

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN



**BÁO CÁO CUỐI KỲ**  
**MÔN: NHẬP MÔN HỌC MÁY**

*Người hướng dẫn:* TS LÊ ANH CƯỜNG

*Người thực hiện:* NGUYỄN HOÀNG THÔNG - 52000861

Lớp : 20050401

Khoá : 24

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**  
**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**  
**MÔN: NHẬP MÔN HỌC MÁY**

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

*Người thực hiện:* **NGUYỄN HOÀNG THÔNG - 52000861**

**Lớp : 20050401**

**Khoá : 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

## LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn rất nhiều toàn thể quý thầy cô, giảng viên và viên chức tại trường Đại Học Tôn Đức Thắng, đặc biệt là các thầy cô ở Khoa Công nghệ thông tin đã dành thời gian và nỗ lực để giúp đỡ chúng em trong quá trình học tập và thực hiện các bài tập, đồ án.

Chúng em muốn bày tỏ sự tri ân đặc biệt đến thầy Lê Anh Cường - giảng viên của Khoa Công nghệ thông tin, đã tận tình truyền đạt kiến thức và hướng dẫn chúng em một cách nghiêm túc và tận tâm. Nhờ sự giảng dạy và hướng dẫn của thầy, chúng em đã có thể tiếp cận với những kiến thức mới và hoàn thành các bài tập, đồ án một cách tốt nhất.

Tuy nhiên, chúng em cũng nhận thấy rằng còn nhiều điều chúng em cần phải học hỏi và cải thiện trong quá trình học tập và làm việc. Chúng em mong muốn nhận được ý kiến đóng góp và gợi ý từ các thầy cô để chúng em có thể phát triển kỹ năng và nâng cao trình độ trong tương lai.

Một lần nữa, chúng em xin gửi lời cảm ơn sâu sắc tới toàn thể quý thầy cô và hy vọng rằng các thầy cô sẽ tiếp tục truyền cảm hứng và hướng dẫn cho chúng em và các sinh viên khác trong những năm tiếp theo. Chúc các thầy cô có nhiều sức khỏe và thành công trong công việc.

## **BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là báo cáo của riêng chúng tôi và được sự hướng dẫn khoa học của thầy Lê Anh Cường. Các nội dung báo cáo, kết quả trong đề tài này là trung thực. 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 báo cáo 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 báo cáo 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 23 tháng 11 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Nguyễn Hoàng Thông*

## MỤC LỤC

1. Giới Thiệu về Bộ Tối Ưu Hóa
1.1. Tổng Quan về Bộ Tối Ưu Hóa
1.2. Cách Thức Hoạt Động của Bộ Tối Ưu Hóa
1.3. Các Bộ Tối Ưu Hóa Phổ Biến
1.3.1. Gradient Descent
1.3.2. Stochastic Gradient Descent
1.3.3. MiniBatch Gradient Descent
1.3.4. Adam
1.3.5. RMSprop
1.3.6. AdaGrad
1.4. Các Khía Cạnh Khác Cần Xem Xét
1.4.1. Chọn Bộ Tối Ưu Hóa Phù Hợp
1.4.2. Kỹ Thuật Tối Ưu Hóa Ngoài Bộ Tối Ưu Hóa Truyền Thống
1.4.3. So Sánh Các Bộ Tối Ưu Hóa
2. Học Liên Tục Và Kiểm Thử Sản Phẩm
2.1. Giới Thiệu về Học Liên Tục
2.1.1 Stateless Retraining so với Stateful Training
2.1.2 Tại sao phải học liên tục?
2.1.3 Những thách thức học liên tục
2.1.4 Bốn giai đoạn học liên tục
2.1.5 Tần suất cập nhật mô hình của bạn
2.2. Giới Thiệu Test Production
2.2.1 Triển khai Shadow
2.2.2 A/B Testing
2.2.3 Canary Release

2.2.4 Thí Nghiệm Xen Kẽ

2.2.5 Thuật toán Bandits

2.2.6 Kết luận

TÀI LIỆU THAM KHẢO

## **DANH MỤC HÌNH ẢNH**

1. Minh Họa Bộ Tối Ưu Hóa
2. Phương Trình Hồi Quy Tuyến Tính
3. Đồ Thị Phương Trình Hồi Quy Tuyến Tính
4. Minh Họa Tốc Độ Học
5. So Sánh Các Bộ Tối Ưu Hóa
6. Sự Đơn Giản Hóa Quá Trình Học Liên Tục
7. Stateless Retraining so với Stateful Training
8. Lấy dữ liệu thời gian thực
9. Đơn giản hóa quá trình trích gán nhãn
10. Kiểm tra hiệu suất với dữ liệu khác nhau xem hiệu suất thay đổi
11. Một minh họa về thử nghiệm xen kẽ so với thử nghiệm A / B
12. Xen kẽ các đề xuất video từ hai thuật toán xếp hạng bằng cách sử dụng bản nháp nhóm

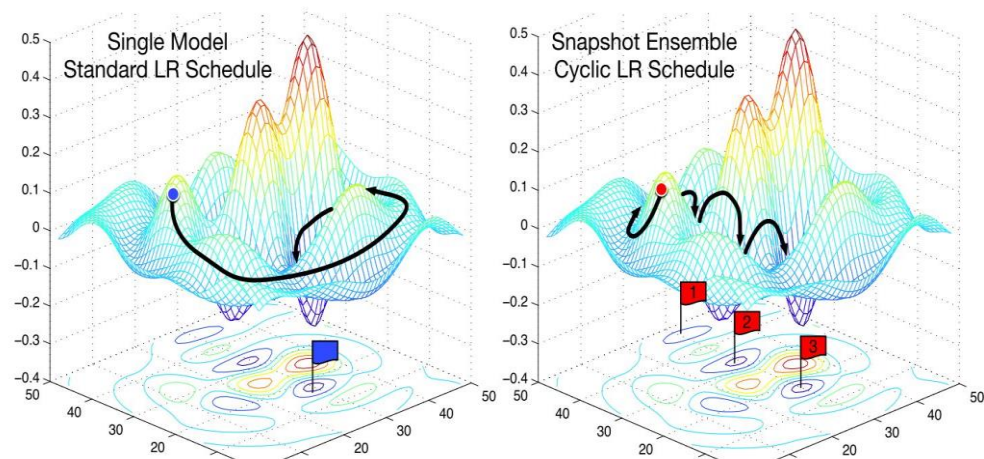
# 1. BÀI 1

## 1.1. GIỚI THIỆU CHUNG VỀ OPTIMIZER

Trong lĩnh vực học máy, mục tiêu chính là xây dựng các mô hình có khả năng hoạt động tốt và đưa ra dự đoán chính xác trong các tình huống cụ thể. Đạt được mục tiêu này phụ thuộc vào quá trình tối ưu hóa, một khía cạnh quan trọng trong học máy.

Tối ưu hóa trong học máy là quá trình điều chỉnh các siêu tham số của mô hình để giảm thiểu hàm mất mát (loss function). Hàm mất mát quan trọng vì nó đo lường sự khác biệt giữa giá trị thực tế và giá trị dự đoán của mô hình. Tối ưu hóa đóng vai trò cải thiện độ chính xác và hiệu suất của mô hình.

Trong quá trình đào tạo, bằng các kỹ thuật và phương pháp toán học... các tham số mô hình được điều chỉnh để giảm thiểu hàm mất mát và cải thiện dự đoán. Quá trình này bao gồm việc sử dụng các trình tối ưu hóa (Optimizer), kết hợp hàm mất mát và tham số mô hình để cập nhật mô hình theo kết quả của hàm mất mát. Trình tối ưu hóa hướng dẫn cách điều chỉnh, thông báo cho optimizer khi mô hình đang tiến bộ hoặc cần điều chỉnh.



Hình 1. Minh họa trình tối ưu hóa

Quá trình tối ưu hóa trong học máy có thể được mô tả như việc điều hướng qua một cảnh quan phức tạp, với mỗi đỉnh và thung lũng tượng trưng cho các cấu hình



tham số khác nhau của mô hình. Trình tối ưu hóa nỗ lực tìm kiếm mức tối thiểu toàn cầu - cấu hình tham số tối ưu nhất, nơi mô hình đạt hiệu suất cao nhất và lỗi thấp nhất.

## 1.2. TRÌNH TỐI ƯU HÓA HOẠT ĐỘNG THỂ NÀO?

Trình tối ưu hóa trong học máy hoạt động bằng cách điều chỉnh các tham số của mô hình qua từng lần lặp để giảm thiểu lỗi hoặc tăng cường hiệu suất của thuật toán học. Nó sử dụng một thuật toán tối ưu hóa cụ thể để duyệt qua không gian tham số và tìm ra cấu hình tối ưu cho mô hình.

Khi bắt đầu quá trình huấn luyện, mô hình được khởi tạo với các giá trị tham số ngẫu nhiên. Trình tối ưu hóa đánh giá hiệu suất hiện tại của mô hình bằng cách tính toán loss-function, phản ánh sai số giữa giá trị dự đoán và giá trị thực.

Dựa trên tổn thất tính toán được, trình tối ưu hóa xác định hướng và mức độ cần thiết để cập nhật các tham số. Nó thực hiện những điều chỉnh nhỏ đến trọng số và độ lệch của mô hình để giảm thiểu tổn thất sau mỗi lần lặp. Quá trình này tiếp tục cho đến khi mô hình hội tụ về giải pháp tối ưu, nơi tổn thất được giảm thiểu và mô hình đạt hiệu suất mong muốn.

Trình tối ưu hóa sử dụng khái niệm gradient, là đạo hàm của loss-function đối với các tham số. Bằng cách tính toán gradient, trình tối ưu hóa xác định hướng giảm của loss-function và cập nhật các tham số theo hướng đó. Đây chính là thuật toán giảm gradient.

Có nhiều biến thể của trình tối ưu hóa dựa trên gradient, như Stochastic Gradient Descent (SGD), Adam, RMSprop, và AdaGrad, mỗi loại áp dụng chiến lược khác nhau trong việc điều chỉnh các tham số. SGD chọn một tập hợp con ngẫu nhiên của dữ liệu huấn luyện trong mỗi lần lặp, giúp tiết kiệm tài nguyên tính toán. Adam kết hợp ưu điểm của Ước lượng Moment Thích Ứng và RMSprop để cải thiện quá trình hội tụ. RMSprop điều chỉnh tốc độ học cho từng tham số dựa trên trung bình di động của gradient bình phương. AdaGrad điều chỉnh tốc độ học dựa trên lịch sử của gradient cho từng tham số.

Bằng cách cập nhật hiệu quả các tham số thông qua các thuật toán tối ưu hóa này, trình tối ưu hóa giúp mô hình học từ dữ liệu đào tạo và áp dụng vào dữ liệu chưa biết. Nó đóng vai trò quan trọng trong việc thành công của các mô hình học máy, tinh chỉnh các tham số và tối ưu hóa hiệu suất.

### 1.3. CÁC TRÌNH TỐI ƯU HÓA PHỔ BIẾN

Trong lĩnh vực học máy, có một số trình tối ưu hóa phổ biến được sử dụng rộng rãi để huấn luyện và tối ưu hóa hiệu suất của các mô hình. Mỗi trình tối ưu hóa có cách tiếp cận và chiến lược riêng trong việc điều chỉnh tham số của mô hình. Dưới đây là một số trình tối ưu hóa thông dụng:

#### 1.3.1 *Gradient Descent*

Gradient Descent được biết đến là một trong những thuật toán tối ưu hóa được sử dụng phổ biến nhất để đào tạo các mô hình học máy bằng cách giảm thiểu lỗi giữa kết quả thực tế và mong đợi. Hơn nữa, gradient descent cũng được sử dụng để đào tạo Neural Networks.

Có ba biến thể của gradient descent:

**Batch Gradient Descent:** Trong giảm độ dốc hàng loạt, các tham số của mô hình được cập nhật bằng cách sử dụng các gradient được tính toán trên toàn bộ tập dữ liệu đào tạo. Cách tiếp cận này cung cấp ước tính chính xác hơn về độ dốc nhưng có thể tốn kém về mặt tính toán, đặc biệt là đối với các bộ dữ liệu lớn.

**Stochastic Gradient Descent (SGD):** Trong giảm độ dốc ngẫu nhiên, các tham số của mô hình được cập nhật bằng cách sử dụng các gradient được tính trên một mẫu đào tạo duy nhất. Cách tiếp cận này hiệu quả về mặt tính toán nhưng giới thiệu nhiễu nhiều tiếng ồn hơn do lựa chọn ngẫu nhiên các mẫu đào tạo.

**Mini-batch Gradient Descent:** Mini-batch gradient descent là sự thỏa hiệp giữa gradient descent hàng loạt và stochastic gradient descent. Theo cách tiếp cận này, các tham số của mô hình được cập nhật bằng cách sử dụng các gradient được tính toán trên một lô nhỏ các mẫu đào tạo. Nó cung cấp sự cân bằng giữa độ chính xác và hiệu quả và là biến thể được sử dụng phổ biến nhất của gradient descent.

Trong thuật ngữ toán học, thuật toán Tối ưu hóa đề cập đến nhiệm vụ giảm thiểu / tối đa hóa một hàm khách quan  $f(x)$  được tham số hóa bởi  $x$ . Tương tự, trong học máy,

tối ưu hóa là nhiệm vụ giảm thiểu Cost-function được tham số hóa bởi các tham số của mô hình. Mục tiêu chính của gradient descent là giảm thiểu hàm lỗi bằng cách sử dụng lặp lại các bản cập nhật tham số. Khi các mô hình học máy này được tối ưu hóa, các mô hình này có thể được sử dụng làm công cụ mạnh mẽ cho Trí tuệ nhân tạo và các ứng dụng khoa học máy tính khác nhau..

