

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



PHẠM PHÚC XUYỀN - 52100378

**BÁO CÁO CUỐI KỲ
NHẬP MÔN HỌC MÁY**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

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



PHẠM PHÚC XUYỀN - 52100378

BÁO CÁO CUỐI KỲ NHẬP MÔN HỌC MÁY

Người hướng dẫn
PGS. TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

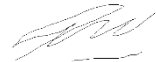
LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn chân thành và sâu sắc tới thầy Lê Anh Cường về sự hướng dẫn và sự hỗ trợ quý báu mà thầy đã dành cho em trong suốt quá trình em thực hiện báo cáo môn học "Nhập môn học máy". Em không thể không biểu thị lòng biết ơn sâu sắc của mình với những điểm mạnh mà em đã phát triển trong khóa học này dưới sự dẫn dắt của thầy. Thầy đã là nguồn động viên và truyền cảm hứng cho em trong việc nghiên cứu và ứng dụng kiến thức học máy. Bài giảng của thầy rất sâu sắc và cách thầy giảng dạy đã giúp em nắm vững kiến thức cơ bản và hiểu rõ hơn về các khía cạnh phức tạp của môn học này. Em cũng muốn cảm ơn thầy về sự kiên nhẫn và hỗ trợ mà thầy đã dành cho em trong quá trình nghiên cứu và viết báo cáo. Những lời góp ý và hướng dẫn của thầy đã giúp em cải thiện báo cáo của mình và phát triển kỹ năng viết. Cuối cùng, em muốn bày tỏ lòng biết ơn sâu sắc đối với thầy vì đã tạo điều kiện tốt để em phát triển không chỉ trong khía cạnh học thuật mà còn trong khía cạnh tư duy và phân tích. Khóa học "Nhập môn học máy" dưới sự hướng dẫn của thầy đã thực sự là một hành trình học tập đáng nhớ và giá trị đối với em.

TP. Hồ Chí Minh, ngày 24 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)



Phạm Phúc Xuyên

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 24 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)



Phạm Phúc Xuyên

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách hoạt động và so sánh các vấn đề với nhau.

MỤC LỤC

DANH MỤC HÌNH VẼ	vi
DANH MỤC CÁC CHỮ VIẾT TẮT.....	vii
CHƯƠNG 1. CÁC THUẬT TOÁN OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY.....	1
1.1 Thuật toán optimizer là gì?	1
1.2 Gradient Descent (GD)	1
<i>1.2.1 Gradient Descent là gì?</i>	<i>1</i>
<i>1.2.2 Learning rate:</i>	<i>2</i>
1.3 Stochastic Gradient Descent (SGD).....	2
<i>1.3.1 Stochastic Gradient Descent là gì?.....</i>	<i>2</i>
<i>1.3.2 Hoạt động của SGD.....</i>	<i>3</i>
1.4 Momentum	3
<i>1.4.1 Momentum là gì?</i>	<i>3</i>
1.5 Adagrad	3
<i>1.5.1 Adagrad là gì?</i>	<i>4</i>
1.6 RMSprop	5
<i>1.6.1 RMSprop là gì?</i>	<i>5</i>
1.7 Adam	5
<i>1.7.1 Adam là gì?</i>	<i>5</i>
1.8 So sánh ưu và nhược điểm của các thuật toán	6
CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION	7

2.1 Continual Learning.....	7
2.1.1 <i>Continual Learning</i> là gì?.....	7
2.1.2 Các loại học liên tục	7
2.1.3 Quá trình học liên tục	8
2.2 Test Production	8
2.2.1 <i>Test Production</i> là gì?.....	8
2.2.2 <i>Shadow deployment</i>	8
2.2.3 <i>A/B Testing</i>	9
2.2.4 <i>Canary Analysis</i>	9
2.2.5 <i>Interleaving Experiments</i>	10
TÀI LIỆU THAM KHẢO	11

DANH MỤC HÌNH VẼ

Hình 1. 1 Sự ảnh hưởng của Learning rate đến mô hình	2
--	---

DANH MỤC CÁC CHỮ VIẾT TẮT

GD	Gradient Descent
SGD	Stochastic Gradient Descent

CHƯƠNG 1. CÁC THUẬT TOÁN OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 Thuật toán optimizer là gì?

