, khá phức tạp và dễ mắc lỗi, nên có thể dẫn tới việc tính sai đạo hàm làm quá trì tìm tham số tối ưu bị sai lệch.

Cách khắc phục: sử dụng các công thức tính xấp sỉ dạo hàm để kiểm tra xem việc tính đạo hàm có chính xác.

**2. Adam (Adaptive Moment Estimation)**

**2.1 Giới thiệu**

**\* Adam là gì:**

* Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa các hàm mục tiêu ngẫu nhiên dựa trên gradient bậc nhất và sự ước tính thích ứng của các mô-men (moment) bậc thấp.
* Một phương pháp tối ưu hóa ngẫu nhiên hiệu quả chỉ yêu cầu gradient bậc nhất với bộ nhớ thấp.
* Phương pháp này tính toán từng tỷ lệ học thích ứng cho các tham số khác nhau.
* Adam sử dụng ước tính của khoảng thời gian thứ nhất và thứ hai để điều chỉnh tốc độ học cho mỗi trọng số của mạng nơ-ron.

**\* Adam sinh ra từ đâu**

* Là sự kết hợp những lợi thế của hai thuật toán tối ưu AdaGrad và RMSProp nhằm giảm dần độ dốc ngẫu nhiên.
* Adam thay vì điều chỉnh các tham số learning rate dựa trên thời điểm trung bình đầu tiên (giá trị trung bình) như RMSProp, Adam sử dụng giá trị trung bình của thời điểm thứ hai của các gradient (Phương sai không tập trung)
* **Adagrad**
  + Thực hiện giảm dần độ dốc bằng cách thay đổi tốc độ học tập
  + Được cải thiện hơn bằng cách cho trọng số học tập chính xác dựa vào đầu vào trước nó để tự điều chỉnh tỉ lệ học theo hướng tối ưu nhất thay vì với một tỉ lệ học duy nhất cho tất cả các nút.
* **RMSProp - Root Mean Square Root**
  + RMSProp là phương pháp điều chỉnh tỷ lệ học (learning rate)
  + Sử dụng trung bình bình phương của gradient để chuẩn hóa gradient
  + Có tác dụng cân bằng kích thước bước - giảm bước cho độ dốc lớn để tránh hiện tượng phát nổ độ dốc (Exploding Gradient), và tăng bước cho độ dốc nhỏ để tránh biến mất độ dốc (Vanishing Gradient).
  + RMSProp tự động điều chỉnh tốc độ học tập, và chọn một tỉ lệ học tập khác nhau cho mỗi tham số.

RMSProp cũng chia learning rate cho   .Hinton đề nghị và giá trị mặc định tốt cho

**2.2 Thuật toán**

Adam, ký hiệu gt2 là bình phương phép nhân element-wise , các tham số khuyến nghị là .Tất cả các toán tử trên vector đều là phép element-wise (tức là hiện tượng tương ứng của từng phần tử)

Ở đây  là các tham số trọng số không âm. Các lựa chọn phổ biến cho chúng là =0.9 và  = 0.999 Điều này có nghĩa là ước lượng phương sai di chuyển chậm hơn nhiều so với số hạng động lượng.  nếu ta khởi tạo =0 =0, thuật toán sẽ có độ chệch ban đầu đáng kể về các giá trị nhỏ hơn. Vấn đề này có thể được giải quyết bằng cách sử dụng

để chuẩn hóa lại các số hạng. Tương tự, các biến trạng thái được chuẩn hóa như sau

and

Đầu tiên, chúng ta điều chỉnh lại giá trị gradient,

**3. Adagrad**

**3.1 Giới thiệu.**

Adagrad là một thuật toán để tối ưu hóa dựa trên gradient descent, thuật toán này giúp điều chỉnh tốc độ học ứng với các tính năng khác nhau, thực hiện các cập nhật nhỏ hơn (tức là tốc độ học thấp) cho các tham số liên quan đến các tính năng thường xuyên xuất hiện và các cập nhật lớn hơn (tức là tốc độ học cao) cho các tham số liên quan đến các tính năng không thường xuyên. Vì lý do này, Adagrad rất thích hợp để xử lý dữ liệu thưa thớt.

Adagrad hiệu quả do cách tính đượ thay bằng tổng bình phương của tất cả các gradient được quan sát trước đó. Cụ thể, nó sử dụng làm công cụ để điều chỉnh tốc độ học. Việc này có hai ưu điểm:

* Thứ nhất ta không cần phải quyết định khi nào thì gradient được coi là đủ lớn.
* Thứ hai nó tự động thay đổi giá trị tùy theo độ lớn của gradient. Các tọa độ thường xuyên có gradient lớn bị giảm đáng kể, trong khi các tọa độ khác với grandient nhỏ được xử lý nhẹ nhàng hơn

**3.2 Công thức**

Trong đó :

* : hằng số
* : gradient tại thời điểm t
* : hệ số tránh lỗi ( 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.

Các phép toán được thực hiện theo từng tọa độ.