Gradient Descent ban đầu được phát hiện bởi "**Augustin-Louis Cauchy**" vào giữa thế kỷ 18. ***Gradient Descent được định nghĩa là một trong những thuật toán tối ưu hóa lặp đi lặp lại được sử dụng phổ biến nhất của học máy để đào tạo các mô hình học máy và học sâu. Nó giúp tìm mức tối thiểu cục bộ của một hàm.***

Cách tốt nhất để xác định tối thiểu cục bộ hoặc cực đại cục bộ của một hàm bằng cách sử dụng gradient descent như sau:

- Nếu chúng ta di chuyển về phía một gradient âm hoặc ra khỏi gradient của hàm tại điểm hiện tại, nó sẽ cho mức **tối thiểu cục bộ** của hàm đó.
- Bất cứ khi nào chúng ta di chuyển về phía một gradient dương hoặc về phía gradient của hàm tại điểm hiện tại, chúng ta sẽ nhận được **cực đại cục bộ** của hàm đó.

Toàn bộ quy trình này được gọi là Gradient Ascent, còn được gọi là steepest descent. ***Mục tiêu chính của việc sử dụng thuật toán gradient descent là giảm thiểu Cost-function bằng cách sử dụng lặp lại.*** Để đạt được mục tiêu này, nó thực hiện hai bước lặp đi lặp lại:

- Tính đạo hàm bậc nhất của hàm để tính gradient hoặc độ dốc của hàm đó.
- Di chuyển ra khỏi hướng của gradient, có nghĩa là độ dốc tăng từ điểm hiện tại theo thời gian alpha, trong đó Alpha được định nghĩa là Tỷ lệ học tập. Nó là một tham số điều chỉnh trong quá trình tối ưu hóa giúp quyết định độ dài của các bước.

### **Cost-funtion là gì?**

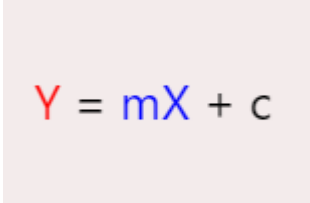
***Cost-function được định nghĩa là phép đo chênh lệch hoặc sai số giữa các giá trị thực tế và giá trị kỳ vọng tại vị trí hiện tại và hiện diện dưới dạng một số thực***

**duy nhất.** Nó giúp tăng và cải thiện hiệu quả học máy bằng cách cung cấp phản hồi cho mô hình này để nó có thể giảm thiểu lỗi và tìm mức tối thiểu cục bộ hoặc toàn cầu. Hơn nữa, nó liên tục lặp lại dọc theo hướng của gradient âm cho đến khi Cost-function gần bằng không. Tại điểm steepest descent này, mô hình sẽ ngừng học thêm. Mặc dù Cost-function và lost-funtion được coi là đồng nghĩa, nhưng cũng có một sự khác biệt nhỏ giữa chúng. Sự khác biệt nhỏ giữa lost-funtion và Cost-function là về lỗi trong quá trình đào tạo các mô hình học máy, vì lost-funtion đề cập đến lỗi của một ví dụ đào tạo, trong khi Cost-function tính toán lỗi trung bình trên toàn bộ tập huấn luyện.

Cost-function được tính toán sau khi đưa ra giả thuyết với các tham số ban đầu và sửa đổi các tham số này bằng cách sử dụng các thuật toán giảm độ dốc trên dữ liệu đã biết để giảm Cost-function.

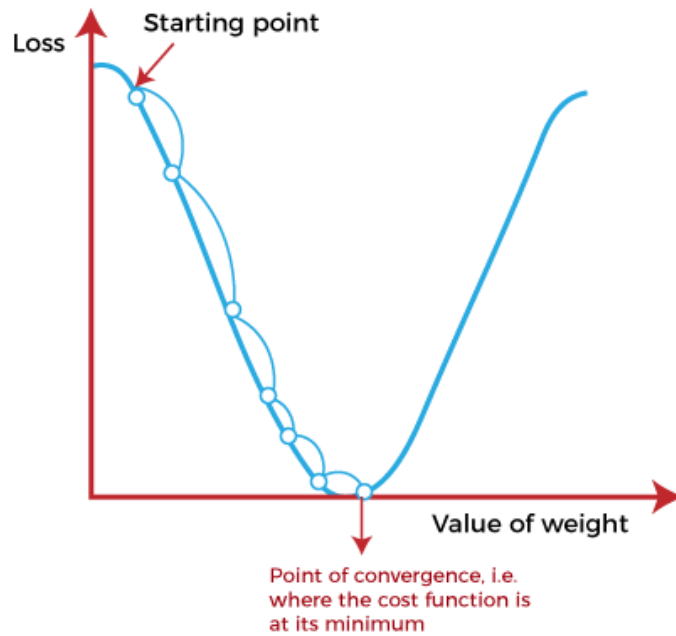
### **Gradient Descent hoạt động như thế nào?**

Trước khi bắt đầu nguyên lý làm việc của gradient descent, chúng ta nên biết một số khái niệm cơ bản để tìm ra độ dốc của một đường thẳng từ hồi quy tuyến tính. Phương trình hồi quy tuyến tính đơn giản được đưa ra như sau:


$$Y = mX + c$$

Hình 2. Phương trình hồi quy tuyến tính

Trong đó 'm' đại diện cho độ dốc của đường thẳng và 'c' đại diện cho các điểm chặn trên trục y. Gradient Descent in Machine Learning.



Hình 3. Đồ thị phương trình hồi quy tuyến tính

Điểm bắt đầu (thể hiện trong hình trên) được sử dụng để đánh giá hiệu suất vì nó được coi là một điểm tùy ý. Tại điểm bắt đầu này, chúng ta sẽ lấy đạo hàm hoặc độ dốc đầu tiên và sau đó sử dụng một đường tiếp tuyến để tính độ dốc của độ dốc này. Hơn nữa, độ dốc này sẽ thông báo cho các bản cập nhật cho các thông số (trọng lượng và độ lệch).

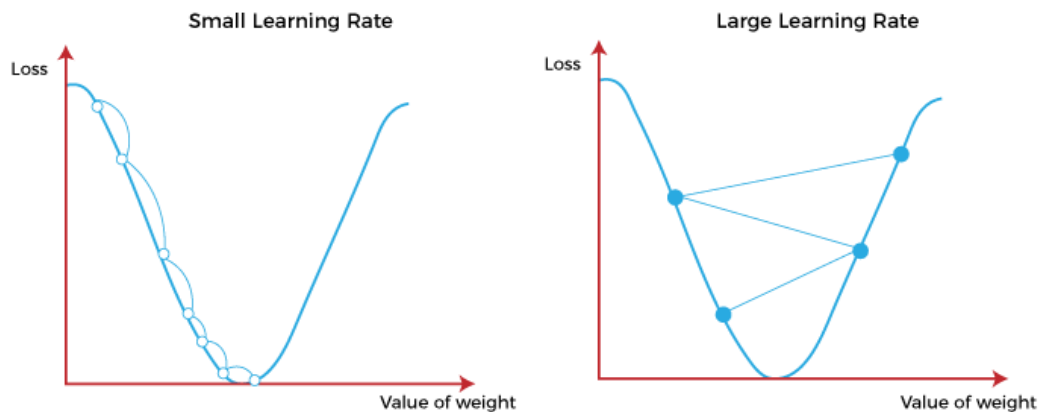
Độ dốc trở nên dốc hơn tại điểm bắt đầu hoặc điểm tùy ý, nhưng bất cứ khi nào các tham số mới được tạo ra, thì độ dốc giảm dần và tại điểm thấp nhất, nó tiếp cận điểm thấp nhất, được gọi là điểm hội tụ. Mục tiêu chính của gradient descent là giảm thiểu hàm chi phí hoặc sai số giữa dự kiến và thực tế. Để giảm thiểu hàm chi phí, cần có hai điểm dữ liệu:

### **Định hướng & Tỷ lệ học tập**

Hai yếu tố này được sử dụng để xác định tính toán đạo hàm từng phần của phép lặp trong tương lai và cho phép nó đến điểm hội tụ hoặc tối thiểu cục bộ hoặc tối thiểu toàn cầu. Hãy thảo luận ngắn gọn về các yếu tố tỷ lệ học tập;

### **Tỷ lệ học tập:**

Nó được định nghĩa là kích thước bước được thực hiện để đạt đến điểm tối thiểu hoặc thấp nhất. Đây thường là một giá trị nhỏ được đánh giá và cập nhật dựa trên hành vi của hàm chi phí. Nếu tỷ lệ học tập cao, nó dẫn đến các bước lớn hơn nhưng cũng dẫn đến nguy cơ vượt quá mức tối thiểu. Đồng thời, tỷ lệ học tập thấp cho thấy kích thước bước nhỏ, ảnh hưởng đến hiệu quả tổng thể nhưng mang lại lợi thế về độ chính xác cao hơn.



Hình 4. Tỷ lệ học tập

### 1.3.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) là một biến thể của thuật toán tối ưu hóa gradient descent cập nhật các tham số của mô hình học máy bằng cách sử dụng gradient được tính trên một mẫu đào tạo duy nhất tại một thời điểm. Không giống như gradient descent hàng loạt, tính toán và cập nhật các tham số bằng cách sử dụng gradient được tính toán trên toàn bộ tập dữ liệu đào tạo, SGD cung cấp một cách tiếp cận hiệu quả hơn về mặt tính toán.

Ý tưởng chính đằng sau SGD là bằng cách chọn ngẫu nhiên một mẫu đào tạo ở mỗi lần lặp, thuật toán không chỉ làm giảm gánh nặng tính toán mà còn giới thiệu mức độ ngẫu nhiên. Sự ngẫu nhiên này ngăn thuật toán bị mắc kẹt trong mức tối thiểu cục bộ và có thể dẫn đến sự hội tụ nhanh hơn.

SGD hoạt động bằng cách cập nhật lặp đi lặp lại các tham số của mô hình bằng cách sử dụng các gradient được tính toán trên mỗi mẫu đào tạo. Quy tắc cập nhật liên quan đến việc nhân tỷ lệ học tập với gradient và trừ nó khỏi giá trị tham số hiện tại.

Bản cập nhật này được thực hiện cho mỗi mẫu đào tạo và quá trình tiếp tục cho đến khi thuật toán đã đi qua tất cả các mẫu đào tạo.

Một lợi thế đáng kể của SGD là khả năng xử lý các bộ dữ liệu lớn. Vì chỉ có một mẫu đào tạo duy nhất được xem xét ở mỗi lần lặp, các yêu cầu bộ nhớ thấp hơn nhiều so với giảm độ dốc hàng loạt. Điều này làm cho SGD trở thành một lựa chọn thuận lợi khi xử lý các vấn đề dữ liệu lớn. Hơn nữa, SGD cho phép học trực tuyến, nơi dữ liệu mới có thể được tích hợp vào mô hình mà không cần đào tạo lại trên toàn bộ tập dữ liệu.

Tuy nhiên, SGD cũng có một số nhược điểm. Một nhược điểm là nó giới thiệu mức độ tiếng ồn cao hơn so với giảm độ dốc hàng loạt. Tiếng ồn này có thể làm cho quá trình hội tụ kém ổn định hơn, đặc biệt là khi xử lý các tính năng không có điều kiện hoặc tương quan cao. Ngoài ra, SGD có thể cần nhiều lần lặp lại hơn để hội tụ do lựa chọn ngẫu nhiên các mẫu đào tạo.

Để giảm thiểu những thách thức này, các kỹ thuật khác nhau có thể được áp dụng, chẳng hạn như sử dụng tỷ lệ học tập thích ứng, kết hợp động lượng và thực hiện phân rã tốc độ học tập. Những kỹ thuật này nhằm mục đích làm cho quá trình học tập ổn định và hiệu quả hơn.

Nhìn chung, Stochastic gradient Descent là một thuật toán tối ưu hóa mạnh mẽ giúp cân bằng hiệu quả tính toán với khả năng xử lý các tập dữ liệu lớn. Bằng cách cập nhật lặp đi lặp lại các tham số bằng cách sử dụng gradient được tính toán trên một mẫu đào tạo duy nhất, SGD cho phép hội tụ nhanh hơn và cho phép học trực tuyến trong các mô hình học máy.

### ***1.3.3 MiniBatch Gradient Descent***

Mini-batch Gradient Descent là một biến thể của thuật toán tối ưu hóa gradient descent kết hợp các ưu điểm của cả gradient descent hàng loạt và stochastic gradient descent. Trong giảm độ dốc hàng loạt nhỏ, các tham số của mô hình được cập nhật bằng cách sử dụng các gradient được tính toán trên một lô nhỏ các mẫu đào tạo ở mỗi lần lặp.

Kích thước lô nhỏ thường được chọn là giá trị vừa phải, chẳng hạn như 32, 64 hoặc 128. Kích thước này cho phép tính toán vector hóa hiệu quả trên phần cứng hiện

đại, tạo ra sự cân bằng giữa độ chính xác của độ dốc hàng loạt và hiệu quả tính toán của độ dốc ngẫu nhiên.

Quá trình giảm độ dốc hàng loạt nhỏ liên quan đến việc chia toàn bộ tập dữ liệu đào tạo thành các lô nhỏ, cung cấp từng lô nhỏ cho mô hình, tính toán độ dốc và cập nhật các tham số cho phù hợp. Quá trình này được lặp lại cho một số lần lặp lại xác định hoặc cho đến khi đạt được sự hội tụ.

Việc sử dụng các lô nhỏ mang lại một số lợi thế. Thứ nhất, nó làm giảm gánh nặng tính toán so với gradient descent hàng loạt bằng cách chỉ xử lý một phần nhỏ dữ liệu đào tạo ở mỗi lần lặp. Điều này làm cho việc giảm độ dốc hàng loạt nhỏ trở nên khả thi hơn đối với các bộ dữ liệu lớn không thể vừa với bộ nhớ.