Thuật toán optimizer là một phần quan trọng trong quá trình huấn luyện mô hình học máy. Nhiệm vụ của thuật toán optimizer là tối ưu hoá các tham số của mô hình để đạt được hiệu suất tốt nhất trong việc dự đoán và học từ dữ liệu.

Các thuật toán optimizer được sử dụng để điều chỉnh các tham số của mô hình dựa trên độ lỗi (loss) được tính toán từ việc so sánh đầu ra dự đoán của mô hình với giá trị thực tế. Mục tiêu của thuật toán optimizer là tìm ra các giá trị tham số tối ưu nhằm giảm thiểu độ lỗi và cải thiện hiệu suất của mô hình.

Các thuật toán optimizer phổ biến như: Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adagrad, RMSprop, Adam

1.2 Gradient Descent (GD)

1.2.1 Gradient Descent là gì?

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.

- Gradient là tỷ lệ thay đổi của độ dốc (rate of inclination or declination) của một dốc. Trong ngữ cảnh toán học, Gradient của một hàm số đề cập đến đạo hàm của hàm số đó đối với từng biến số. Trong trường hợp của hàm số đơn biến, chúng ta thường sử dụng thuật ngữ "Derivative" thay vì "Gradient".
- Descent là từ viết tắt của descending, nghĩa là giảm dần.

Ý tưởng chung: Điều chỉnh các tham số để lặp đi lặp lại thông qua mỗi dữ liệu huấn luyện để giảm thiểu hàm chi phí.

Các bước thực thi:

1. Khởi tạo biến nội tại.
2. Đánh giá model dựa vào biến nội tại và hàm mất mát (Loss function).
3. Cập nhật các biến nội tại theo hướng tối ưu hàm mất mát (finding optimal points).
4. Lặp lại bước 2, 3 cho tới khi thỏa điều kiện dừng.
5. Công thức cập nhật cho GD có thể được viết là:

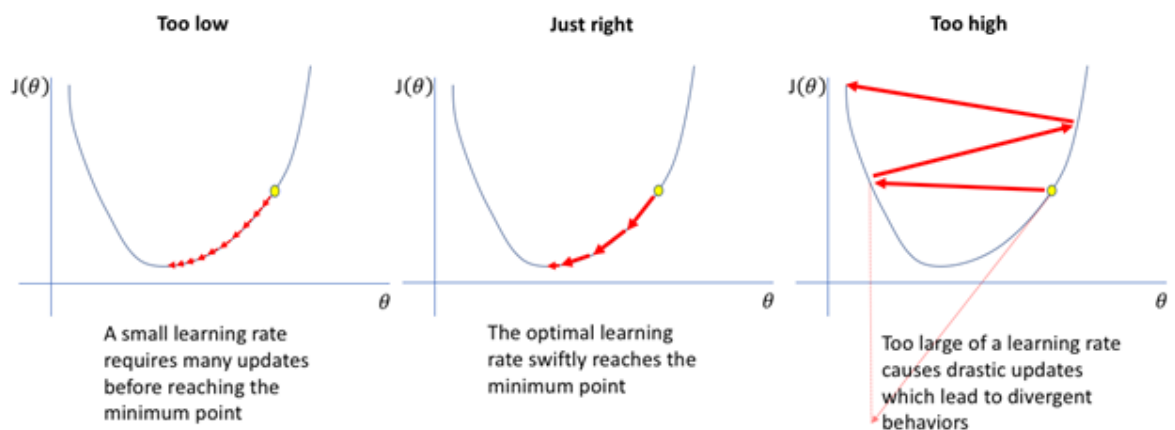
$$\theta = \theta - \eta \nabla f(\theta)$$

Trong đó, θ là tập các biến cần cập nhật, η là tốc độ học (learning rate), $\nabla f(\theta)$ là Gradient của hàm mất mát f theo tập θ . Tốc độ η xác định kích thước của các bước di chuyển đến giá trị cực tiểu (hoặc cực tiểu địa phương).