**3.3 Thuật toán:**

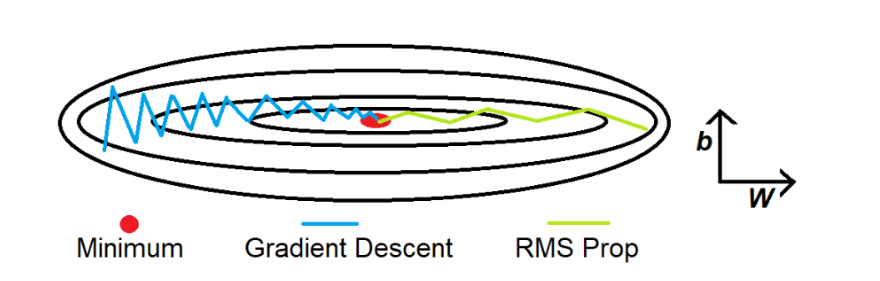
Ta sử dụng biến  để tích luỹ phương sai của các gradient trong quá khứ

Tương tự như trường hợp sử dụng động lượng, ta cần phải theo dõi các biến bổ trợ để mỗi toạ độ có một tốc độ học độc lập. Cách này không làm tăng chi phí của Adagrad, lý do đơn giản là bởi chi phí chính yếu thường nằm ở bước tính  và đạo hàm của nó.

Tổng bình phương của các gradient trong thuật toán có thể được hiểu một cách cơ bản là giá trị của tăng dần theo cách tuyến tính, mặc dù ở mức ban đầu, các gradient ban đầu sẽ được co lại. Điều này dẫn đến tốc độ học là , mặc dù nó đã được điều chỉnh cho từng tọa độ riêng lẻ. Điều này hoàn toàn đủ đối với các bài toán lồi.

**4. RMSProp - Root Mean Square Root**

**4.1 Giới thiệu**

****

RMSProp là một thuật toán tối ưu hóa thông thường được sử dụng trong gradient descent, đặc biệt phù hợp để xử lý dữ liệu thưa thớt và điều chỉnh tỷ lệ học động đối với các đặc trưng khác nhau. RMSProp nhằm mục tiêu điều chỉnh tỷ lệ học động một cách động dựa trên thông tin lịch sử về gradient. Việc này giúp thực hiện các cập nhật nhỏ hơn (tốc độ học động thấp) cho các đặc trưng thường xuyên xuất hiện và các cập nhật lớn hơn (tốc độ học động cao) cho các đặc trưng ít xuất hiện, hiệu quả trong việc xử lý dữ liệu thưa thớt. RMSProp hiệu quả vì nó thay thế tổng bình phương gradient đơn giản được sử dụng trong Adagrad bằng một trung bình động của gradient bình phương. Phương pháp này mang lại hai lợi ích:

* Chúng ta không cần phải đặt một ngưỡng để quyết định khi nào gradient được xem là đủ lớn.
* Tốc độ học động tự động được điều chỉnh dựa trên độ lớn của gradient. Các đặc trưng thường xuyên với gradient lớn bị giảm đáng kể, trong khi các đặc trưng ít xuất hiện với gradient nhỏ được xử lý nhẹ nhàng hơn.

Nhược điểm:

* Thuật toán RMSprop có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum .

**4.2 Công thức**

**4.3 Thuật toán**

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.Đầu tiên chúng ta cần phải

* Khởi tạo biến: Bắt đầu bằng cách khởi tạo các biến, bao gồm tỷ lệ học toàn cục và một hằng số nhỏ (thường là để tránh việc chia cho không).
* Tính đạo hàm: Tính toán đạo hàm của hàm mất mát của mô hình đối với các tham số của nó.
* Tính bình phương đạo hàm: Đối với mỗi tham số, tính giá trị bình phương của đạo hàm của nó. Điều này giúp tập trung vào các tham số có đạo hàm lớn hơn.
* Trung bình động của bình phương đạo hàm: Duy trì một giá trị trung bình động của các bình phương đạo hàm này theo thời gian, đóng vai trò là một phiên bản trơn của các đạo hàm gần đây.
* Cập nhật tham Số: Điều chỉnh tốc độ học cho từng tham số bằng cách chia đạo hàm hiện tại cho căn bậc hai của giá trị trung bình động của bình phương đạo hàm. Điều này đảm bảo rằng các tham số có đạo hàm lớn sẽ có tốc độ học nhỏ hơn và ngược lại.

**5.Momentum**

Để hiểu Momentum từ góc độ vật lý, hãy tưởng tượng có hai viên bi thả từ hai điểm khác nhau trên một đồi. Một viên bi sẽ trượt xuống một vị trí cụ thể, và viên bi còn lại sẽ trượt xuống một vị trí khác. Tuy nhiên, muốn viên bi thứ hai tiếp tục lăn xuống vị trí của viên bi thứ nhất, không dừng lại ở một vị trí cục bộ. Để làm được điều này,ban đầu tạo cho viên bi thứ hai một vận tốc đủ lớn để vượt qua vị trí cục bộ và đến vị trí của viên bi thứ nhất. Thuật toán Momentum thực hiện tương tự, nó giúp đối phó với vấn đề dừng lại ở điểm cục bộ.

Từ góc độ toán học, công thức của thuật toán Momentum có dạng:

Trong đó:

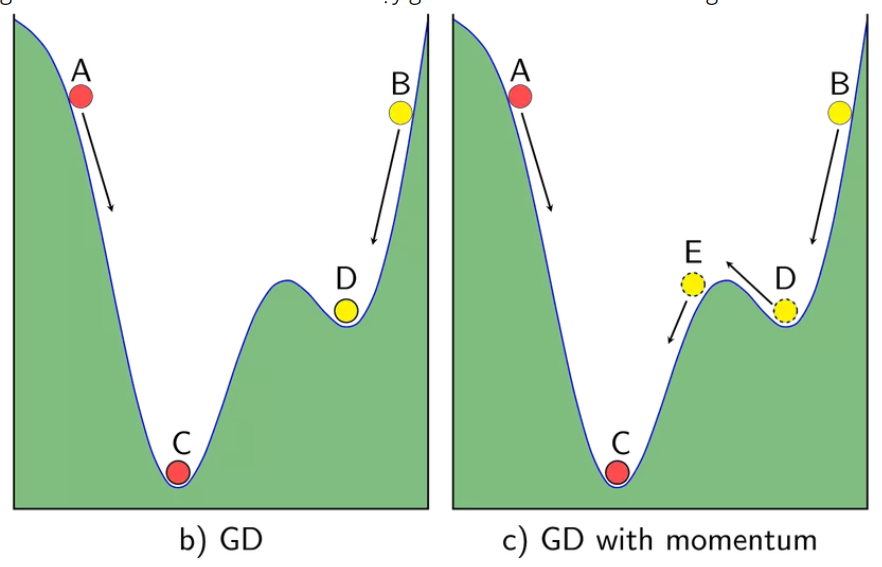
* : tọa độ mới của tham số cần tối ưu hóa
* : tọa độ cũ của tham số.
* là một biến tốc độ
* gamma: một tham số được gọi là momentum (thường có giá trị khoảng 0.9).
* learning\_rate: tốc độ học, giống như trong Gradient Descent.
* gradient: đạo hàm của hàm mất mát.

Ưu điểm của Gradient with Momentum:

* Giúp tránh dừng lại ở điểm cục bộ (local minimum) bằng cách tạo đà cho quá trình tối ưu hóa.
* Giúp hội tụ nhanh hơn và điều chỉnh mượt mà hơn đối với hàm mất mát có định dạng phức tạp.

Nhược điểm:

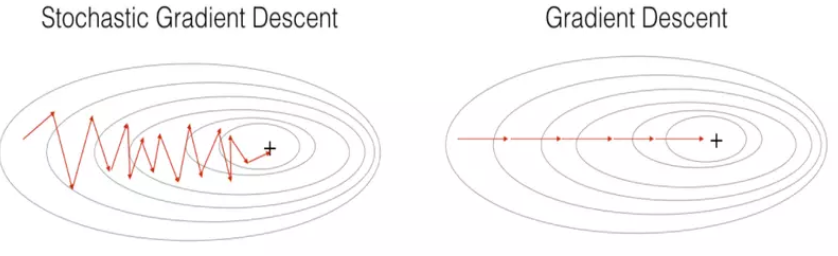
* Khi tiến đến gần điểm cực tiểu (global minimum), thuật toán vẫn mất thời gian giao động qua lại trước khi dừng hoàn toàn. Điều này liên quan đến khái niệm về đà (momentum), nó có thể khiến việc dừng lại trở nên không ổn định.



**6. Stochastic Gradient Descent (SGD)**

**6.1 Giới thiệu**

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.



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,..Các thuật toán này sẽ được trình bày ở phần sau

**6.2 Công thức:**

**6.3 Thuật toán:**

* Khởi tạo tham số: Bắt đầu bằng việc khởi tạo ngẫu nhiên các tham số ban đầu của mô hình. Các tham số này sẽ được điều chỉnh theo hướng giảm thiểu hàm mất mát.
* Lặp qua dữ liệu: SGD hoạt động qua từng ví dụ trong tập dữ liệu huấn luyện một cách ngẫu nhiên. Điều này khác với Gradient Descent thông thường, nơi một vòng lặp duy nhất sử dụng toàn bộ tập dữ liệu.
* Tính đạo hàm: Tại mỗi vòng lặp, tính đạo hàm của hàm mất mát tại tham số hiện tại. Đạo hàm này cho biết hướng và độ lớn của thay đổi cần thiết để giảm thiểu hàm mất mát cho ví dụ hiện tại.
* Cập nhật tham số: Sử dụng đạo hàm tính được và tốc độ học (learning rate), cập nhật tham số của mô hình.
* Kết quả: Sau khi hoàn thành các vòng lặp, tham số của mô hình sẽ được điều chỉnh để cố gắng tối ưu hóa hàm mất mát và thích nghi tốt hơn với dữ liệu huấn luyện.

# Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy.

## Continual Learning

* 1. Khái niệm

Học liên tục là một mô hình đào tạo liên quan đến việc cập nhật mô hình với dữ liệu mới theo thời gian. Tuy nhiên, việc cập nhật mô hình với mỗi mẫu dữ liệu đến trong quá trình sản xuất có thể dẫn đến việc lãng quên dữ liệu cũ, nơi mô hình đột ngột quên thông tin đã học trước đây. Để giải quyết vấn đề này, hiện nay thường sử dụng micro-batches để cập nhật mô hình, cho phép họ đạt được sự cân bằng giữa việc cập nhật thường xuyên và tính hiệu quả tính toán.

Quy trình này bao gồm việc tạo một bản sao của mô hình hiện tại, được, và cập nhật nó với dữ liệu mới. Mô hình cập nhật này sau đó được đánh giá so với mô hình hiện tại. Nếu mô hình mới vượt trội hơn thì sẽ được cập nhật thay thế mô hình hiện tại. Tuy nhiên, quản lý nhiều mô hình mới cùng một lúc và xử lý thất bại có thể phức tạp hơn so với quy trình đơn giản này.

Ngược lại với quan điểm rằng học liên tục liên quan đến việc cập nhật thường xuyên, có thể có những mô hình không cần phải cập nhật thường xuyên do lưu lượng dữ liệu hạn chế và sự suy giảm chậm của mô hình. Chìa khóa là tìm ra lịch trình huấn luyện lại tối ưu cân bằng giữa nhu cầu cập nhật và hiệu quả tính toán.