Thứ hai, giảm độ dốc hàng loạt nhỏ giới thiệu một mức độ nhiễu do lựa chọn ngẫu nhiên các mẫu đào tạo trong mỗi lô nhỏ. Tiếng ồn này giúp thuật toán thoát khỏi mức tối thiểu cục bộ và đạt được mức tối ưu tốt hơn, cải thiện khả năng khái quát hóa của mô hình.

Hơn nữa, độ dốc mini-batch gradient descent cung cấp sự hội tụ tốt hơn so với giảm độ dốc ngẫu nhiên. Bằng cách sử dụng một lô nhỏ gồm nhiều mẫu, các bản cập nhật tham số ổn định hơn, dẫn đến các bản cập nhật ít nhiễu hơn và hội tụ mượt mà hơn.

Tuy nhiên, hậu duệ gradient hàng loạt mini cũng có một số cân nhắc. Chọn kích thước lô nhỏ thích hợp là rất quan trọng, vì kích thước quá nhỏ có thể dẫn đến hội tụ chậm hơn, trong khi kích thước quá lớn có thể làm giảm hiệu ứng tiếng ồn và cản trở việc thoát khỏi mức tối thiểu cục bộ.

Một yếu tố khác cần xem xét là tỷ lệ học tập. Khi các đợt nhỏ giới thiệu tiếng ồn, việc điều chỉnh tốc độ học tập có thể có lợi. Các kỹ thuật như giảm tốc độ học tập và tỷ lệ học tập thích ứng, chẳng hạn như trình tối ưu hóa RMSprop hoặc Adam, có thể giúp duy trì tốc độ học tập tối ưu trong suốt quá trình đào tạo.

Nhìn chung, mini-batch gradient descent kết hợp các lợi ích của gradient descent batch và stochastic gradient descent. Nó cung cấp một cách tiếp cận hiệu quả



tính toán trong khi cải thiện khả năng hội tụ và khái quát hóa. Bằng cách xử lý các lô mẫu đào tạo nhỏ, biến thể giảm độ dốc này cho phép các mô hình học máy tối ưu hóa các tham số của chúng một cách hiệu quả.

### ***1.3.4 Adam***

Adam, viết tắt của Adaptive Moment Estimation, là một thuật toán tối ưu hóa thường được sử dụng trong học máy. Nó kết hợp các ưu điểm của hai phương pháp tối ưu hóa khác, Thuật toán chuyển màu thích ứng (AdaGrad) và Truyền bình phương trung bình gốc (RMSprop), để cung cấp một cách tiếp cận hiệu quả và hiệu quả để cập nhật các tham số của mô hình.

Trình tối ưu hóa Adam điều chỉnh tốc độ học tập cho từng tham số dựa trên các khoảng khắc bậc nhất (giá trị trung bình) và khoảng khắc bậc hai (phương sai không căn giữa) của các gradient. Nó tính toán trung bình phân rã theo cấp số nhân của các gradient trong quá khứ và các giá trị bình phương của chúng, ước tính hiệu quả khoảng khắc đầu tiên (trung bình) và khoảng khắc thứ hai (phương sai) của gradient.

Thuật toán sử dụng ước tính giá trị trung bình và phương sai này để cập nhật các tham số. Nó kết hợp các lợi ích của AdaGrad, giúp mở rộng tỷ lệ học tập dựa trên độ dốc lịch sử và RMSprop, điều chỉnh tốc độ học tập một cách thích ứng dựa trên đường trung bình động của các gradient bình phương.

Ưu điểm chính của Adam là khả năng xử lý các gradient thưa thớt và các mục tiêu không cố định. Bằng cách duy trì tốc độ học tập riêng biệt cho từng tham số, nó có thể cập nhật thích ứng các tham số dựa trên mức độ liên quan của chúng với quá trình tối ưu hóa. Điều này làm cho Adam đặc biệt hiệu quả trong các tình huống mà các tham số khác nhau có thể có các yêu cầu cập nhật khác nhau.

Ngoài ra, Adam giúp khắc phục một số hạn chế của các thuật toán tối ưu hóa khác. Nó thể hiện các đặc tính hội tụ tốt và mạnh mẽ đến các gradient ồn ào hoặc thưa thớt. Việc điều chỉnh tốc độ học tập thích ứng cho phép Adam hội tụ nhanh chóng và đạt được hiệu suất tốt trên một loạt các tác vụ học máy.

Một tính năng quan trọng khác của Adam là bước điều chỉnh thiên vị. Vì Adam tính toán khoảng khắc đầu tiên và thứ hai của gradient bằng cách sử dụng đường trung bình động hàm mũ, các ước tính có xu hướng về không, đặc biệt là trong giai đoạn đầu đào tạo. Để sửa chữa sai lệch này, Adam áp dụng một hệ số hiệu chỉnh cho các ước tính được sửa sai lệch. Điều này giúp cải thiện độ chính xác của ước tính thời điểm và nâng cao hơn nữa hiệu suất của trình tối ưu hóa.

Tóm lại, Adam là một thuật toán tối ưu hóa thích ứng kết hợp các lợi ích của AdaGrad và RMSprop. Bằng cách điều chỉnh tốc độ học tập dựa trên khoảng khắc đầu tiên và thứ hai của độ dốc, Adam cập nhật hiệu quả các thông số và đạt được sự hội tụ tốt. Nó được sử dụng rộng rãi trong học sâu và các tác vụ học máy khác do tính hiệu quả, mạnh mẽ và khả năng xử lý các mục tiêu không cố định.

### ***1.3.5 RMSprop***

RMSprop, viết tắt của Root Mean Square Propagation, là một thuật toán tối ưu hóa thường được sử dụng trong học máy. Nó được thiết kế để giải quyết một số hạn chế của các phương pháp tối ưu hóa dựa trên gradient truyền thống bằng cách điều chỉnh tốc độ học tập cho từng tham số dựa trên độ dốc lịch sử.

Mục tiêu chính của RMSprop là tăng tốc độ hội tụ bằng cách điều chỉnh tốc độ học tập dựa trên đường trung bình động của các gradient bình phương. Nó chia tốc độ học tập cho căn bậc hai của trung bình phân rã theo cấp số nhân của các gradient bình phương. Sự chuẩn hóa này ngăn tốc độ học tập trở nên quá lớn khi độ dốc luôn lớn và ngược lại khi độ dốc luôn nhỏ.

Trực giác đằng sau RMSprop là nó điều chỉnh tốc độ học tập khác nhau cho từng tham số dựa trên lịch sử gradient gần đây của chúng. Độ dốc có cường độ lớn hơn sẽ dẫn đến tỷ lệ học tập nhỏ hơn, trong khi độ dốc có cường độ nhỏ hơn sẽ dẫn đến tỷ lệ học tập lớn hơn. Sự điều chỉnh thích ứng của tốc độ học tập này giúp cải thiện sự hội tụ và giảm bớt vấn đề dao động hoặc phân kỳ có thể xảy ra với tốc độ học tập cố định.

Ngoài ra, RMSprop có hiệu quả trong việc xử lý dữ liệu thưa thớt hoặc các mục tiêu không cố định. Bằng cách duy trì đường trung bình động của các gradient bình phương, nó điều chỉnh tốc độ học tập theo đặc điểm của độ dốc gặp phải trong quá trình đào tạo. Điều này làm cho RMSprop đặc biệt hữu ích trong các tình huống mà độ dốc khác nhau đáng kể giữa các tham số hoặc điểm dữ liệu khác nhau.

Một tính năng quan trọng của RMSprop là khả năng xử lý các tốc độ học tập khác nhau cho các thông số khác nhau. Nó cho phép thuật toán điều chỉnh tốc độ học tập riêng lẻ cho từng tham số, điều này có thể có lợi khi một số thông số nhất định yêu cầu cập nhật chậm hơn hoặc nhanh hơn so với các thông số khác.

Nhìn chung, RMSprop là một thuật toán tối ưu hóa mạnh mẽ điều chỉnh tốc độ học tập dựa trên độ dốc lịch sử. Nó cung cấp một cách tiếp cận hiệu quả và hiệu quả để đào tạo các mô hình học máy. Bằng cách chuẩn hóa tốc độ học tập dựa trên đường trung bình động của các gradient bình phương, RMSprop cho phép hội tụ tốt hơn, cải thiện độ ổn định và nâng cao hiệu suất trong các tác vụ học máy khác nhau.

### **1.3.6 AdaGrad**

AdaGrad, viết tắt của Adaptive Gradient, là một thuật toán tối ưu hóa thường được sử dụng trong học máy. Nó được thiết kế để điều chỉnh thích ứng tốc độ học tập cho từng tham số dựa trên độ dốc lịch sử, làm cho nó đặc biệt hữu ích cho dữ liệu thưa thớt hoặc các chức năng mục tiêu không tĩnh.

Ý tưởng chính đằng sau AdaGrad là chỉ định tốc độ học tập khác nhau cho mỗi tham số dựa trên độ lớn lịch sử của độ dốc. Nó bắt đầu bằng cách tích lũy tổng các gradient bình phương cho mỗi tham số theo thời gian. Sự tích lũy này hoạt động như một thước đo tần suất một tham số đã được cập nhật, mang lại nhiều trọng số hơn cho các tham số được cập nhật không thường xuyên và ít trọng lượng hơn cho các tham số được cập nhật thường xuyên.

AdaGrad đạt được sự điều chỉnh này bằng cách chia tỷ lệ học tập cho căn bậc hai của tổng gradient bình phương tích lũy. Các thông số có độ dốc lớn hơn sẽ có tốc độ học tập nhỏ hơn, trong khi các tham số có độ dốc nhỏ hơn sẽ có tốc độ học tập lớn

hơn. Cách tiếp cận này cho phép AdaGrad thay đổi thích ứng tốc độ học tập cho từng tham số, giải quyết vấn đề gradient biến mất hoặc nổ.

Một lợi thế chính của AdaGrad là khả năng xử lý dữ liệu thưa thớt. Trong nhiều tác vụ học máy, dữ liệu đầu vào thưa thớt, có nghĩa là chỉ một phần nhỏ các tính năng hoặc đầu vào có giá trị khác không. Các thuật toán tối ưu hóa truyền thống xử lý tất cả các kích thước như nhau có thể gán các bản cập nhật quá mức cho các thứ nguyên khác không, dẫn đến hiệu suất kém. AdaGrad giảm thiểu vấn đề này bằng cách mở rộng tỷ lệ học tỷ lệ nghịch với căn bậc hai của tổng gradient bình phương tích lũy, nhấn mạnh hơn vào các tính năng hoặc đầu vào hiếm khi xảy ra.

Tuy nhiên, một nhược điểm tiềm năng của AdaGrad là tốc độ học tập có xu hướng trở nên quá nhỏ khi sự tích lũy của gradient bình phương tiếp tục. Điều này có thể cản trở việc học trong các giai đoạn đào tạo sau này. Để giải quyết vấn đề này, các thuật toán tối ưu hóa khác như RMSprop và Adam đã được phát triển, sử dụng các chiến lược khác nhau để điều chỉnh tốc độ học tập.

Tóm lại, AdaGrad là một thuật toán tối ưu hóa điều chỉnh tốc độ học tập cho từng tham số dựa trên độ dốc lịch sử. Nó cung cấp một cách tiếp cận thích ứng để xử lý dữ liệu thưa thớt và các chức năng khách quan không cố định. Mặc dù có những hạn chế, AdaGrad đóng vai trò là cột mốc quan trọng trong việc phát triển các thuật toán tối ưu hóa và đã mở đường cho những tiến bộ hơn nữa trong lĩnh vực này.

## **1.4. NHIỀU HƠN**

### ***1.4.1. Chọn trình tối ưu hóa phù hợp***

Chọn trình tối ưu hóa phù hợp là một quyết định quan trọng trong học máy vì nó có thể ảnh hưởng đáng kể đến quá trình đào tạo và hiệu suất của mô hình. Các trình tối ưu hóa khác nhau có điểm mạnh và điểm yếu của chúng và có thể phù hợp hơn cho các tác vụ hoặc bộ dữ liệu cụ thể. Dưới đây là một số yếu tố chính cần xem xét khi chọn trình tối ưu hóa:

#### ***1. Tập dữ liệu và vấn đề***

Các đặc điểm của tập dữ liệu và vấn đề của bạn đóng một vai trò quan trọng trong việc xác định trình tối ưu hóa phù hợp nhất. Xem xét kích thước của tập dữ liệu, độ phức tạp của dữ liệu và bản chất của vấn đề bạn đang cố gắng giải quyết. Đối với các bộ dữ liệu nhỏ hơn, các trình tối ưu hóa dựa trên gradient truyền thống như Gradient Descent hoặc các biến thể của nó có thể hoạt động tốt. Đối với các bộ dữ liệu lớn hơn, các thuật toán tối ưu hóa ngẫu nhiên như Stochastic Gradient Descent (SGD) hoặc các trình tối ưu hóa thích ứng như Adam hoặc RMSprop có thể hiệu quả hơn.

## 2. Kiến trúc mô hình

Cấu trúc và độ phức tạp của mô hình của bạn có thể ảnh hưởng đến lựa chọn trình tối ưu hóa. Mạng nơ-ron sâu với nhiều lớp và thông số thường được hưởng lợi từ các trình tối ưu hóa thích ứng như Adam hoặc RMSprop. Những trình tối ưu hóa này có thể giúp điều hướng không gian tham số hiệu quả hơn và tăng tốc độ hội tụ. Đối với các mô hình đơn giản hơn, các trình tối ưu hóa đơn giản hơn như Gradient Descent hoặc SGD có thể đủ.