1.2.2 Learning rate:

Learning rate (tốc độ học) là một tham số quan trọng trong Gradient Descent.

Nếu learning rate quá nhỏ, thuật toán sẽ thực hiện nhiều bước để hội tụ và sẽ mất thời gian. Tuy nhiên, nếu learning rate quá lớn sẽ khiến thuật toán đi qua cực tiểu, và vượt hẳn ra ngoài khiến thuật toán không thể hội tụ được.



Hình 1. 1 Sự ảnh hưởng của Learning rate đến mô hình

1.3 Stochastic Gradient Descent (SGD)

1.3.1 Stochastic Gradient Descent là gì?

SGD là một biến thể của GD. Nó được cải tiến hơn là chỉ tính đạo hàm của hàm mất mát dựa trên một điểm dữ liệu x_i rồi cập nhật θ dựa trên đạo hàm này. Thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta} J(\theta_t; x^{(i)}; y^{(i)})$$

1.3.2 Hoạt động của SGD

SGD hoạt động giống với GD. Tuy nhiên, SGD sẽ lấy ngẫu nhiên 1 phần tử ở tập huấn luyện và thực thi tính toán lại vector độ dốc chỉ dựa trên 1 điểm dữ liệu, sau đó lặp đi lặp lại đến khi kết thúc.

1.4 Momentum

1.4.1 Momentum là gì?

Momentum là một thuật toán tối ưu, tương tự như GD nhưng được cải tiến lên để khắc phục việc nghiệm của GD rơi vào một điểm local minimum không mong muốn hay nói cách khác là tìm kiếm điểm cực tiểu của hàm mất mát.

Thuật toán Momentum tính toán một đại lượng gọi là "momentum" để điều chỉnh quá trình cập nhật tham số. Momentum được tính bằng cách kết hợp giữa độ dốc của điểm hiện tại và độ dốc của các điểm trước đó.

$$v_t = \text{gamma} * v_{t-1} + \text{learning_rate} * \text{gradient}$$

Trong đó:

- v_t là giá trị momentum tại thời điểm t .
- gamma là hệ số momentum, thường có giá trị khoảng 0.9.
- v_{t-1} là giá trị momentum tại thời điểm trước đó.
- learning_rate là tỷ lệ học, quyết định tốc độ cập nhật tham số.
- gradient là độ dốc của điểm hiện tại.

Quá trình cập nhật tham số được thực hiện bằng công thức:

$$\theta = \theta - v_t$$

1.5 Adagrad

1.5.1 Adagrad là gì?

Adagrad là một thuật toán tối ưu. Nó như các thuật toán trước, tuy nhiên, nó coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên tùy thuộc vào trọng số: tốc độ học thấp đối với trọng số tương ứng với các đặc trưng phổ biến, tốc độ học cao đối với các trọng số tương ứng với các đặc trưng ít phổ biến.

Ký hiệu g_t là gradient của hàm mất mát tại bước t . $g_{t,i}$ là đạo hàm riêng của hàm mất mát theo θ_i tại bước t .

$$g_{t,i} = \nabla_{\theta} J(\theta_{t,i})$$

Quy tắc cập nhật của Adagrad:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

Theo quy tắc cập nhật, Adagrad điều chỉnh tốc độ học η tại bước t tương ứng với trọng số θ_i xác định trên các gradient đã tính được theo θ_i .

Quy tắc cập nhật trên có thể viết dưới dạng tổng quát hơn:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

Trong đó:

- η : hằng số
- g_t : gradient tại thời điểm t
- ϵ : hệ số tránh lỗi
- G : ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vector tham số tại thời điểm t .

Adagrad thường khá hiệu đối với bài toán có dữ liệu phân mảnh. Tuy nhiên, hạn chế của Adagrad là các tổng bình phương ở mẫu số ngày càng lớn khiến tốc độ học ngày càng giảm và có thể tiệm cận đến giá trị 0 khiến cho quá trình huấn luyện

gần như đóng băng. Bên cạnh đó, giá trị tốc độ học η cũng phải được thực hiện một cách thủ công.