Học liên tục có thể được phân loại thành việc đào tạo không lưu trạng thái và đào tạo lưu trạng thái. Đào tạo không lưu trạng thái bao gồm việc đào tạo mô hình từ đầu mỗi lần, trong khi đào tạo lưu trạng thái cho phép mô hình tiếp tục học trên dữ liệu mới. Đào tạo lưu trạng thái, còn được biết đến là làm mịn hoặc học bổ sung, có lợi thế về việc cập nhật mô hình với ít dữ liệu hơn, giảm chi phí tính toán đào tạo và tăng hiệu suất. Nó cũng có thể loại bỏ nhu cầu lưu trữ dữ liệu vĩnh viễn.

*\*Ý Tưởng Cốt Lõi: Học Tập Liên Tục bao gồm việc cập nhật định kỳ các mô hình ML để thích ứng với dữ liệu mới, chủ yếu để giải quyết sự thay đổi phân phối dữ liệu trong môi trường sản xuất..*

*\*Hiểu Lầm Thường Gặp: Thường được nghĩ là cập nhật mô hình với mỗi mẫu dữ liệu mới ngay lập tức, nhưng thực tế hiếm khi xảy ra do các mô hình mới như quên dữ liệu cũ và không hiệu quả trong xử lý từng mẫu dữ liệu.*

**1.2 Thách thức và cách khắc phục:**

Có 3 thách thức chính mà em đã tìm hiểu được:

Thách thức Truy cập dữ liệu mới (Fresh Data Access Challenge): Để cập nhật mô hình mỗi giờ, cần dữ liệu mới mỗi giờ. Tốc độ kéo dữ liệu từ kho dữ liệu phụ thuộc vào tốc độ dữ liệu được gửi đến kho dữ liệu. Một giải pháp thay thế là kéo dữ liệu trực tiếp từ các dịch vụ truyền tải thời gian thực như Kafka và Kinesis.

Thách thức đánh giá (Evaluation Challenge): Thách thức lớn nhất trong học liên tục không phải viết hàm cập nhật mô hình mà là đảm bảo rằng cập nhật này đủ tốt để triển khai. Cần thực hiện kiểm tra kỹ lưỡng mỗi cập nhật mô hình để đảm bảo hiệu suất và an toàn trước khi triển khai rộng rãi. Rủi ro của sự cố tăng lên với tần suất cập nhật mô hình cao hơn.

Thách thức thuật toán (Algorithm Challenge): Thách thức này chỉ ảnh hưởng đến một số thuật toán và tần suất huấn luyện nhất định, đặc biệt là với mô hình dựa trên ma trận và cây quyết định cần được cập nhật rất nhanh (ví dụ: hàng giờ). Các mô hình như mạng nơ-ron thích nghi tốt hơn với mô hình học liên tục so với mô hình dựa trên ma trận và cây.Test Production

**Cách khắc phục :**

Giảm chiều dữ liệu và Mô hình dựa trên ma trận: Mô hình lọc cộng tác dựa vào một ma trận người dùng-sản phẩm và các kỹ thuật giảm chiều dữ liệu. Việc cập nhật thường xuyên yêu cầu xây dựng lại toàn bộ ma trận người dùng-sản phẩm trước khi áp dụng giảm chiều dữ liệu. Thực hiện giảm chiều dữ liệu trên các ma trận lớn với mỗi cập nhật có thể chậm và tốn kém, làm cho nó ít phù hợp hơn cho việc học liên tục với bộ dữ liệu một phần.

Mô hình dựa trên cây: Trong khi mạng neural linh hoạt hơn đối với học liên tục, các mô hình dựa trên cây, như Cây Hoeffding và các biến thể của nó, như Cây Hoeffding Window và Cây Hoeffding Adaptive, có thể học từ lượng dữ liệu tăng dần. Tuy nhiên, những thuật toán này chưa phổ biến.

Mã trích xuất đặc trưng: Thuật toán học và mã trích xuất đặc trưng phải có khả năng làm việc với bộ dữ liệu một phần. Việc tỉ lệ giảm kích thước đặc trưng (ví dụ: sử dụng min, max, trung bình, phương sai) thường đòi hỏi duyệt qua toàn bộ bộ dữ liệu. Khi chỉ có một phần dữ liệu, việc tính toán những thống kê này cho từng phần có thể dẫn đến dao động lớn, ảnh hưởng đến khả năng tổng quát hóa của mô hình qua các phần khác nhau.

Tính toán trực tuyến của thống kê: Để duy trì tính ổn định qua các phần khác nhau, việc tính toán thống kê trực tuyến là quan trọng. Điều này bao gồm tính toán hoặc ước lượng thống kê một cách tăng dần khi có dữ liệu mới.Các framework hiện tại cung cấp một số khả năng tính toán thống kê trực tuyến, nhưng các phương pháp tích hợp thường chậm và không hỗ trợ một loạt các thống kê trực tuyến.

**1.3 Ý tưởng xây dựng:**

**Lý thuyết**

Nguyên Lý Cơ Bản: Mô hình học liên tục được thiết kế để thích nghi với sự thay đổi trong phân phối dữ liệu qua thời gian, giúp mô hình duy trì hiệu suất tốt ngay cả khi dữ liệu đầu vào thay đổi. Xử lý vấn đề "quên lãng" (catastrophic forgetting) trong đó mô hình mất dần khả năng dự đoán chính xác dựa trên dữ liệu cũ khi học dữ liệu mới.