## 3. Tốc độ hội tụ

Nếu bạn có thời gian hoặc tài nguyên tính toán hạn chế, tốc độ hội tụ của trình tối ưu hóa trở nên quan trọng. Các trình tối ưu hóa thích ứng như Adam và RMSprop thường hội tụ nhanh hơn so với các phương pháp truyền thống như Gradient Descent hoặc SGD. Tuy nhiên, cần lưu ý rằng sự hội tụ nhanh hơn không phải lúc nào cũng đảm bảo hiệu suất tốt hơn, vì vậy điều quan trọng là phải đạt được sự cân bằng giữa tốc độ và độ chính xác.

## 4. Khả năng chịu tiếng ồn

Một số trình tối ưu hóa, như SGD, giới thiệu nhiều cho quá trình tối ưu hóa do tính chất ngẫu nhiên của chúng. Tiếng ồn này có thể hoạt động như một bộ điều chỉnh, ngăn ngừa quá mức và cải thiện khái quát hóa. Nếu bạn có một mô hình phức tạp hoặc một lượng dữ liệu đào tạo hạn chế, sử dụng trình tối ưu hóa giới thiệu tiếng ồn, như SGD, có thể có lợi.

## 5. Thử nghiệm

Cuối cùng, việc lựa chọn trình tối ưu hóa có thể yêu cầu thử nghiệm và đánh giá thực nghiệm. Người ta thường thử các trình tối ưu hóa khác nhau và so sánh hiệu suất của chúng trên một bộ xác thực. Điều này cho phép bạn đánh giá trình tối ưu hóa nào hoạt động tốt nhất cho sự cố và tập dữ liệu cụ thể của bạn.

Hãy nhớ rằng việc lựa chọn trình tối ưu hóa không cố định và có thể được đánh giá lại và thay đổi trong quá trình đào tạo. Các trình tối ưu hóa khác nhau có thể có tác dụng khác nhau ở các giai đoạn đào tạo khác nhau, vì vậy điều quan trọng là phải theo dõi hiệu suất và điều chỉnh nếu cần thiết.

Bằng cách xem xét các yếu tố này và đánh giá sự đánh đổi, bạn có thể đưa ra quyết định sáng suốt về trình tối ưu hóa phù hợp nhất cho nhiệm vụ học máy của bạn.

### ***1.4.2 Kỹ thuật tối ưu hóa: Ngoài các trình tối ưu hóa truyền thống***

Trong khi các trình tối ưu hóa truyền thống như Gradient Descent, Stochastic Gradient Descent (SGD), Adam, RMSprop và AdaGrad được sử dụng rộng rãi trong học máy, có một số kỹ thuật tối ưu hóa vượt xa các phương pháp truyền thống này. Những kỹ thuật này nhằm mục đích cải thiện hơn nữa hiệu suất, tốc độ hội tụ và độ mạnh mẽ của các mô hình học máy. Dưới đây là một vài kỹ thuật tối ưu hóa đáng chú ý:

#### **1. Tối ưu hóa động lượng**

Tối ưu hóa động lượng giới thiệu khái niệm động lượng cho các bản cập nhật tham số. Nó tích lũy một vector vận tốc xác định hướng và tốc độ di chuyển qua không gian tham số. Kỹ thuật này giúp quá trình tối ưu hóa bằng cách tăng tốc hội tụ và giảm dao động, đặc biệt là ở những khu vực có độ cong cao hoặc độ dốc ồn ào.

#### **2. Độ dốc gia tốc Nesterov (NAG)**

Nesterov Accelerated Gradient (NAG) là một kỹ thuật tối ưu hóa giúp tăng cường tối ưu hóa động lượng. Nó điều chỉnh các cập nhật tham số bằng cách tính đến động lượng ở vị trí bước trước. Bằng cách xem xét động lượng, NAG cải thiện tốc độ hội tụ và cho phép cập nhật chính xác hơn so với tối ưu hóa động lượng truyền thống.

### 3. Adadelata

Adadelata là một kỹ thuật tối ưu hóa tỷ lệ học tập thích ứng nhằm khắc phục những hạn chế của AdaGrad. Không giống như AdaGrad, Adadelata tích lũy mức trung bình chạy của các gradient và các bản cập nhật. Nó tiếp tục bình thường hóa tỷ lệ học tập bằng cách chia các bản cập nhật tham số cho trung bình phân rã theo cấp số nhân của các bản cập nhật bình phương. Adadelata giúp tránh vấn đề tỷ lệ học tập giảm dần mà AdaGrad phải đối mặt, cải thiện tính ổn định và hội tụ.

### 4. RMSprop với Nesterov Momentum (RMSprop với NAG)

Kỹ thuật này là sự kết hợp giữa RMSprop và Nesterov Accelerated Gradient (NAG). Nó kết hợp tốc độ học tập thích ứng của RMSprop và khái niệm động lượng của NAG. Bằng cách sử dụng cả hai kỹ thuật này, RMSprop với NAG đạt được hiệu suất tối ưu hóa tốt hơn, hội tụ nhanh hơn và cải thiện độ ổn định so với chỉ sử dụng một trong hai kỹ thuật.

### 5. Tỷ lệ học tập theo chu kỳ

Tỷ lệ học tập theo chu kỳ (CLR) là một chính sách tỷ lệ học tập tự động điều chỉnh tỷ lệ học tập trong một phạm vi xác định trong quá trình đào tạo. Thay vì sử dụng một tỷ lệ học tập cố định, CLR khám phá một loạt các tỷ lệ học tập, định kỳ tăng và giảm tỷ lệ học tập theo các khoảng thời gian đã định. Việc khám phá theo chu kỳ này giúp mô hình thoát khỏi các điểm tối thiểu và yên ngựa cục bộ, dẫn đến cải thiện tối ưu hóa và khái quát hóa tiềm năng tốt hơn.

Những kỹ thuật tối ưu hóa này, cùng với nhiều kỹ thuật khác, liên tục được phát triển và nghiên cứu để cải thiện quá trình đào tạo các mô hình học máy. Tùy thuộc vào vấn đề, kiến trúc mô hình và tập dữ liệu, việc kết hợp các kỹ thuật nâng cao này có thể mang lại lợi ích bổ sung và nâng cao hơn nữa hiệu suất của các mô hình của bạn.

#### ***1.4.3 So sánh các trình tối ưu hóa***

Phương Pháp	Mục Đích	Đặc Điểm Chính	Phù Hợp Cho	Ưu Điểm	Nhược Điểm
Gradient Ascent (Steepest Descent)	Tối đa hóa hàm bằng cách di chuyển theo gradient dương	Hướng tới gradient dương; tìm cực đại cục bộ	Vấn đề cân tìm cực đại	Đơn giản, dễ hiểu	Có thể hội tụ tới cực đại cục bộ; phụ thuộc điều kiện ban đầu
Gradient Descent	Giảm thiểu Cost-function	Phổ biến trong ML, đặc biệt trong hồi quy	Phổ biến trong ML	Rộng rãi và hiệu quả	Có thể chậm trên dữ liệu lớn; mắc kẹt tại cực tiểu cục bộ

Stochastic Gradient Descent (SGD)	Cải thiện hiệu quả tính toán	Cập nhật tham số cho từng mẫu đào tạo	Dữ liệu lớn, cân hiệu quả tính toán	Hiệu quả tính toán	Cập nhật nhiều; hội tụ không ổn định
MiniBatch Gradient Descent	Kết hợp ưu điểm của GD và SGD	Cập nhật dựa trên lô nhỏ dữ liệu	Dữ liệu lớn, cân bằng độ chính xác và hiệu quả	Hiệu quả trên phân cứng hiện đại	Cần chỉnh kích thước lô
Adam	Cập nhật hiệu suất cao cho tham số mô hình	Hiệu quả với gradient thưa; thích nghi tỷ lệ học cho tham số	Dữ liệu thưa và yêu cầu cập nhật tham số khác nhau	Thích nghi tỷ lệ học; hiệu quả cho bài toán lớn	Phức tạp hơn GD; cân tính toán nhiều hơn mỗi vòng lặp

RMSprop	Tăng tốc độ hội tụ bằng cách điều chỉnh tỷ lệ học tập	Hiệu quả với dữ liệu thưa; chuẩn hóa tỷ lệ học tập	Dữ liệu thưa; mục tiêu không cố định	Ngăn chặn tỷ lệ học tập cập nhật mạnh	Có thể cần chỉnh tỷ lệ học tập; nhạy cảm với cài đặt ban đầu
AdaGrad	Điều chỉnh tỷ lệ học tập cho từng tham số	Tự động điều chỉnh tỷ lệ học tập	Dữ liệu thưa	Tự động thích nghi tỷ lệ học tập	Tỷ lệ học tập có thể giảm quá mức trong thời gian dài

Hình 5. So sánh các trình tối ưu hóa



### **1.4.3 Kết luận**

Có thể nói trình tối ưu hóa là thành phần thiết yếu trong học máy đóng vai trò quan trọng trong các mô hình đào tạo và cải thiện hiệu suất của chúng. Chọn trình tối ưu hóa phù hợp có thể ảnh hưởng lớn đến tốc độ hội tụ, độ ổn định và khái quát hóa của mô hình. Bằng cách hiểu các thuật toán tối ưu hóa khác nhau và đặc điểm của chúng, bạn có thể đưa ra quyết định sáng suốt để tối ưu hóa các mô hình học máy của mình. Cần lưu ý rằng việc lựa chọn trình tối ưu hóa không phải là cách tiếp cận một kích thước phù hợp với tất cả. Tùy thuộc vào vấn đề, tập dữ liệu và mô hình cụ thể, các trình tối ưu hóa khác nhau có thể mang lại kết quả khác nhau. Điều quan trọng là phải thử nghiệm với các trình tối ưu hóa khác nhau, theo dõi hiệu suất của chúng và tinh chỉnh lựa chọn dựa trên các quan sát thực nghiệm.

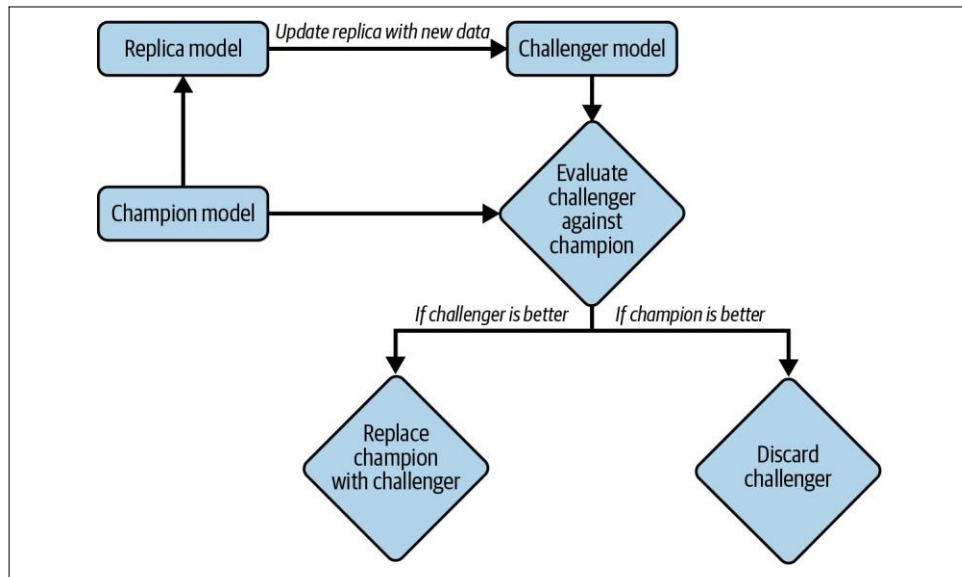
Tóm lại, tối ưu hóa là công cụ mạnh mẽ ảnh hưởng lớn đến hiệu suất của các mô hình học máy. Bằng cách hiểu các điểm mạnh và đặc điểm của các trình tối ưu hóa khác nhau và xem xét các yêu cầu cụ thể của nhiệm vụ, bạn có thể chọn trình tối ưu hóa phù hợp và tối ưu hóa các mô hình của mình để có hiệu suất và sự hội tụ tốt hơn.

## **2. BÀI 2**

### **2.1. GIỚI THIỆU CHUNG VỀ CONTINUAL LEARNING**

Khi nói về "học liên tục" trong học máy, nhiều người thường nghĩ đến mô hình tự cập nhật với mỗi mẫu dữ liệu mới trong quá trình sản xuất. Tuy nhiên, ít công ty thực sự áp dụng phương pháp này. Một lý do chính là nếu mô hình là mạng lưới thần kinh, việc học từ mỗi mẫu dữ liệu mới có thể dẫn đến "quên thảm khốc", nghĩa là mạng lưới thần kinh có thể nhanh chóng quên thông tin đã học trước đó khi tiếp nhận thông tin mới.

Một vấn đề khác là việc đào tạo như vậy có thể tốn kém, bởi vì hầu hết phần cứng hiện nay được thiết kế cho xử lý hàng loạt, không phải xử lý từng mẫu một. Các công ty thực hiện học liên tục thường cập nhật mô hình của họ trong các lô nhỏ. Mô hình cập nhật cần được đánh giá trước khi triển khai, và không nên thay đổi trực tiếp mô hình hiện tại mà thay vào đó là cập nhật trên bản sao của mô hình.