1.6 RMSprop

1.6.1 RMSprop là gì?

Root Mean Square Propagation (RMSprop) là một thuật toán tối ưu. Giúp cải thiện quá trình tối ưu hoá bằng cách điều chỉnh learning rate cho từng tham số mô hình một cách tự động và linh hoạt. Giải quyết được vấn đề tỷ lệ giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

$$E[g^2]_t = 0,9E[g^2]_{t-1} + 0,1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

1.7 Adam

1.7.1 Adam là gì?

Adam là một thuật toán tối ưu, Để cải thiện tốc độ huấn luyện và đạt được sử dụng nhanh chóng. Adam được viết tắt của Adaptive Moment Estimation. Và nó được thiết kế để tính toán tỷ lệ học tập tương ứng cho với mỗi trọng số. Adam không chỉ lưu trữ trung bình bình phương các gradient trước đó mà còn lưu cả giá trị trung bình mô-men m_t . Các giá trị m_t và v_t được tính bởi công thức:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

trong đó β_1 và β_2 là các trọng số không âm, thường được chọn là $\beta_1 = 0.9$ và $\beta_2 = 0.999$. Nếu khởi tạo m_t và v_t là các vector 0, các giá trị này có khuynh hướng nghiêng về 0, đặc biệt là khi β_1 và β_2 xấp xỉ bằng 1. Do vậy, để khắc phục, các giá trị này được ước lượng bằng cách:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Sau đó cập nhật các trọng số theo công thức:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \quad \epsilon \text{ thường bằng } 10^{-8}$$

1.8 So sánh ưu và nhược điểm của các thuật toán

	Ưu điểm	Nhược điểm
Gradient Descent	<ul style="list-style-type: none"> - Hiệu suất tính toán cao - Hội tụ đến điểm cực tiểu của hàm - Dễ dàng triển khai 	<ul style="list-style-type: none"> - Rơi vào điểm cực tiểu cục bộ - Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate - Tốc độ hội tụ chậm
SGD	<ul style="list-style-type: none"> - Giải quyết được cơ sở dữ liệu lớn - Tốc độ nhanh - Tiết kiệm bộ nhớ 	<ul style="list-style-type: none"> - Độ nhiễu cao - Không đảm bảo tìm được điểm cực tiểu toàn cục
Momentum	<ul style="list-style-type: none"> - Giảm thời gian hội tụ - Vượt qua các điểm yên ngựa và địa phương tối thiểu - Giảm dao động 	<ul style="list-style-type: none"> - Đòi hỏi tinh chỉnh tham số - Khó khăn trong việc lựa chọn giá trị momentum - Có khả năng bị mắc kẹt ở điểm tối thiểu cục bộ.
Adagrad	<ul style="list-style-type: none"> - Tự cập nhật learning rate - Điều chỉnh tỷ lệ học dựa trên tần suất 	<ul style="list-style-type: none"> - Tích phân bình phương. Điều này có thể dẫn đến việc hội tụ chậm hoặc dừng lại sớm hơn mong đợi - Vấn đề bộ nhớ
RSMprop	<ul style="list-style-type: none"> - Hội tụ nhanh - Ổn định học - Số lượng siêu tham số ít 	<ul style="list-style-type: none"> - Tích phân bình phương. Điều này có thể dẫn đến việc hội tụ chậm hoặc dừng lại sớm hơn mong đợi - Vấn đề bộ nhớ

	- Hiệu suất tốt trên các vấn đề phi lỗi	
Adam	- Tốc độ học hiệu quả - Phù hợp với các tập dữ liệu lớn - Ổn định với siêu tham số	- Yêu cầu bộ nhớ lớn - Nhạy cảm với siêu tham số

CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual Learning

2.1.1 Continual Learning là gì?

Continual Learning (học liên tục), còn được gọi là học máy liên tục (Continuous Machine Learning – CML), là quá trình mà một mô hình học từ dòng dữ liệu mới mà không cần phải được huấn luyện lại. Khác với các mô hình học máy truyền thống được huấn luyện trên một tập dữ liệu tĩnh và yêu cầu huấn luyện định kỳ, các mô hình học liên tục cập nhật các tham số của mình theo phân phối dữ liệu mới, cho phép chúng duy trì tính liên quan và thích ứng với tính chất động của dữ liệu thực tế.

