TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN PHÁT TRIỂN ỨNG DỤNG WEB VỚi NODEJS

WEBSITE QUẢN LÝ CỬA HÀNG ĐIỆN THOẠi VÀ PHỤ KIỆN

Người hướng dẫn: **THS LÊ ANH CƯỜNG**

Người thực hiện: **TRƯƠNG BỈNH THUẬN- 52100322**

Lớp : **21050301**

Khoá : **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN PHÁT TRIỂN ỨNG DỤNG WEB VỚi NODEJS

WEBSITE QUẢN LÝ CỬA HÀNG ĐIỆN THOẠi VÀ PHỤ KIỆN

Người hướng dẫn: **THS LÊ ANH CƯỜNG**

Người thực hiện: **TRƯƠNG BỈNH THUẬN- 52100322**

Lớp : **21050301**

Khoá : **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc đến tất cả những người đã hỗ trợ và đóng góp cho đồ án này trong suốt quá trình học môn Nhập môn Học máy. Đồ án này đã mang lại cho chúng em những trải nghiệm quý báu và kiến thức mới mẻ.

Đầu tiên và quan trọng nhất, chúng em xin gửi lời cảm ơn đến giảng viên hướng dẫn cũng là giảng viên của môn học, thầy Lê Anh Cường, vì đã cung cấp kiến thức và hỗ trợ chúng em trong suốt quá trình nghiên cứu và phát triển. Những lời chỉ dẫn của thầy đã giúp chúng em tiến bộ và hiểu rõ hơn về phát triển ứng dụng web.

Chúng em cũng xin gửi lời cảm ơn đến bạn bè trong lớp học vì đã chia sẻ kiến thức và kinh nghiệm cùng nhau. Những cuộc thảo luận và sự hỗ trợ tận tâm đã giúp chúng em vượt qua những khó khăn trong quá trình học tập và thực hiện dự án.

Chúng em tự hào về sự nỗ lực cùng nhau để hoàn thành đồ án này, tuy nhiên với kinh nghiệm non nớt không thể tránh khỏi những thiếu sót, chúng em mong thầy thông cảm và góp ý để chúng em tiếp tục nỗ lực cải thiện và phát triển trong tương lai.

Xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 21 tháng 10 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Thuận

Trương Bỉnh Thuận

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của Giảng viên bộ môn. 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 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 chúng 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 21 tháng 04 năm 2023

(ký tên và ghi rõ họ tên)

Thuận

Trương Bỉnh Thuận

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Báo cáo này gồm 2 phần chính. Phần đầu trình bày về Optimazer bao gồm khái niệm, các thuật toán tối ưu. Phần hai trình bày về Continual Learning và Test Production, phần này nóI rõ về các kháI niệm và cơ chế hoạt động cũng như ảnh hưởng của nó đốI với trước khi sản xuất rộng rãi.

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 - OPTIMIZER

1.1 Khái niệm Optimizer

Optimizer là các thuật toán tối ưu hóa được sử dụng để cập nhật trọng số của mạng nơ-ron trong quá trình huấn luyện. Điều này giúp mạng nơ-ron học được từ dữ liệu huấn luyện và cải thiện độ chính xác của dự đoán.

Ứng dụng của optimizer trong mạng nơ-ron rất quan trọng vì quá trình huấn luyện mạng nơ-ron là quá trình tìm kiếm các trọng số tối ưu để mô hình có thể dự đoán chính xác. Các thuật toán optimizer như Gradient Descent, Stochastic Gradient Descent (SGD), Adam, RMSprop, và nhiều thuật toán khác giúp cập nhật các trọng số của mạng nơ-ron sao cho mô hình có thể học được từ dữ liệu một cách hiệu quả.

Tại sao cần phải sử dụng optimizer trong học máy và mạng nơ-ron?

- Học hiệu quả: Optimizer giúp mô hình học từ dữ liệu một cách hiệu quả hơn bằng cách điều chỉnh các trọng số của mạng nơ-ron để giảm thiểu sai số.

- Tối ưu hóa hàm mất mát: Trong quá trình huấn luyện, mục tiêu là tối thiểu hóa hàm mất mát (loss function), và optimizer giúp điều này bằng cách điều chỉnh trọng số để đạt được giá trị tối thiểu của hàm mất mát.

- Điều chỉnh thông số: Các thuật toán optimizer cũng cho phép điều chỉnh các siêu tham số như tốc độ học (learning rate), momentum, và độ chính xác để tối ưu hóa quá trình huấn luyện.

Tóm lại, sử dụng optimizer là cực kỳ quan trọng trong học máy và mạng nơ-ron để đảm bảo mô hình có thể học được từ dữ liệu một cách hiệu quả và đạt được độ chính xác cao trong việc dự đoán.

1.2 Các thuật toán tối ưu

1.2.1 Gradient Descent (GD)

Trong các bài toán tối ưu, chúng ta thường tìm giá trị nhỏ nhất của 1 hàm số nào đó, mà hàm số đạt giá trị nhỏ nhất khi đạo hàm bằng 0. Nhưng đâu phải lúc nào đạo

hàm hàm số cũng được, đối với các hàm số nhiều biến thì đạo hàm rất phức tạp, thậm chí là bất khả thi. Nên thay vào đó người ta tìm điểm gần với điểm cực tiểu nhất và xem đó là nghiệm bài toán.

Gradient Descent là một phương pháp cơ bản nhất, cố gắng đi theo hướng ngược với đạo hàm của hàm mất mát để cập nhật các trọng số.

Cơ chế hoạt động của Gradient Descent là:

- Bước 1: KhởI tạo trọng số ban đầu: Trọng số của mô hình được khởi tạo ngẫu nhiên.