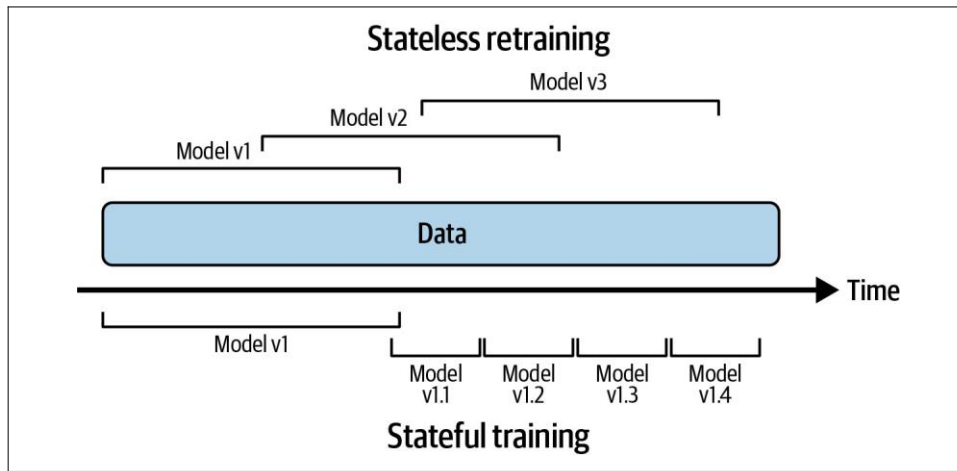


Hình 6. Đơn giản hóa quá mức

Thuật ngữ "học liên tục" thường khiến người ta nghĩ đến việc cập nhật mô hình học máy rất thường xuyên, ví dụ mỗi 5 hoặc 10 phút. Tuy nhiên, đa số các công ty không cần cập nhật mô hình liên tục đến mức đó. Lý do đầu tiên là họ không có đủ dữ liệu mới liên tục để việc cập nhật liên tục trở nên cần thiết. Lý do thứ hai là mô hình của họ không suy giảm chất lượng quá nhanh để cần cập nhật thường xuyên. Điều này cho thấy việc thay đổi lịch trình cập nhật từ hàng tuần sang hàng ngày có thể không mang lại lợi ích đáng kể nhưng lại gây ra chi phí cao và không cần thiết.

### 2.1.1 Stateless Retraining so với Stateful Training

Học liên tục không chỉ là về tần suất tái đào tạo, mà còn về cách mô hình được tái đào tạo. Hầu hết các công ty thực hiện tái đào tạo không lưu trạng thái (stateless retraining) - mô hình được đào tạo lại từ đầu mỗi lần. Học liên tục còn bao gồm tái đào tạo lưu trạng thái (stateful training) - mô hình tiếp tục đào tạo trên dữ liệu mới. Đào tạo lưu trạng thái cho phép cập nhật mô hình với ít dữ liệu hơn và thường được áp dụng cho việc cập nhật dữ liệu, chứ không phải thay đổi kiến trúc mô hình. Điều này cung cấp khả năng cập nhật mô hình nhanh chóng, dù từ đầu hoặc tinh chỉnh, và triển khai cập nhật nhanh chóng.



Hình 7. Stateless Retraining so với Stateful Training

Một đặc điểm đáng chú ý nhưng thường bị bỏ qua của đào tạo lưu trạng thái (stateful training) là khả năng tránh lưu trữ dữ liệu hoàn toàn. Trong đào tạo không lưu trạng thái (stateless retraining) truyền thống, một mẫu dữ liệu có thể được sử dụng lại nhiều lần, yêu cầu lưu trữ dữ liệu. Trong đào tạo lưu trạng thái, mỗi cập nhật mô hình chỉ sử dụng dữ liệu mới, nghĩa là một mẫu dữ liệu chỉ được sử dụng một lần để đào tạo. Điều này có thể cho phép đào tạo mô hình mà không cần lưu trữ dữ liệu vĩnh viễn, giúp giảm bớt mối quan tâm về quyền riêng tư dữ liệu. Tuy nhiên, nhiều công ty vẫn ngần ngại loại bỏ dữ liệu do xu hướng lưu giữ mọi thứ ngày nay. Đào tạo lưu trạng thái không có nghĩa là không cần đào tạo từ đầu. Các công ty thành công nhất trong việc sử dụng đào tạo lưu trạng thái cũng đôi khi đào tạo mô hình của họ từ đầu với một lượng lớn dữ liệu để hiệu chỉnh nó. Họ cũng có thể đào tạo mô hình từ đầu song song với đào tạo lưu trạng thái và sau đó kết hợp cả hai mô hình cập nhật sử dụng các kỹ thuật như máy chủ tham số (parameter server).

Học liên tục trong học máy tập trung vào việc thiết lập cơ sở hạ tầng để nhà khoa học dữ liệu hoặc kỹ sư ML có thể cập nhật mô hình theo nhu cầu, dù là từ đầu hay thông qua tinh chỉnh, và triển khai nhanh chóng. Hai loại cập nhật mô hình chính là lặp lại mô hình, nơi thêm tính năng mới hoặc thay đổi kiến trúc mô hình, và lặp lại dữ liệu, nơi giữ nguyên kiến trúc mô hình nhưng làm mới với dữ liệu mới. Đào tạo lưu trạng thái (stateful training) thường được áp dụng cho việc lặp lại dữ liệu. Công nghệ như chuyển giao kiến thức và phẫu thuật mô hình có thể giúp bỏ qua đào tạo từ đầu cho các cập nhật mô hình.

### 2.1.2 Tại sao phải học liên tục?

Chúng ta đã thảo luận về lý do tại sao học liên tục là quan trọng và cần thiết trong lĩnh vực machine learning và deep learning. Học liên tục giúp chúng ta thích ứng nhanh chóng với các thay đổi trong dữ liệu và các sự kiện hiếm gặp, cung cấp khả năng đưa ra dự đoán chính xác ngay cả trong trường hợp khởi động nguội liên tục. Việc này đặc biệt quan trọng trong các ứng dụng thời gian thực như các dịch vụ chia sẻ chuyến đi và trang web thương mại điện tử.

Vấn đề khởi động nguội xảy ra khi mô hình máy học phải đưa ra gợi ý hoặc dự đoán cho người dùng mới mà không có dữ liệu lịch sử của họ. Ví dụ, trong trường hợp đề xuất các bộ phim cho người dùng, hệ thống đề xuất thường dựa vào lịch sử xem phim của người dùng để tạo ra các gợi ý cá nhân. Tuy nhiên, khi người dùng mới đăng ký, không có thông tin về lịch sử xem phim của họ. Điều này đưa ra thách thức vì hệ thống phải đưa ra các gợi ý ban đầu dựa trên thông tin chung chung như các bộ phim phổ biến nhất trên trang web.

Và hãy tưởng tượng bạn đang xây dựng một mô hình để xác định giá cho một dịch vụ chia sẻ chuyến đi như Lyft. Trong lịch sử, nhu cầu đi xe vào tối thứ Năm ở khu phố đặc biệt này rất chậm, vì vậy mô hình dự đoán giá đi xe thấp, điều này khiến người lái xe ít hấp dẫn hơn khi lên đường. Tuy nhiên, vào tối thứ Năm này, có một sự kiện lớn trong khu phố, và đột nhiên nhu cầu đi xe tăng vọt. Nếu mô hình của bạn không thể phản ứng với thay đổi này đủ nhanh bằng cách tăng dự đoán giá và huy động nhiều tài xế hơn đến khu vực lân cận đó, người lái sẽ phải đợi một thời gian dài để đi xe, điều này gây ra trải nghiệm người dùng tiêu cực. Họ thậm chí có thể chuyển sang đối thủ cạnh tranh, điều này khiến bạn mất doanh thu. Một trường hợp sử dụng khác của việc học liên tục là thích nghi với các sự kiện hiếm gặp. Ví như bạn làm việc cho một trang web thương mại điện tử như Amazon. Black Friday là một sự kiện mua sắm quan trọng chỉ diễn ra mỗi năm một lần. Không có cách nào bạn có thể thu thập đủ dữ liệu lịch sử cho mô hình của mình để có thể đưa ra dự đoán chính xác về cách khách hàng của bạn sẽ hành xử trong suốt Thứ Sáu Đen năm nay. Để cải thiện hiệu suất, mô hình của bạn nên học suốt cả ngày với dữ liệu mới. Năm 2019, Alibaba đã mua lại Data Artisans, nhóm dẫn đầu việc phát triển khung xử lý luồng Apache Flink,

với giá 103 triệu USD để nhóm có thể giúp họ điều chỉnh Flink cho các trường hợp sử dụng ML. Trường hợp sử dụng hàng đầu của họ là đưa ra các đề xuất tốt hơn vào Ngày độc thân, một dịp mua sắm ở Trung Quốc tương tự như Thứ Sáu Đen ở Mỹ.

Học liên tục có khả năng biến các mô hình thành mạnh mẽ hơn và có khả năng dự đoán tốt hơn, đặc biệt trong các ứng dụng như TikTok, nơi mô hình có thể thích ứng với từng người dùng trong thời gian ngắn. Một điều thú vị là, việc học liên tục không chỉ là việc cải thiện hiệu suất, mà còn là việc khám phá tiềm năng mới trong machine learning.

Việc đặt câu hỏi "Tại sao phải học liên tục?" có thể thay bằng "Tại sao không học liên tục?" bởi vì học liên tục là một sự tiến bộ trong so sánh với việc học hàng loạt truyền thống. Nó không chỉ cho phép bạn duy trì và cải thiện mô hình của bạn theo thời gian mà còn giúp bạn xử lý các trường hợp sử dụng đa dạng hơn.

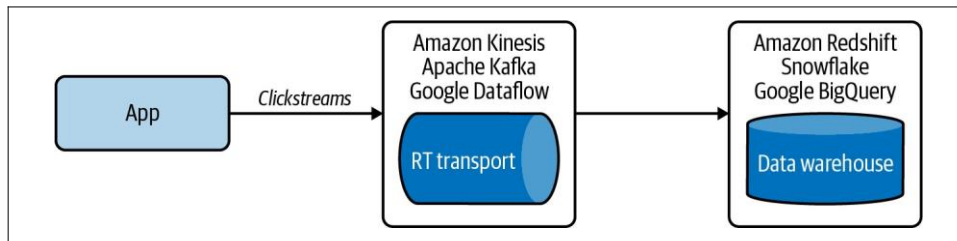
Dù việc học liên tục có thể đòi hỏi nỗ lực và tài nguyên tương tự như học hàng loạt, nhưng sự phát triển của công cụ MLOps cho học liên tục đang diễn ra nhanh chóng, đưa ra hy vọng rằng trong tương lai, việc triển khai học liên tục sẽ trở nên dễ dàng hơn và hiệu quả hơn.

### ***2.1.3 Những thách thức học liên tục***

Học liên tục là một lĩnh vực quan trọng và hứa hẹn trong machine learning, nhưng nó vẫn đối diện với một số thách thức quan trọng: truy cập dữ liệu mới, đánh giá và thuật toán.

#### ***2.1.3.1 Truy cập dữ liệu mới***

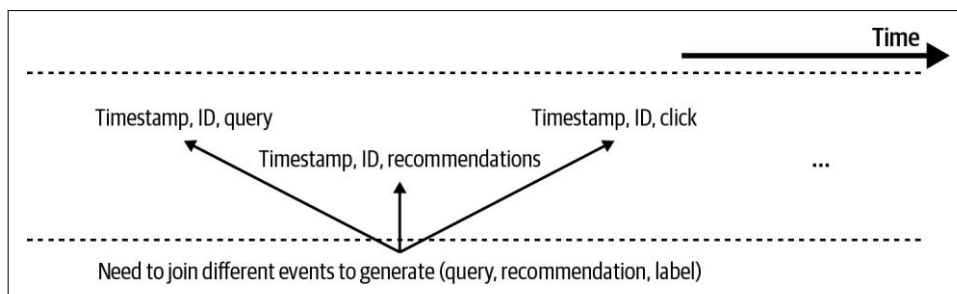
Thách thức đầu tiên trong việc học liên tục là thu thập dữ liệu mới một cách nhanh chóng. Để cập nhật mô hình hàng giờ, bạn cần dữ liệu mới hàng giờ. Hiện nay, nhiều công ty trích dẫn dữ liệu đào tạo từ kho dữ liệu của họ. Tuy nhiên, tốc độ trích dẫn dữ liệu từ kho dữ liệu phụ thuộc vào tốc độ dữ liệu được lưu vào kho dữ liệu đó. Một cách khác là cho phép trích dẫn dữ liệu trước khi nó được lưu vào kho dữ liệu, ví dụ, trực tiếp từ các kênh truyền thời gian thực như Kafka và Kinesis, để vận chuyển dữ liệu từ ứng dụng đến kho dữ liệu.



Hình 8. Lấy dữ liệu thời gian thực

Tốc độ trích dẫn dữ liệu mới cũng chưa đủ. Nếu mô hình của bạn cần dữ liệu có nhãn để cập nhật (như hầu hết các mô hình hiện nay), thì dữ liệu này cũng cần được gắn nhãn. Các nhiệm vụ tốt nhất cho học liên tục là những nhiệm vụ mà bạn có thể dễ dàng thu thập nhãn tự nhiên với chu kỳ phản hồi ngắn, như định giá động, ước tính thời gian đến, dự đoán giá cổ phiếu, dự đoán việc người dùng bấm vào quảng cáo và hệ thống đề xuất nội dung trực tuyến.

Tuy nhiên, nhãn tự nhiên này thường không được tạo ra dễ dàng mà thay vào đó là các hoạt động hành vi cần phải được trích xuất thành nhãn. Quá trình này được gọi là tính toán nhãn và có thể tốn kém nếu số lượng dữ liệu lớn. Tuy nhiên, có cách để tăng tốc quá trình gắn nhãn bằng cách sử dụng công cụ gắn nhãn tự động và nhãn từ cộng đồng.



Hình 9. Đơn giản hóa quá trình trích gắn nhãn

Mặc dù việc xây dựng hạ tầng xử lý dòng để truy cập dữ liệu mới và trích xuất nhãn nhanh chóng từ các kênh truyền thời gian thực có thể đòi hỏi nhiều công sức và chi phí kỹ thuật, nhưng công cụ xung quanh xử lý dòng đang phát triển nhanh chóng, làm cho việc xây dựng hạ tầng cho học liên tục trở nên dễ dàng và giá cả hợp lý hơn cho các công ty.