Cách tiếp cận:

Học có trạng thái (Stateful Learning): Thay vì huấn luyện lại từ đầu, mô hình tiếp tục học trên dữ liệu mới, giữ lại kiến thức đã học trước đó.

Xử lý dữ liệu mới (Fresh Data Handling): Tích hợp dữ liệu mới liên tục vào quá trình huấn luyện mô hình.

Đánh giá và kiểm thử: Áp dụng các kỹ thuật kiểm thử liên tục để đánh giá hiệu suất mô hình với dữ liệu thực tế trong sản xuất.

**Thực tiễn:**

Thiết Lập Cơ Sở Hạ Tầng: Xây dựng hệ thống hạ tầng hỗ trợ việc cập nhật mô hình liên tục, bao gồm cơ sở dữ liệu, hệ thống xử lý dữ liệu thời gian thực và lưu trữ mô hình. Tích hợp hệ thống theo dõi và giám sát để phát hiện sự thay đổi trong dữ liệu và hiệu suất mô hình.

Xử Lý Dữ liệu: Phát triển cơ chế tự động để thu thập, xử lý và nhãn dữ liệu mới. Áp dụng kỹ thuật xử lý dữ liệu thời gian thực như Apache Kafka hoặc AWS Kinesis.

Cập Nhật và Triển Khai Mô Hình: Sử dụng các mô hình "champion" và "challenger" để so sánh hiệu suất giữa mô hình hiện tại và mô hình đã cập nhật. Áp dụng các kỹ thuật kiểm thử trong sản xuất như A/B testing, shadow deployment hoặc canary releases để kiểm tra mô hình mới trước khi triển khai rộng rãi.

Quản Lý và Kiểm Soát: Thiết lập quy trình quản lý phiên bản mô hình để theo dõi lịch sử thay đổi và cải tiến. Thực hiện các biện pháp an ninh và quản lý rủi ro để đảm bảo an toàn dữ liệu và mô hình.

Gồm 4 bước để xây dựng:

Giai đoạn 1: Huấn luyện lại thủ công, không lưu trạng thái Đội ngũ ML tập trung vào việc phát triển các mô hình ML để giải quyết càng nhiều vấn đề kinh doanh càng tốt. Các mô hình hiện tại chỉ được cập nhật khi hiệu suất giảm đến mức gây hại, và đội ngũ có thời gian cập nhật. Quá trình cập nhật mô hình là thủ công và tùy ý, đòi hỏi người kỹ sư dữ liệu thực hiện các bước truy vấn dữ liệu, làm sạch dữ liệu, trích xuất đặc trưng, huấn luyện lại mô hình từ đầu và triển khai mô hình cập nhật. Có thể có thay đổi mã trong quá trình huấn luyện nhưng không được sao chép đến môi trường sản xuất, gây ra lỗi khó theo dõi.

Giai đoạn 2: Huấn luyện tự động Sau vài năm, đội ngũ đã triển khai các mô hình để giải quyết hầu hết các vấn đề rõ ràng. Ưu tiên không còn là phát triển mô hình mới, mà là duy trì và cải thiện các mô hình hiện tại. Quá trình cập nhật thủ công và tùy ý trở nên quá đau đớn để bỏ qua. Đội quyết định viết một script để tự động hóa tất cả các bước cập nhật. Kịch bản này sau đó được chạy định kỳ bằng các quy trình đợt như Spark. Hầu hết các công ty có cơ sở hạ tầng ML đủ mạnh mẽ đều ở giai đoạn này. Một số công ty tinh tế chạy thử nghiệm để xác định tần suất cập nhật tối ưu. Tuy nhiên, đối với hầu hết các công ty ở giai đoạn này, tần suất cập nhật được đặt dựa trên cảm giác cá nhân - ví dụ: "mỗi ngày một lần có vẻ đúng" hoặc "hãy kích hoạt quá trình cập nhật mỗi đêm khi chúng ta có máy tính không sử dụng". Khi tạo script để tự động hóa quá trình cập nhật hệ thống, cần xem xét rằng các mô hình khác nhau trong hệ thống có thể yêu cầu các lịch trình cập nhật khác nhau. Ví dụ, xem xét một hệ thống gợi ý gồm hai mô hình: một mô hình để tạo ra các nhúng cho tất cả các sản phẩm và một mô hình để xếp hạng độ tương quan của mỗi sản phẩm đối với một truy vấn. Mô hình nhúng có thể cần được cập nhật ít thường xuyên hơn so với mô hình xếp hạng. Vì đặc tính của các sản phẩm không thay đổi thường xuyên, có thể thoải mái với việc cập nhật nhúng của mình mỗi tuần, trong khi mô hình xếp hạng có thể cần được cập nhật mỗi ngày. Kịch bản tự động có thể trở nên phức tạp hơn nếu có sự phụ thuộc giữa các mô hình. Ví dụ, vì mô hình xếp hạng phụ thuộc vào nhúng, khi nhúng thay đổi, mô hình xếp hạng cũng nên được cập nhật.

Giai đoạn 3: Huấn luyện tự động, có trạng thái Ở giai đoạn 2, mỗi lần huấn luyện lại mô hình, huấn luyện nó từ đầu (huấn luyện không lưu trạng thái). Điều này làm cho quá trình huấn luyện trở nên đắt đỏ, đặc biệt là khi có tần suất huấn luyện cao.Vậy nên chúng ta cần phải

Thay đổi tư duy: Thừa nhận quy tắc của việc huấn luyện lại từ đầu và nhận ra lợi ích của việc huấn luyện lưu trạng thái. Cam kết với ý tưởng cập nhật mô hình bằng cách tải checkpoint trước đó vào bộ nhớ trước khi tiếp tục huấn luyện.