2.1.2 Các loại học liên tục

Có nhiều loại học máy:

- Học tăng dần (incremental learning)
- Học chuyển giao (transfer learning)
- Học suốt đời (life learning)
- Lặp lại kinh nghiệm (experience replay)
- Kỹ thuật điều chỉnh (regularization)

Sự lựa chọn học máy phụ thuộc vào các yếu tố như dữ liệu, kiến trúc mô hình, hiệu suất mong muốn, độ phức tạp của nhiệm vụ và tài nguyên tính toán có sẵn.

Thường thì việc kết hợp nhiều phương pháp được thực hiện để nâng cao khả năng học tập.

2.1.3 Quá trình học liên tục

Quá trình học liên tục bao gồm những nguyên tắc mô hình hoá quen thuộc như tiền xử lý, lựa chọn mô hình, điều chỉnh tham số, đào tạo, triển khai và giám sát. Ngoài ra, hai bước bổ sung được yêu cầu: luyện tập lại dữ liệu (data rehearsal) và thực hiện một chiến lược học liên tục. Các bước này đảm bảo rằng mô hình học hiệu quả từ các dòng dữ liệu mới dựa trên ứng dụng và ngữ cảnh của nhiệm vụ dữ liệu.

2.2 Test Production

2.2.1 Test Production là gì?

Thử nghiệm trong sản xuất là quá trình kiểm thử và đánh giá mô hình học máy khi nó được triển khai và sử dụng trong môi trường sản xuất thực tế. Mục tiêu của thử nghiệm trong sản xuất là đảm bảo rằng mô hình không chỉ hoạt động hiệu quả trên dữ liệu kiểm thử ngoại tuyến mà chúng ta sử dụng để đào tạo và đánh giá ban đầu, mà còn giữ được hiệu suất và ổn định khi đối mặt với dữ liệu thực tế trong quá trình triển khai.

Có những kỹ thuật để giúp chúng ta đánh giá các mô hình của mình trong quá trình sản xuất một cách an toàn. Chúng ta sẽ tìm hiểu các kỹ thuật sau: triển khai bóng (shadow deployment), thử nghiệm A/B (A/B testing), Phát hành hoàng yến (canary analysis) và thí nghiệm xen kẽ (interleaving experiments)

2.2.2 Shadow deployment

Đây là một phương pháp an toàn để triển khai mô hình mới hoặc cập nhật phần mềm, giúp giảm thiểu rủi ro.

Việc triển khai Shadow hoạt động như sau:

1. Triển khai mô hình ứng cử viên song song với mô hình hiện có.
2. Đối với mỗi yêu cầu đến, hãy định tuyến nó đến cả hai mô hình để đưa ra dự đoán, nhưng chỉ phục vụ dự đoán của mô hình hiện có cho người dùng.

3. Ghi lại các dự đoán từ mô hình mới cho mục đích phân tích.

Tuy nhiên, kỹ thuật này khá đắt tiền. Nó tăng gấp đôi số lượng dự đoán mà hệ thống bạn phải tạo ra, điều này thường có nghĩa là tăng gấp đôi chi phí tính toán suy luận của bạn.

2.2.3 A/B Testing

Đây là một phương pháp so sánh hiệu suất giữa hai hoặc nhiều biến thể của một đối tượng, thường sử dụng trong việc đánh giá các mô hình ML. Mục tiêu là xác định xem có sự khác biệt đáng kể nào giữa các biến thể hay không, thông qua việc so sánh các chỉ số hoặc mục tiêu quan trọng.

Trong ngữ cảnh học máy và triển khai mô hình, thử nghiệm A/B thường được áp dụng để so sánh hiệu suất giữa mô hình hiện tại (biến thể A) và một mô hình mới (biến thể B).