### 2.1.3.2 Thách thức đánh giá

Thách thức lớn nhất của việc học liên tục không nằm trong việc viết một hàm để liên tục cập nhật mô hình của bạn - bạn có thể làm điều đó bằng cách viết một kịch bản! Thách thức lớn nhất đó là đảm bảo rằng việc cập nhật này đủ tốt để triển khai. Với học liên tục, cơ hội để việc cập nhật mô hình thất bại tăng lên do cập nhật thường xuyên hơn. Hơn nữa, học liên tục làm cho mô hình của bạn dễ bị tác động và tấn công từ phía người dùng. Điều này có nguy cơ người dùng cung cấp dữ liệu độc hại để đánh lừa mô hình.

Để tránh các sự cố tương tự hoặc tồi tệ hơn, quan trọng đánh giá kỹ lưỡng mỗi cập nhật của mô hình để đảm bảo hiệu suất và an toàn trước khi triển khai cập nhật đó cho một đối tượng mở rộng hơn. Điều này có thể làm chậm tốc độ cập nhật mô hình. Ví dụ, một công ty thanh toán trực tuyến muốn cập nhật hệ thống phát hiện gian lận nhanh chóng để thích nghi với các thay đổi nhanh chóng trong mẫu gian lận. Tuy nhiên, quá trình đánh giá mất thời gian do tính không cân đối của nhiệm vụ, khiến họ chỉ có thể cập nhật hệ thống mỗi hai tuần.

### ***2.1.3.3 Thách thức thuật toán***

Thách thức chính liên quan đến thuật toán trong học liên tục nằm ở khả năng làm việc với tập dữ liệu một cách hiệu quả, đặc biệt là đối với các thuật toán dựa trên ma trận và cây quyết định mà muốn được cập nhật rất nhanh (ví dụ: hàng giờ).

So sánh hai loại mô hình khác nhau: mạng neural và mô hình dựa trên ma trận, như một mô hình lọc cộng tác. Mô hình lọc cộng tác sử dụng ma trận người dùng-sản phẩm và kỹ thuật giảm chiều.

Bạn có thể cập nhật mô hình mạng neural với bất kỳ kích thước batch dữ liệu nào. Bạn thậm chí có thể thực hiện bước cập nhật chỉ với một ví dụ dữ liệu. Tuy nhiên, nếu bạn muốn cập nhật mô hình lọc cộng tác, bạn cần sử dụng toàn bộ tập dữ liệu trước khi thực hiện giảm chiều trên nó. Tất nhiên, bạn có thể áp dụng giảm chiều vào ma trận của bạn mỗi khi bạn cập nhật ma trận với một ví dụ dữ liệu mới, nhưng nếu ma trận của bạn lớn, việc giảm chiều này sẽ quá chậm và tốn kém để thực hiện thường xuyên. Do đó, mô hình này ít thích hợp hơn cho việc học từ tập dữ liệu một cách phân mềm hóa so với mô hình mạng neural.

### ***2.1.4 Bốn giai đoạn học liên tục***

Có bốn giai đoạn trong quá trình học liên tục:

**\*\*Giai đoạn 1: Retraining thủ công, không lưu trạng thái\*\***

Ở giai đoạn này, nhóm máy học thường tập trung vào việc phát triển các mô hình máy học để giải quyết càng nhiều vấn đề kinh doanh càng tốt. Các mô hình hiện tại chỉ được cập nhật khi hiệu suất của chúng giảm đến mức làm hại hơn là có lợi, và nhóm có đủ thời gian để cập nhật. Các mô hình có thể được cập nhật từ một lần mỗi sáu tháng đến một lần mỗi quý hoặc thậm chí một năm.

**\*\*Giai đoạn 2: Retraining tự động\*\***

Sau vài năm, đội ngũ của bạn đã triển khai các mô hình để giải quyết hầu hết các vấn đề cơ bản. Quá trình cập nhật mô hình thủ công đã trở thành một vấn đề lớn không thể bỏ qua. Đội ngũ của bạn quyết định viết một tập lệnh để thực hiện tất cả các bước cập nhật mô hình tự động. Kịch bản này được thực thi định kỳ bằng cách sử dụng quy trình hàng loạt như Spark.

**\*\*Giai đoạn 3: Retraining tự động, có lưu trạng thái\*\***

Ở giai đoạn này, bạn cấu hình lại kịch bản cập nhật tự động sao cho khi cập nhật mô hình, nó trước tiên tìm kiếm điểm kiểm tra trước đó và tải nó vào bộ nhớ trước khi tiếp tục huấn luyện trên điểm kiểm tra này.

**\*\*Giai đoạn 4: Học liên tục\*\***

Ở giai đoạn này, mô hình của bạn sẽ tự động cập nhật mỗi khi phân phối dữ liệu thay đổi và hiệu suất của mô hình giảm đi. Mục tiêu cuối cùng là kết hợp học liên tục với triển khai tại điểm cuối, tức là mô hình có khả năng tự động cập nhật và thích nghi với môi trường của nó mà không cần đồng bộ với máy chủ tập trung.

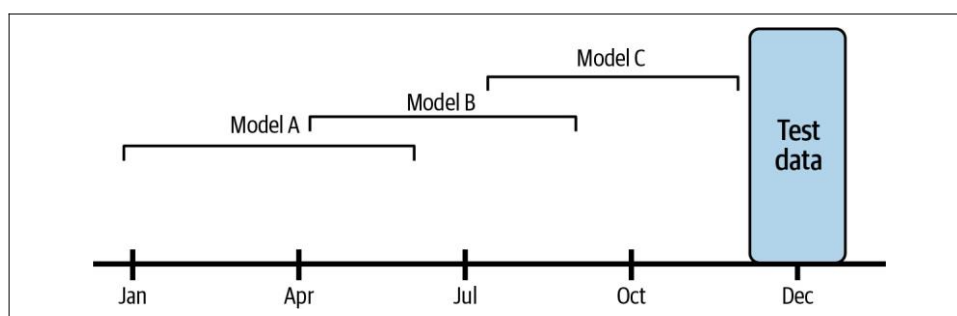


Để chuyển từ giai đoạn trước đó sang giai đoạn tiếp theo, bạn cần đáp ứng các yêu cầu cụ thể và có sự thay đổi trong cơ sở tư duy và cơ sở hạ tầng của bạn. Các giai đoạn này đại diện cho sự phát triển và chuyển đổi của doanh nghiệp trong việc triển khai và quản lý học liên tục.

### 2.1.5 Tần suất cập nhật mô hình của bạn

Xác định tần suất cập nhật mô hình học máy thực sự phụ thuộc vào giá trị mà bạn có thể thu được từ việc sử dụng dữ liệu mới. Nếu bạn thấy việc cập nhật mô hình từ mỗi tháng lên hàng tuần hoặc hàng ngày mang lại cải thiện đáng kể trong hiệu suất, thì bạn nên xem xét cập nhật thường xuyên hơn.

Một phương pháp để xác định tần suất cập nhật là thực hiện thí nghiệm bằng cách huấn luyện mô hình trên dữ liệu từ các khoảng thời gian khác nhau trong quá khứ và đánh giá hiệu suất của mô hình trên dữ liệu hiện tại. Ví dụ, bạn có thể huấn luyện phiên bản mô hình A trên dữ liệu từ tháng 1 đến tháng 6, phiên bản mô hình B trên dữ liệu từ tháng 4 đến tháng 9 và phiên bản mô hình C trên dữ liệu từ tháng 6 đến tháng 11. Sau đó, bạn đánh giá hiệu suất của từng phiên bản mô hình này trên dữ liệu từ tháng 12. Sự khác biệt trong hiệu suất của các phiên bản sẽ giúp bạn xác định giá trị mà mô hình của bạn có thể đạt được từ việc sử dụng dữ liệu mới. Nếu mô hình được huấn luyện trên dữ liệu cách đó một tháng kém hiệu suất so với mô hình được huấn luyện trên dữ liệu cách đó một tháng, thì bạn biết rằng bạn không nên đợi một tháng để cập nhật mô hình.



Hình 10. Kiểm tra hiệu suất với dữ liệu khác nhau xem hiệu suất thay đổi

Câu trả lời cho câu hỏi về tần suất cập nhật mô hình sẽ phụ thuộc vào kết quả của các thí nghiệm và sự cân nhắc giữa việc cải thiện kiến trúc mô hình và sử dụng

hiệu quả dữ liệu mới. Khi hệ thống cơ sở hạ tầng của bạn phát triển và quá trình cập nhật mô hình trở nên tự động hơn, câu trả lời cho câu hỏi này sẽ phụ thuộc vào giá trị mà bạn có thể thu được từ việc sử dụng dữ liệu mới.

## 2.2. GIỚI THIỆU CHUNG VỀ TEST PRODUCTION

Chúng ta đã nói về nguy cơ triển khai các mô hình mà chưa được đánh giá đầy đủ. Để đánh giá đủ sâu mô hình của bạn, bạn cần một sự kết hợp giữa đánh giá ngoại tuyến và đánh giá trực tuyến được thảo luận trong phần này. Để hiểu tại sao việc đánh giá ngoại tuyến không đủ, hãy cùng đi qua hai loại kiểm tra quan trọng trong đánh giá ngoại tuyến: test split (phân chia thử nghiệm) và backtest (kiểm tra lại).

Loại đánh giá mô hình đầu tiên mà bạn có thể nghĩ đến là việc sử dụng test split (phân chia thử nghiệm) cổ điển mà bạn có thể sử dụng để đánh giá mô hình của bạn ngoại tuyến. Các phân chia thử nghiệm này thường là cố định và phải là cố định để bạn có một chỉ số tham chiếu đáng tin cậy để so sánh nhiều mô hình. Sẽ khó để so sánh kết quả thử nghiệm của hai mô hình nếu chúng được thử nghiệm trên các tập thử nghiệm khác nhau.

Tuy nhiên, nếu bạn cập nhật mô hình để thích nghi với một phân phối dữ liệu mới, thì việc đánh giá mô hình mới này trên các phân chia thử nghiệm từ phân phối cũ không đủ. Giả sử rằng càng mới mẻ dữ liệu, càng có khả năng nó sẽ xuất phát từ phân phối hiện tại, một ý tưởng là kiểm tra mô hình của bạn trên dữ liệu gần đây nhất mà bạn có quyền truy cập. Vậy sau khi bạn đã cập nhật mô hình của mình trên dữ liệu từ ngày trước đó, bạn có thể muốn kiểm tra mô hình này trên dữ liệu từ giờ trước đó (giả sử rằng dữ liệu từ giờ trước đó không được bao gồm trong dữ liệu được sử dụng để cập nhật mô hình của bạn). Phương pháp kiểm tra một mô hình dự đoán trên dữ liệu từ một khoảng thời gian cụ thể trong quá khứ được gọi là backtest (kiểm tra lại). Câu hỏi là liệu backtest có đủ để thay thế cho phân chia thử nghiệm tĩnh hay không. Không hoàn toàn. Nếu có điều gì đó sai với đường ống dữ liệu của bạn và một số dữ liệu từ giờ trước đó bị hỏng, việc đánh giá mô hình của bạn chỉ dựa trên dữ liệu gần đây này không đủ.

Với backtest, bạn vẫn nên đánh giá mô hình của mình trên một tập thử nghiệm cố định mà bạn đã nghiên cứu kỹ lưỡng và (phần lớn) tin tưởng như một hình thức kiểm tra tinh thần.

Bởi vì phân phối dữ liệu thay đổi, việc một mô hình hoạt động tốt trên dữ liệu từ giờ trước đó không có nghĩa là nó sẽ tiếp tục hoạt động tốt trên dữ liệu trong tương lai. Cách duy nhất để biết liệu một mô hình có hoạt động tốt trong sản xuất hay không là triển khai nó. Thông điệp này đã dẫn đến một khái niệm có vẻ kinh hoàng nhưng cần thiết: thử nghiệm trong môi trường sản xuất. Tuy nhiên, thử nghiệm trong môi trường sản xuất không nhất thiết phải đáng sợ. Có các kỹ thuật giúp bạn đánh giá mô hình của mình trong môi trường sản xuất một cách (phần lớn) an toàn. Trong phần này, chúng ta sẽ bàn về các kỹ thuật sau đây: triển khai bóng (shadow deployment), thử nghiệm A/B, phân tích canary, thử nghiệm xen kẽ (interleaving experiments) và thuật toán bandit.

### **2.2.1 Triển khai bóng (Shadow Deployment)**

Triển khai bóng (Shadow Deployment) là một phương pháp được sử dụng để triển khai mô hình máy học trong môi trường thực tế mà không gây ảnh hưởng đến người dùng chính cho đến khi mô hình máy học mới đã được đánh giá và xác định là đủ tốt để thay thế mô hình hiện tại. Dưới đây là một số mục đích và tính năng của triển khai bóng:

1. Kiểm tra hiệu quả: Triển khai bóng giúp đánh giá hiệu quả của mô hình máy học trong môi trường thực tế mà không đánh đổi hiệu suất của hệ thống. Điều này cho phép xác định xem mô hình máy học mới có hoạt động tốt trong thực tế hay không.

2. Mô hình độc đáo: Trong quá trình triển khai bóng, một phiên bản mô hình độc đáo được tạo ra để chạy song song với mô hình hiện tại. Mô hình độc đáo này không phục vụ dự đoán cho người dùng cuối, nhưng thay vào đó, nó được sử dụng để lưu trữ kết quả dự đoán cho mục đích đánh giá.

3. So sánh với mô hình hiện đang: Triển khai bóng cho phép so sánh hiệu suất của mô hình máy học mới (mô hình độc đáo) với mô hình hiện tại. Điều này giúp đội

ngữ huấn luyện máy học xác định liệu mô hình mới có cải thiện so với mô hình cũ hay không.