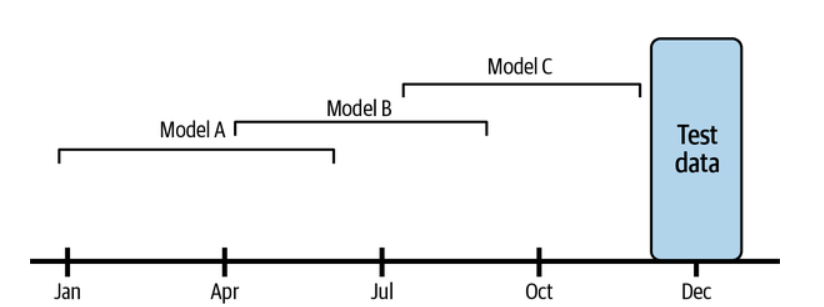
Điều chỉnh bản cập nhật: Sửa đổi kịch bản cập nhật tự động để tích hợp việc huấn luyện lưu trạng thái. Khi cập nhật mô hình được kích hoạt, định vị checkpoint trước đó và tải nó vào bộ nhớ. Tiếp tục huấn luyện trên checkpoint hiện tại, tích hợp dữ liệu mới mà không bắt đầu từ đầu.

Theo dõi dữ liệu gốc và mô hình: Thiết lập hệ thống để theo dõi nguyên gốc dữ liệu và mô hình. Theo dõi sự tiến triển của các mô hình theo thời gian, chỉ ra phiên bản mô hình nào đã phục vụ làm mô hình cơ sở và ghi lại dữ liệu được sử dụng cho mỗi cập nhật. Theo dõi này quan trọng để có khả năng tái tạo, gỡ lỗi và hiểu rõ về sự tiến triển của mô hình.

Xây dựng giải pháp nội bộ: Chú ý rằng các kho mô hình hiện tại có thể không đủ khả năng để theo dõi nguồn gốc mô hình. Xem xét việc xây dựng một giải pháp nội bộ để đáp ứng nhu cầu cụ thể của việc theo dõi phiên bản mô hình và dữ liệu liên quan của chúng.

Thách thức truy cập dữ liệu mới: Nếu chọn lựa dữ liệu mới từ các phương tiện truyền thông thời gian thực thay vì từ kho dữ liệu, giải quyết những thách thức được thảo luận trong phần "Thách thức truy cập dữ liệu mới". Đánh giá và có thể cần cải thiện đường ống truyền dữ liệu thời gian thực nếu nó chưa đủ chín muồi để lấy dữ liệu mới một cách hiệu quả.

Giai đoạn 4: Học liên tục Ở giai đoạn 3, các mô hình vẫn được cập nhật dựa trên một lịch trình cố định được đặt ra bởi các nhà phát triển. Tìm ra lịch trình tối ưu không phải là một công việc đơn giản và có thể phụ thuộc vào tình huống. Ví dụ, tuần trước, không có nhiều điều xảy ra trên thị trường, nên mô hình của không suy giảm nhanh chóng. Tuy nhiên, tuần này, nhiều sự kiện xảy ra, nên mô hình của suy giảm nhanh chóng hơn và yêu cầu một lịch trình huấn luyện nhanh hơn. Thay vì phụ thuộc vào một lịch trình cố định,có thể muốn mô hình của được tự động cập nhật mỗi khi phân phối dữ liệu thay đổi và hiệu suất của mô hình giảm đáng kể.



Cách để cập nhật mô hình thường xuyên:

Thực hiện thử nghiệm về giá trị của sự tươi mới của dữ liệu: Để đo giá trị này, có thể huấn luyện các phiên bản mô hình trên dữ liệu từ các khoảng thời gian khác nhau trong quá khứ và đánh giá chúng trên dữ liệu hiện tại. Chúng ta có thể so sánh mô hình phiên bản A (dữ liệu từ tháng 1 đến tháng 6 năm 2020), phiên bản B (dữ liệu từ tháng 4 đến tháng 9), và phiên bản C (dữ liệu từ tháng 6 đến tháng 11) trên dữ liệu từ tháng 12. Sự khác biệt trong hiệu suất của các phiên bản này sẽ cung cấp cho chúng ta cái nhìn về giá trị mà mô hình có thể đạt được từ dữ liệu tươi mới. Tính toán sự cải thiện trong hiệu suất: Nếu mô hình được huấn luyện trên dữ liệu từ cách đây một quý và kết quả kém hơn so với mô hình được huấn luyện trên dữ liệu từ một tháng trước, điều này cho thấy không nên chờ đợi một quý để cập nhật mô hình. Sự tươi mới của dữ liệu có giá trị. Tối ưu hóa tần suất cập nhật: Dựa trên giá trị của dữ liệu tươi mới, chúng ta có thể quyết định tần suất cập nhật mô hình. Ví dụ, nếu việc cập nhật hàng ngày cung cấp cải thiện đáng kể trong hiệu suất so với cập nhật hàng tháng,chúng ta nên xem xét việc chuyển sang cập nhật hàng ngày.