Dưới đây là hoạt động của thử nghiệm A/B:

1. Triển khai mô hình ứng cử viên và mô hình hiện có
2. Ngẫu nhiên hoá lưu lượng truy cập
3. Theo dõi và đánh giá hiệu suất
4. Xác định mô hình tốt hơn
5. Kiểm soát yếu tố ngẫu nhiên và chiều dài thời gian

Thử nghiệm A/B mang lại một cách chặt chẽ để so sánh hiệu suất giữa các biến thể và đưa ra quyết định dựa trên cơ sở dữ liệu có độ tin cậy thống kê.

2.2.4 Canary Analysis

Đây là một kỹ thuật để giảm nguy cơ giới thiệu một phiên bản phần mềm mới trong sản xuất bằng cách từ từ triển khai thay đổi cho một tập hợp con người dùng trước khi triển khai nó cho toàn bộ cơ sở hạ tầng và làm cho nó có sẵn cho mọi người.

Phát hành Canary hoạt động như sau:

1. Triển khai mô hình ứng cử viên cùng với mô hình hiện có. Mô hình ứng cử viên được gọi là chim hoàng yến.
2. Một phần lưu lượng truy cập được chuyển đến mô hình ứng cử viên.

3. Nếu hiệu suất của nó đạt yêu cầu, hãy tăng lưu lượng truy cập đến mô hình ứng cử viên.
4. Nếu không, hãy hủy bỏ chim hoàng yến và định tuyến tất cả lưu lượng truy cập trở lại mô hình hiện có.
5. Dừng lại khi chim hoàng yến phục vụ tất cả lưu lượng truy cập (mô hình ứng cử viên đã thay thế mô hình hiện có) hoặc khi chim hoàng yến bị hủy bỏ.

Hiệu suất của mô hình ứng cử viên được đo lường dựa trên hiệu suất của mô hình hiện có theo các số liệu mà chúng ta quan tâm. Nếu các chỉ số chính của mô hình ứng cử viên suy giảm đáng kể, hoàng yến sẽ bị hủy bỏ và tất cả lưu lượng truy cập sẽ được chuyển đến mô hình hiện có.

2.2.5 Interleaving Experiments

Đây là phương pháp đánh giá mô hình bằng cách đưa ra đề xuất từ nhiều mô hình có người dùng và đo lường hiệu suất của chúng dựa trên sự tương tác của người dùng. Thay vì chia người dùng thành hai nhóm riêng biệt như trong thử nghiệm A/B, các thí nghiệm xen kẽ cho phép mỗi người dùng trải nghiệm đề xuất từ cả hai mô hình.

Cụ thể, trong một thí nghiệm xen kẽ, mỗi lượt đề xuất được chia thành 2 phần: một phần từ mô hình A và một phần từ mô hình B. Người dùng sẽ tiếp xúc với cả hai đề xuất và có thể tương tác với chúng. Hiệu suất của mỗi mô hình được đánh giá dựa trên chỉ số kết quả của tương tác người dùng, chẳng hạn như tỷ lệ nhấp vào, doanh số bán hàng, hoặc bất kỳ chỉ số quan trọng khác.

TÀI LIỆU THAM KHẢO

Tiếng Việt

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

Machine Learning cơ bản. (2017). Bài 7: Gradient Descent (phần 1/2). <https://machinelearningcoban.com/2017/01/12/gradientdescent/>

Machine Learning cơ bản. (2017). Bài 7: Gradient Descent (phần 2/2). <https://machinelearningcoban.com/2017/01/16/gradientdescent2/>

Nguyễn Đình Quý. (2021). 4.Gradient Descent.

<https://ndquy.github.io/posts/gradient-descent-2/>

Tiếng Anh

O'Reilly Media, Inc, Chip Huyen. (2022). *Designing Machine Learning System, Chapter 9*

Rahul Agarwal. (2023). Complete Guide to the Adam Optimization Algorithm. <https://builtin.com/machine-learning/adam-optimization>

Deepchecks Glossary. (2023). RMSprop.

<https://deepchecks.com/glossary/rmsprop/>

Opengenius. (2023). Adam Optimizer. <https://iq.opengenus.org/adam-optimizer/>

Sergio Rodríguez. (2023). Chapter 9 – Continual learning and test in production.<https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md#Continual%20Learning>