4. Phát hành mô hình: Nếu kết quả đánh giá cho thấy mô hình máy học mới (mô hình độc đáo) hoạt động tốt hơn mô hình hiện tại, thì mô hình độc đáo có thể thay thế mô hình hiện tại trong quá trình triển khai thực tế. Điều này giúp tối ưu hóa hiệu suất hệ thống.

Triển khai bóng là một cách an toàn để thử nghiệm mô hình mới trong môi trường sản xuất mà không gây ảnh hưởng đến hoạt động hiện tại. Mô hình mới được triển khai song song với mô hình hiện tại, và mỗi yêu cầu đều được định tuyến đến cả hai mô hình.

Mặc dù cả hai mô hình đều tạo ra dự đoán, chỉ có dự đoán từ mô hình hiện tại được phục vụ cho người dùng. Dự đoán từ mô hình mới được ghi lại để phân tích. Nếu dự đoán của mô hình mới đủ tốt, nó sẽ thay thế mô hình hiện tại. Điều này giúp kiểm tra hiệu suất của mô hình mới trong môi trường thực tế mà không gây rủi ro cho hoạt động hiện tại. Tuy nhiên, kỹ thuật này tốn kém vì nó gấp đôi số lượng dự đoán cần phải xử lý.

### **2.2.2 A/B testing**

A/B testing (còn được gọi là thử nghiệm chia tách) là một phương pháp được sử dụng để so sánh hai biến thể của một đối tượng, thường bằng cách thử nghiệm phản ứng đối với hai biến thể này và xác định biến thể nào hiệu quả hơn. Trong ngữ cảnh học máy, A/B testing có thể được sử dụng để đánh giá và so sánh hiệu suất của các mô hình khác nhau, chẳng hạn như mô hình hiện tại và một mô hình mới vừa được cập nhật gần đây. Các bước chính của A/B testing như sau:

1. Triển khai mô hình mới song song với mô hình hiện tại.
2. Một phần trăm lưu lượng được định tuyến đến mô hình mới để thực hiện các dự đoán, trong khi phần còn lại được định tuyến đến mô hình hiện tại để thực hiện các dự đoán.

3. Theo dõi và phân tích các dự đoán và phản hồi từ người dùng từ cả hai mô hình để xác định xem sự khác biệt trong hiệu suất của họ có ý nghĩa thống kê hay không.

A/B testing đã trở nên ngày càng phổ biến, với các công ty như Google và Amazon thực hiện hàng nghìn thử nghiệm A/B hàng năm. Đây là một phản ứng phổ biến trong số các kỹ sư học máy về cách đánh giá các mô hình học máy trong môi trường thực tế.

Có một số thách thức và xem xét khi thực hiện A/B testing:

- Ngẫu nhiên hóa: Lưu lượng được định tuyến đến từng mô hình phải thực sự ngẫu nhiên. Nếu không, kết quả thử nghiệm sẽ không hợp lệ. Ví dụ, nếu có sự thiên vị lựa chọn trong cách lưu lượng được định tuyến đến hai mô hình, như người dùng được tiếp cận với mô hình A thường sử dụng điện thoại di động trong khi người dùng được tiếp cận với mô hình B thường sử dụng máy tính để bàn, thì nếu mô hình A có độ chính xác cao hơn mô hình B, không rõ liệu điều này là do A tốt hơn B hay "sử dụng điện thoại" ảnh hưởng đến chất lượng dự đoán.
- Kích thước mẫu: Số lượng mẫu cần thiết cho một thử nghiệm A/B là một câu hỏi phức tạp. A/B testing đòi hỏi một số lượng mẫu đủ để có đủ tin cậy về kết quả. Tính toán số lượng mẫu cần thiết cho một thử nghiệm A/B là một câu hỏi đơn giản nhưng có một câu trả lời rất phức tạp, và được khuyến nghị tham khảo một cuốn sách về A/B testing để biết thêm thông tin.
- Ý nghĩa thống kê: Mặc dù hữu ích, sự ý nghĩa thống kê không phải lúc nào cũng là điều chắc chắn. Nếu giá trị p được thu được từ một kiểm tra hai mẫu nhỏ hơn ngưỡng ý nghĩa (thông thường là 0,05), điều này không đảm bảo rằng kết quả là hợp lệ. Ví dụ, nếu cùng một thử nghiệm A/B được thực hiện nhiều lần, kết quả có thể thay đổi, và ngay cả khi kết quả có ý nghĩa thống kê, có khả năng rằng nếu thử nghiệm được thực hiện lại, một mô hình khác sẽ được chọn.
- Nhiều biến thể: Trong một số trường hợp, có thể có nhiều hơn hai biến thể cần thử nghiệm, chẳng hạn như thử nghiệm A/B/C/D. Điều này có thể giúp so sánh nhiều mô hình đồng thời và lựa chọn mô hình hoạt động tốt nhất[3]. Tuy nhiên,

thử nghiệm với nhiều biến thể có thể dẫn đến tính toán và diễn giải phức tạp hơn, và quan trọng là cân nhắc về sự đánh đổi giữa số lượng biến thể và độ tin cậy của kết quả.

Tổng kết, A/B testing là một phương pháp quý báu để so sánh hiệu suất của các mô hình khác nhau và lựa chọn mô hình tốt nhất dựa trên các chỉ số được định trước. Tuy nhiên, quan trọng là xem xét các thách thức và xem xét được đề cập ở trên để đảm bảo tính hợp lệ và đáng tin cậy của kết quả thử nghiệm.

### **2.2.3 Canary Release**

Canary Release (Phát Hành Canary) là một kỹ thuật triển khai giúp giảm nguy cơ khi đưa một phiên bản phần mềm hoặc mô hình học máy mới vào môi trường sản xuất. Kỹ thuật này bao gồm việc triển khai từ từ các thay đổi đối với một tập người dùng nhỏ trước khi triển khai cho toàn bộ hệ thống và làm cho chúng có sẵn cho mọi người. Phương pháp Canary Release thường được sử dụng trong ngữ cảnh triển khai cả phần mềm và mô hình học máy và hoạt động như sau:

#### **1. \*\*Triển Khai Mô Hình Mới (Canary)\*\*:**

- Trong ngữ cảnh triển khai mô hình học máy, "mô hình mới" là mô hình mới hoặc đã được cập nhật mà bạn muốn kiểm tra.
- Mô hình mới được triển khai song song với mô hình hiện tại (thường được gọi là "mô hình kiểm soát" hoặc "mô hình gốc").

#### **2. \*\*Định Tuyến Giao Lưu\*\*:**

- Ban đầu, chỉ một phần của lưu lượng truy cập được chuyển đến mô hình mới (canary). Phần lớn lưu lượng vẫn được phục vụ bởi mô hình hiện tại.
- Tỷ lệ lưu lượng truy cập được đưa vào mô hình canary có thể dần dần tăng theo thời gian.

#### **3. \*\*Đánh Giá Hiệu Suất\*\*:**

- Quá trình theo dõi và đánh giá hiệu suất của mô hình canary là quan trọng.
- Các chỉ số hiệu suất quan trọng đối với ứng dụng cụ thể của bạn (ví dụ: độ chính xác, độ trễ, tương tác người dùng) được theo dõi và so sánh với mô hình hiện

tại.

#### 4. **\*\*Các Điểm Quyết Định\*\***:

- Nếu mô hình canary hoạt động ổn định và đáp ứng các tiêu chí đã định trước, bạn có thể dần dần tăng lưu lượng truy cập vào mô hình canary. Điều này có nghĩa là mô hình canary đang từ từ thay thế mô hình hiện tại.

- Nếu hiệu suất của mô hình canary giảm sút hoặc ảnh hưởng xấu đối với các chỉ số quan trọng, thì thường mô hình này sẽ bị hủy và toàn bộ lưu lượng sẽ được định tuyến trở lại mô hình hiện tại.

#### 5. **\*\*Hoàn Thành hoặc Hủy Bỏ\*\***:

- Quá trình tiếp tục cho đến khi đáp ứng một trong các điều kiện sau:
  - Mô hình canary phục vụ toàn bộ lưu lượng, cho thấy quá trình triển khai thành công.
  - Mô hình canary bị hủy bỏ do hiệu suất không đủ và toàn bộ lưu lượng truy cập đều được định tuyến trở lại mô hình hiện tại.

Phương pháp Canary Release rất hữu ích để giảm nguy cơ và đảm bảo rằng phiên bản phần mềm hoặc mô hình học máy mới không gây ra sự cố cho toàn bộ người dùng nếu xảy ra vấn đề. Mặc dù Canary Release có nhiều điểm tương đồng với kiểm tra A/B, nhưng không luôn luôn yêu cầu sự ngẫu nhiên trong việc định tuyến lưu lượng, điều này làm cho nó phù hợp với nhiều tình huống triển khai khác nhau. Ví dụ, bạn có thể bắt đầu bằng cách triển khai mô hình canary cho một thị trường ít quan trọng hơn hoặc một đoạn người dùng cụ thể trước khi triển khai rộng rãi.

Netflix và Google nổi tiếng với việc sử dụng phân tích canary tự động, và họ đã chia sẻ thông tin về cách kỹ thuật này được áp dụng trong các công ty của họ. Công cụ phân tích canary tự động giúp theo dõi và đưa ra quyết định một cách hiệu quả trong quá trình phát hành canary để đảm bảo sự chuyển đổi trơn tru sang phiên bản phần mềm hoặc mô hình học máy mới trong khi giảm thiểu các rủi ro.

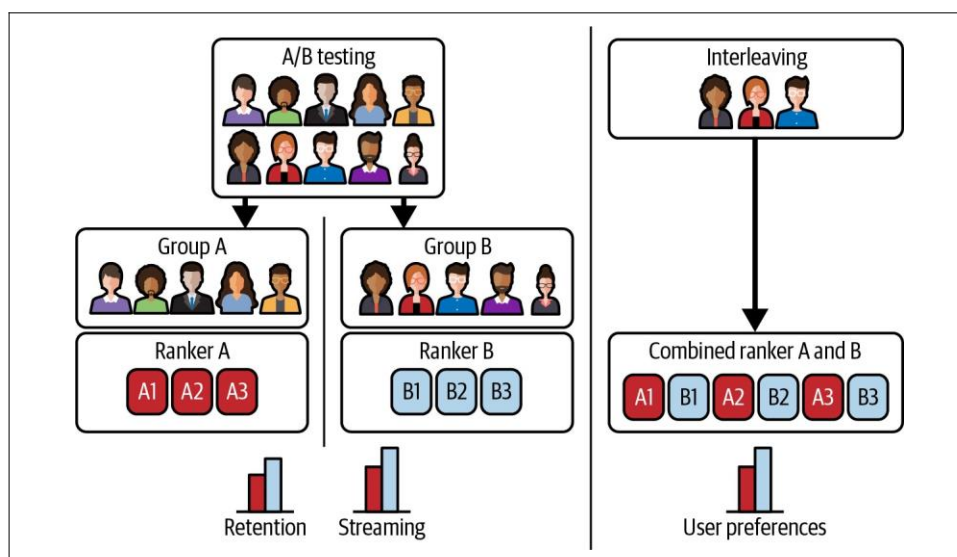
### **2.2.4 Interleaving Experiments (Thí nghiệm xen kẽ)**

Hãy tưởng tượng bạn có hai hệ thống giới thiệu, A và B, và bạn muốn đánh giá

cái nào tốt hơn. Mỗi lần, một mô hình đề xuất 10 mặt hàng mà người dùng có thể thích. Với thử nghiệm A / B, bạn sẽ chia người dùng của mình thành hai nhóm: một nhóm tiếp xúc với A và nhóm còn lại tiếp xúc với B. Mỗi người dùng sẽ được tiếp xúc với các đề xuất được thực hiện bởi một mô hình.

Thay vì tiết lộ cho một người dùng các đề xuất từ một mô hình, chúng ta tiết lộ cho người dùng đó các đề xuất từ cả hai mô hình và xem họ sẽ nhấp vào các đề xuất của mô hình nào. Đó là ý tưởng đằng sau các thí nghiệm xen kẽ, một ý tưởng được đề xuất ban đầu bởi Thorsten Joachims vào năm 2002 để đánh giá xếp hạng tìm kiếm. Trong các thí nghiệm, Netflix đã tìm thấy rằng việc xen kẽ "xác định một cách đáng tin cậy các thuật toán tốt nhất với kích thước mẫu nhỏ hơn đáng kể so với việc thử nghiệm A/B truyền thống."

Trong thử nghiệm A / B, các số liệu cốt lõi như tỷ lệ duy trì và phát trực tuyến được đo lường và so sánh giữa hai nhóm. Trong xen kẽ, hai thuật toán có thể được so sánh bằng cách đo lường sở thích của người dùng. Bởi vì xen kẽ có thể được quyết định bởi sở thích của người dùng, không có gì đảm bảo rằng sở thích của người dùng sẽ dẫn đến các số liệu cốt lõi tốt hơn.