## Test Production

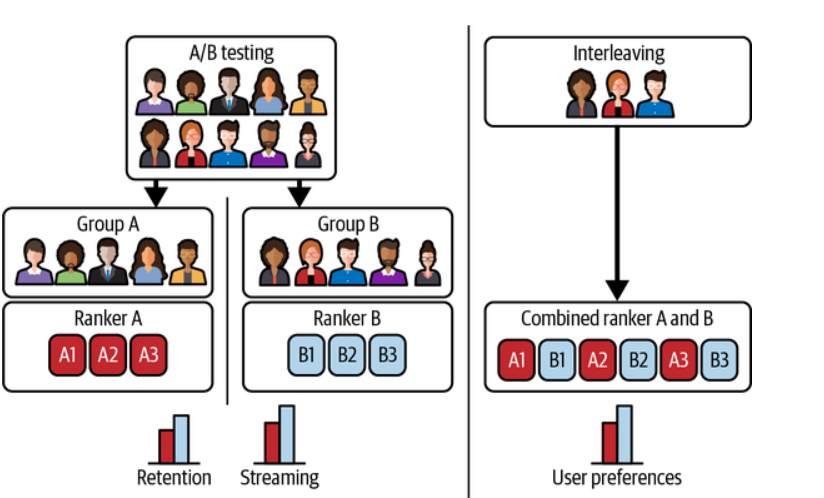
Nghiên cứu về học liên tục trong Machine Learning là một lĩnh vực quan trọng trong đào tạo các mô hình máy học để xử lý dữ liệu ngày càng phức tạp và thay đổi liên tục. Trong nhiều ứng dụng thực tế, mô hình máy học cần phải được triển khai và cập nhật thường xuyên để duy trì hiệu suất và sự thích nghi với dữ liệu mới. Thử nghiệm sản xuất (test production) là một phần quan trọng của quá trình này, giúp đảm bảo rằng các mô hình đang hoạt động trong môi trường sản xuất hoạt động đúng cách và đáp ứng các yêu cầu hiệu suất.

**2.1 Các cách kiểm tra:**

**\* Shadow Deployment (Triển khai ẩn):**

Shadow Deployment là một phương pháp an toàn để đánh giá mô hình mới mà không ảnh hưởng đến trải nghiệm của người dùng chính. Đây là cách hoạt động của nó: Triển khai Song Song: Mô hình mới được triển khai song song với mô hình hiện có trên cơ sở hạ tầng sản xuất. Điều này đảm bảo rằng mô hình mới hoạt động và sẵn sàng được sử dụng. Dự Đoán Bóng Bài: Mô hình mới không được phục vụ trực tiếp cho người dùng cuối. Thay vào đó, nó "dự đoán trong bóng bài" (shadow prediction), nghĩa là nó tạo ra các dự đoán nhưng không thực sự gửi chúng cho người dùng. So Sánh Hiệu Suất: Dữ liệu từ mô hình mới và mô hình hiện có được thu thập và so sánh để đánh giá hiệu suất của mô hình mới. Điều này cho phép xác định xem mô hình mới có cải thiện hay không. Sự An Toàn Cho Người Dùng: Người dùng chính không bị ảnh hưởng bởi mô hình mới và vẫn nhận được dự đoán từ mô hình hiện có, đảm bảo sự an toàn cho trải nghiệm của họ.

**\* A/B Testing (Kiểm Tra A/B):**



A/B Testing là một phương pháp phổ biến để so sánh hiệu suất giữa mô hình hiện có và mô hình mới. Đây là cách hoạt động của nó: Chia Lưu Lượng Truy Cập: Người dùng được chia thành hai nhóm ngẫu nhiên: Nhóm A và Nhóm B. Nhóm A tiếp tục nhận dự đoán từ mô hình hiện có, trong khi Nhóm B nhận dự đoán từ mô hình mới. Theo Dõi Kết Quả: Theo dõi sự tương tác và phản hồi từ cả hai nhóm. Điều này có thể bao gồm việc đo lường các chỉ số như tỷ lệ nhấp chuột, tỷ lệ chuyển đổi, hoặc các chỉ số liên quan khác đối với mục tiêu cụ thể. So Sánh Hiệu Suất: Dựa trên dữ liệu thu thập, so sánh hiệu suất giữa hai nhóm. Xác định xem mô hình mới có cải thiện hiệu suất so với mô hình hiện có hay không. Quyết Định Cuối Cùng: Dựa trên kết quả, quyết định liệu có nên triển khai mô hình mới cho toàn bộ người dùng hay không.

**\* Canary Release (Triển Khai Canary):**

Canary Release là một kỹ thuật để giảm rủi ro khi triển khai mô hình mới. Đây là cách hoạt động của nó: Triển Khai Bản Canary: Mô hình mới được triển khai cho một phần nhỏ và cụ thể của người dùng, gọi là "bản canary." Phần lớn người dùng vẫn tiếp tục sử dụng mô hình hiện có. Theo Dõi Sự Cải Thiện: Theo dõi hiệu suất của bản canary. Nếu nó hoạt động tốt và không gây ra lỗi nghiêm trọng, bạn có thể tiếp tục triển khai mô hình mới cho phần lớn người dùng. Giảm Rủi Ro: Triển khai bản canary giúp giảm rủi ro, vì chỉ một phần nhỏ người dùng bị ảnh hưởng nếu có lỗi.