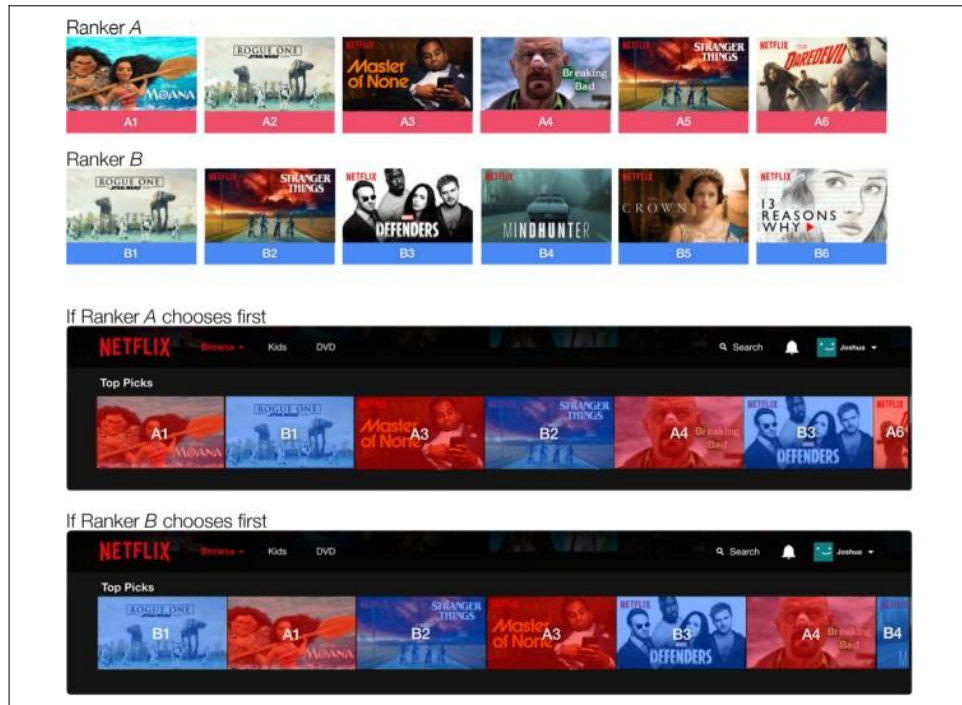


Hình 11. Một minh họa về thử nghiệm xen kẽ so với thử nghiệm A / B

Khi chúng ta hiển thị đề xuất từ nhiều mô hình cho người dùng, điều quan trọng cần lưu ý là vị trí của một đề xuất sẽ ảnh hưởng đến khả năng người dùng nhấp vào nó. Ví dụ, người dùng rất có khả năng nhấp vào đề xuất hàng đầu hơn là đề xuất ở



phía dưới. Để đảm bảo rằng kết quả của việc xen kẽ là hợp lệ, chúng ta cần đảm bảo rằng tại bất kỳ vị trí nào, một đề xuất được tạo ra bằng xác suất bằng nhau bởi mô hình A hoặc B. Để đảm bảo điều này, một phương pháp mà chúng ta có thể sử dụng là xen kẽ theo phong cách "chọn đội," mô phỏng quá trình lựa chọn trong thể thao. Đối với mỗi vị trí đề xuất, chúng ta ngẫu nhiên chọn mô hình A hoặc B với xác suất bằng nhau, và mô hình được chọn sẽ chọn đề xuất hàng đầu mà chưa được chọn.



Hình 12. Xen kẽ các đề xuất video từ hai thuật toán xếp hạng bằng cách sử dụng bản nháp nhóm

## 2.2.5 Thuật toán Bandits

Các thuật toán Bandit (còn được gọi là thuật toán máy cưa) có nguồn gốc từ lĩnh vực đánh bạc. Tưởng tượng một sòng bạc có nhiều máy đánh bạc với tỷ lệ trả thưởng khác nhau. Máy đánh bạc còn được gọi là "one-armed bandit," do đó có tên là "bandit." Bạn không biết máy đánh bạc nào trả thưởng cao nhất. Bạn có thể tiến hành thử nghiệm theo thời gian để tìm ra máy đánh bạc nào là tốt nhất và cực đại hóa số tiền bạn nhận được. Thuật toán Bandit đa cánh (Multi-armed bandit) là các thuật toán cho phép bạn cân bằng giữa việc tận dụng (chọn máy đánh bạc đã trả thưởng nhiều nhất trong quá khứ) và việc khám phá (chọn các máy đánh bạc khác có thể trả thưởng nhiều hơn).

Hiện nay, phương pháp tiêu chuẩn để kiểm tra các mô hình trong quá trình sản xuất là thử nghiệm A/B. Với thử nghiệm A/B, bạn định tuyến ngẫu nhiên lưu lượng đến từng mô hình để thực hiện các dự đoán và sau đó đo đạc cuối thử nghiệm xem mô hình nào hoạt động tốt hơn. Thử nghiệm A/B không có trạng thái: bạn có thể định tuyến lưu lượng đến từng mô hình mà không cần biết về hiệu suất hiện tại của chúng. Bạn có thể thực hiện thử nghiệm A/B ngay cả với dự đoán theo lô (batch prediction).

Khi bạn có nhiều mô hình để đánh giá, mỗi mô hình có thể được coi như một máy đánh bạc có tỷ lệ trả thưởng (tức là độ chính xác dự đoán) mà bạn không biết. Thuật toán Bandit cho phép bạn xác định cách định tuyến lưu lượng đến từng mô hình để xác định mô hình tốt nhất trong khi tối đa hóa độ chính xác dự đoán cho người dùng của bạn. Bandit có trạng thái: trước khi định tuyến yêu cầu đến một mô hình, bạn cần tính toán hiệu suất hiện tại của tất cả các mô hình. Điều này đòi hỏi ba điều kiện cần thiết:

- Mô hình của bạn phải có khả năng thực hiện dự đoán trực tuyến.
- Ưu tiên là có chu kỳ phản hồi ngắn: bạn cần nhận phản hồi về việc một dự đoán có tốt hay không. Điều này thường đúng đối với các nhiệm vụ mà nhân có thể xác định từ phản hồi của người dùng, như trong các gợi ý - nếu người dùng nhấp vào một gợi ý, nó được gán định là tốt. Nếu chu kỳ phản hồi ngắn, bạn có thể cập nhật tỷ lệ trả thưởng của mỗi mô hình nhanh chóng.
- Cần có một cơ chế để thu thập phản hồi, tính toán và theo dõi hiệu suất hiện tại của từng mô hình và định tuyến các yêu cầu dự đoán đến các mô hình khác nhau dựa trên hiệu suất hiện tại của chúng.

Thuật toán Bandit đã được nghiên cứu rộng rãi trong giới học thuật và đã được chứng minh là tiết kiệm dữ liệu hơn rất nhiều so với thử nghiệm A/B (trong nhiều trường hợp, thuật toán Bandit thậm chí còn tối ưu). Bandit đòi hỏi ít dữ liệu hơn để xác định mô hình nào là tốt nhất và đồng thời giảm thiểu chi phí cơ hội vì nó định tuyến lưu lượng đến mô hình tốt hơn nhanh chóng hơn. Tuy nhiên, thuật toán Bandit

khó triển khai hơn rất nhiều so với thử nghiệm A/B vì nó đòi hỏi tính toán và theo dõi tỷ lệ trả thưởng của các mô hình. Do đó, thuật toán Bandit không phổ biến trong ngành công nghiệp ngoại trừ một số công ty công nghệ lớn.

### **Contextual bandits as an exploration strategy**

Các thuật toán bandit ngữ cảnh (contextual bandits) giúp cân bằng việc hiển thị các mục người dùng sẽ thích và các mục cần phản hồi. Chúng là một kỹ thuật hiệu quả trong việc nâng cao hiệu suất của mô hình, đặc biệt trong các hệ thống gợi ý hoặc quảng cáo. Bandit ngữ cảnh khác với bandit thông thường ở chỗ chúng xem xét ngữ cảnh cụ thể của mỗi hành động. Tuy nhiên, việc triển khai chúng phức tạp hơn do phụ thuộc vào kiến trúc của mô hình học máy. Việc đánh giá mô hình học máy không chỉ liên quan đến việc chạy các bài kiểm tra, mà còn liên quan đến người chạy các bài kiểm tra đó. Quy trình đánh giá tốt nên được tự động hóa và kiểm soát chặt chẽ để đảm bảo chất lượng mô hình trước khi triển khai.

Hãy tưởng tượng bạn đang xây dựng một hệ thống gợi ý với 1,000 mục để gợi ý, điều này biến nó thành một vấn đề máy cưa 1,000 cánh. Mỗi lần, bạn chỉ có thể gợi ý 10 mục có liên quan nhất cho người dùng. Trong thuật ngữ bandit, bạn sẽ phải chọn ra 10 cánh tốt nhất. Các mục được hiển thị sẽ nhận phản hồi từ người dùng, thông qua việc người dùng có nhấp vào chúng hay không. Nhưng bạn sẽ không nhận được phản hồi về 990 mục còn lại. Điều này được gọi là vấn đề phản hồi một phần, còn được gọi là phản hồi bandit. Bạn cũng có thể nghĩ về contextual bandits như là một vấn đề phân loại với phản hồi bandit.

Hãy tưởng tượng rằng mỗi khi người dùng nhấp vào một mục, mục này nhận được 1 điểm giá trị. Khi một mục có 0 điểm giá trị, có thể là do mục đó chưa bao giờ được hiển thị cho người dùng hoặc đã được hiển thị nhưng không được nhấp vào. Bạn muốn hiển thị cho người dùng các mục có giá trị cao nhất đối với họ, nhưng nếu bạn tiếp tục hiển thị cho người dùng chỉ các mục có điểm giá trị cao nhất, bạn sẽ tiếp tục gợi ý các mục phổ biến cùng, và các mục chưa từng được hiển thị sẽ tiếp tục có 0 điểm giá trị.

Contextual bandits là một phương pháp nâng cao hiệu suất mô hình học máy, đặc biệt trong các hệ thống gợi ý. Chúng giúp cân nhắc giữa việc khám phá (hiển thị mục mới) và tận dụng (hiển thị mục đã biết là hấp dẫn). Thuật toán này đặc biệt hiệu quả trong việc xử lý các vấn đề phản hồi một phần, nơi mà chỉ một số hành động nhất định nhận được phản hồi từ người dùng. Contextual bandits đã được chứng minh là cải thiện đáng kể hiệu suất trong các nghiên cứu của các công ty lớn như Twitter và Google. Để tìm hiểu sâu hơn, bài báo "Deep Bayesian Bandits: Exploring in Online Personalized Recommendations" của Guo và cộng sự (2020) là một nguồn tài nguyên hữu ích.

Quy trình đánh giá mô hình học máy không chỉ liên quan đến việc chọn các kiểm tra phù hợp mà còn bao gồm việc xác định ai nên thực hiện các kiểm tra này. Các nhà khoa học dữ liệu thường chịu trách nhiệm đánh giá mô hình mà họ phát triển, nhưng cách tiếp cận này có thể mang lại thiên vị và kết quả không nhất quán. Để đảm bảo chất lượng mô hình, quan trọng là phải thiết lập quy trình đánh giá rõ ràng, tự động hóa và thực hiện một cách nghiêm ngặt.

### **2.2.6 Kết luận**

Làm thế nào để liên tục cập nhật các mô hình của bạn trong sản xuất để thích ứng với việc thay đổi phân phối dữ liệu. Chúng ta đã thảo luận về bốn giai đoạn mà một công ty có thể trải qua trong quá trình hiện đại hóa cơ sở hạ tầng của họ để học tập liên tục: từ thủ công, đào tạo từ giai đoạn đầu đến học tập liên tục tự động, không trạng thái.

Sau đó, chúng ta đã kiểm tra câu hỏi ám ảnh các kỹ sư ML tại các công ty thuộc mọi hình dạng và quy mô, "Tôi nên cập nhật mô hình của mình bao lâu một lần?" bằng cách thúc giục họ xem xét giá trị của độ mới dữ liệu đối với mô hình của họ và sự đánh đổi giữa lặp lại mô hình và lặp lại dữ liệu.

Tiếp tục, học tập liên tục đòi hỏi một cơ sở hạ tầng phát trực tuyến trưởng thành. Phần đào tạo của việc học liên tục có thể được thực hiện theo đợt, nhưng phần đánh giá trực tuyến yêu cầu phát trực tuyến. Nhiều kỹ sư lo lắng rằng phát trực tuyến rất khó và tốn kém. Đó là sự thật ba năm trước, nhưng công nghệ phát trực tuyến đã

trưởng thành đáng kể kể từ đó. Ngày càng có nhiều công ty cung cấp các giải pháp để giúp các công ty chuyển sang phát trực tuyến dễ dàng hơn, bao gồm Spark Streaming, Snowflake Streaming, Materialize, Decodable, Vectorize, v.v. Học liên tục là một vấn đề cụ thể đối với ML, nhưng nó phần lớn đòi hỏi một giải pháp cơ sở hạ tầng. Để có thể tăng tốc chu kỳ lặp lại và phát hiện lỗi trong các bản cập nhật mô hình mới một cách nhanh chóng, chúng ta cần thiết lập cơ sở hạ tầng của mình đúng cách. Điều này đòi hỏi nhóm khoa học dữ liệu / ML và nhóm nền tảng phải làm việc cùng nhau.

# TÀI LIỆU THAM KHẢO

[1] How to Choose a Feature Selection Method For Machine Learning

Access: <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>

[2] Correlation-based Feature Selection for Machine Learning

Access: <https://www.cs.waikato.ac.nz/~mhall/thesis.pdf>

[3] Tran Duc Tan, *Mô hình phân lớp Naive Bayes*, VIBLO

Access: <https://viblo.asia/p/mo-hinh-phan-lop-naive-bayes-vyDZO0A7lwj> [4] *Bài 7: Gradient Descent (phần 1/2)*, machinelearningcoban

Access:

<https://machinelearningcoban.com/2017/01/12/gradientdescent/> [5] Code Byte,

Stochastic Gradient Descent code from scratch in python Access:

[https://youtu.be/\\_g-rLDPbrgE](https://youtu.be/_g-rLDPbrgE)

[6] DataRobot, Introduction to Optimizer

Access: <https://www.datarobot.com/blog/introduction-to-optimizers/> [7] Deepak Battini, Adam Optimization Algorithm

Access: <https://github.com/tech-quantum/techquantum-demos/blob/master/Python/Adam%20Optimization%20Algorithm.ipynb>