**\* (Thử Nghiệm Xen Kẽ):**

Interleaving Experiments là một cách để so sánh hai mô hình bằng cách giới thiệu gợi ý từ cả hai mô hình cho người dùng và theo dõi sự lựa chọn của họ. Đây là cách hoạt động của nó: Hiển Thị Gợi Ý Từ Cả Hai Mô Hình: Một nhóm người dùng nhận được gợi ý từ mô hình hiện có, trong khi nhóm khác nhận gợi ý từ mô hình mới. Theo Dõi Lựa Chọn: Theo dõi lựa chọn của người dùng để xác định xem mô hình mới có gợi ý tốt hơn hay không. Đánh Giá Hiệu Suất: Sự lựa chọn của người dùng cung cấp thông tin về hiệu suất của hai mô hình và giúp xác định mô hình nào hoạt động tốt hơn.

* + *Mỗi phương pháp có ưu điểm và hạn chế riêng, và việc lựa chọn phụ thuộc vào bối cảnh cụ thể của dự án và mục tiêu .Kết hợp các phương pháp này có thể giúp đảm bảo mô hình mới hoạt động tốt và đáp ứng được nhu cầu của người dùng.*

**2.2 Ý nghĩa :**

Đảm bảo Ổn Định Sản Phẩm: Khi triển khai mô hình trong môi trường sản xuất, điều quan trọng là đảm bảo rằng mô hình hoạt động ổn định và không gây ra các vấn đề không mong muốn như lỗi hoặc hiệu suất kém.

Đánh giá Học Liên Tục: Thử nghiệm sản xuất cung cấp cơ hội để đánh giá hiệu suất của mô hình máy học trong thời gian thực và xác định xem liệu mô hình có thể thích nghi và học từ dữ liệu mới không.

Phát hiện sớm sự cố: Bằng cách thực hiện thử nghiệm trong quá trình sản xuất, ta có thể phát hiện sớm các vấn đề hoặc biểu hiện của mô hình máy học và thực hiện các biện pháp sửa chữa hoặc điều chỉnh khi cần thiết.

**2.3 Quá trình thử nghiệm :**

Triển khai Mô hình: Bắt đầu bằng việc triển khai mô hình máy học trong môi trường sản xuất. Điều này bao gồm việc tích hợp mô hình vào hệ thống thực tế và bắt đầu sử dụng nó để đưa ra dự đoán.

Sử dụng Thực Tế: Mô hình sẽ được sử dụng để xử lý dữ liệu thực tế và đưa ra dự đoán. Dữ liệu đầu vào mới liên tục được đưa vào hệ thống.

Giám Sát Hiệu Suất: Các hệ thống giám sát được sử dụng để theo dõi hiệu suất của mô hình trong môi trường sản xuất. Điều này bao gồm việc ghi lại các metric hiệu suất và theo dõi sự thay đổi của chúng theo thời gian.

Thử nghiệm và Kiểm tra: Thử nghiệm và kiểm tra định kỳ được thực hiện để đảm bảo rằng mô hình hoạt động đúng cách và đáp ứng các tiêu chuẩn hiệu suất.

Cập Nhật và Điều Chỉnh: Nếu mô hình hiển thị bất kỳ dấu hiệu không bình thường hoặc hiệu suất không đạt yêu cầu, quy trình cập nhật và điều chỉnh sẽ được thực hiện. Điều này có thể bao gồm việc điều chỉnh siêu tham số của mô hình hoặc cập nhật dữ liệu đào tạo.

**2.4 Thách thức:**

Tích hợp liên tục: Cần xem xét cách triển khai và tích hợp liên tục các phiên bản cập nhật của mô hình vào môi trường sản xuất mà không gây gián đoạn quá lớn. Sử dụng lượng dữ liệu nhỏ: Trong học liên tục, có thể xảy ra tình huống khi lượng dữ liệu mới có hạn. Việc thử nghiệm trong điều kiện này đòi hỏi các phương pháp đặc biệt để ước tính hiệu suất mô hình. Quản lý Chi phí tính toán: Thực hiện thử nghiệm sản xuất có thể đòi hỏi nhiều tính toán, do đó, quản lý chi phí tính toán là một yếu tố quan trọng.

**2.5 Kết Luận:**

Thử nghiệm sản xuất trong nghiên cứu học liên tục trong Machine Learning là một phần quan trọng của quá trình triển khai và duy trì các mô hình máy học trong môi trường sản xuất. Nó giúp đảm bảo rằng các mô hình đáp ứng yêu cầu hiệu suất và có khả năng thích nghi với dữ liệu mới. Tuy có thách thức, nhưng việc thực hiện thử nghiệm sản xuất là cần thiết để đảm bảo tính ổn định và hiệu suất của các mô hình máy học trong thực tế.

# TÀI LIỆU THAM KHẢO

Tiếng Việt

[Trần Trung Trực (2020) Optimizer- Hiểu sâu về các thuật toán tối ưu](https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8)

[Tiep Vu Huu (2017) Bài 7: Gradient Descent (phần 1/2)](https://machinelearningcoban.com/2017/01/12/gradientdescent/)

[NEOS VIETNAM(2021) Gradient Descent trong trí tuệ nhân tạo](https://blog.neoscorp.vn/gradient-descent-trong-tri-tue-nhan-tao/)

[Đoàn Võ Duy Thanh, Nguyễn Văn Cường, Lê Khắc Hồng Phúc, Phạm Hồng Vinh ,Phạm Minh Đức (2021) Đắm mình vào học sâu](https://d2l.aivivn.com/chapter_optimization/index_vn.html)

Tiếng Anh

[Designing Machine Learning Systems( 2022) PUBLISHED BY: O'Reilly Media, Inc.](https://learning.oreilly.com/library/view/designing-machine-learning/9781098107956/ch09.html#why_continual_learningquestion_mark)