- Bước 2: Tính đạo hàm của hàm mất mát: Gradient của hàm mất mát $\frac{\partial L}{\partial \omega}$ (đạo hàm của hàm mất mát theo từng trọng số) được tính toán thông qua quá trình lan truyền ngược (backpropagation) trong mạng nơ-ron.

- Bước 3 Cập nhật trọng số: Trọng số được cập nhật theo hướng ngược với đạo hàm (đi ngược theo gradient) để giảm thiểu hàm mất mát:

+ Công thức: $\omega_{\text{new}} = \omega_{\text{old}} - \alpha * \frac{\partial L}{\partial \omega}$

Trong đó:

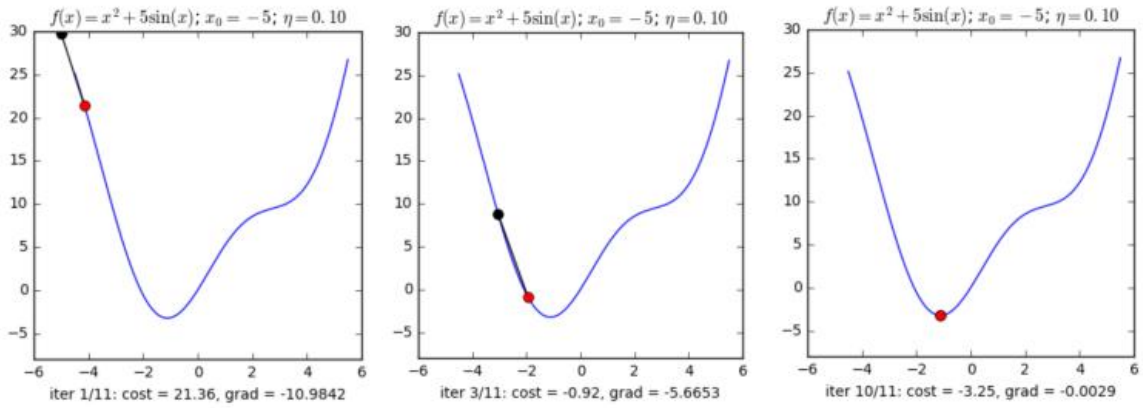
* ω_{new} là giá trị mớI của trọng số sau khi cập nhật

* ω_{old} là giá trị cũ của trọng số trước khi cập nhật

* α là tốc độ học, quyết định độ lớn của bước cập nhật

* $\frac{\partial L}{\partial \omega}$ là đạo hàm riêng của hàm mất mát L theo từng trọng số ω

- Bước 4: Lặp lại quá trình: Quá trình cập nhật trọng số được lặp lại qua nhiều vòng lặp (epochs) cho đến khi hàm mất mát hội tụ đến giá trị tối thiểu hoặc đạt đến một điều kiện dừng khác.



Hình 1: Ví dụ về cơ chế hoạt động của Gradient Descent

Ưu điểm:

- Dễ hiểu, giải quyết được các vấn đề tốI ưu model neural network bằng cách cập nhật các trọng số.

Nhược điểm:

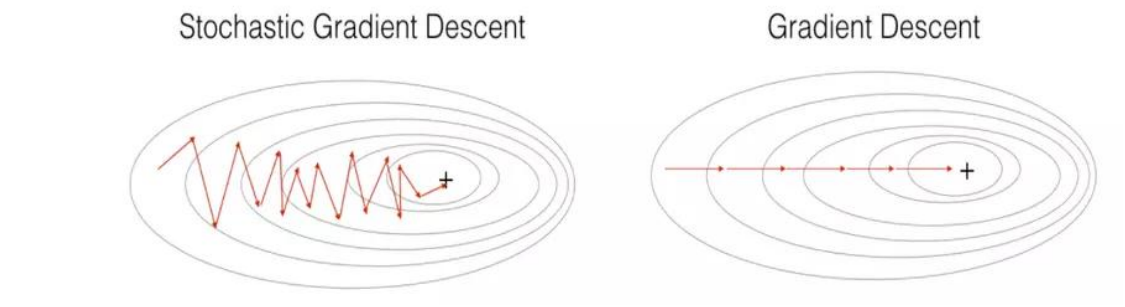
- Còn phụ thuộc vào trọng số được khởI tạo ban đầu.

- Còn phụ thuộc vào learning rate.

- Không đáp ứng được online learning. Vì khi có dữ liệu mớI phảI đạo hàm lạI toàn bộ dữ liệu.

1.2.2 Stochastic Gradient Descent (SGD)

Stochastic là 1 biến thể của Gradient Descent . Thay vì sau mỗi epoch chúng ta sẽ cập nhật trọng số (Weight) 1 lần thì trong mỗi epoch có N điểm dữ liệu chúng ta sẽ cập nhật trọng số N lần. Nhìn vào 1 mặt , SGD sẽ làm giảm đi tốc độ của 1 epoch. Tuy nhiên nhìn theo 1 hướng khác,SGD sẽ hội tụ rất nhanh chỉ sau vài epoch. Công thức SGD cũng tương tự như GD nhưng thực hiện trên từng điểm dữ liệu.



Hình 2: Minh hoạ về SGD

Nhìn vào 2 hình trên, ta thấy SGD có đường đi khá là zig zắc , không mượt như GD. Dễ hiểu điều đó vì 1 điểm dữ liệu không thể đại diện cho toàn bộ dữ liệu. Ở đây, GD có hạn chế đối với cơ sở dữ liệu lớn (vài triệu dữ liệu) thì việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh. SGD ra đời để giải quyết vấn đề online learning của GD, vì mỗi lần thêm dữ liệu mới vào chỉ cần cập nhật trên 1 điểm dữ liệu đó thôi, phù hợp với online learning. Online learning chính là quá trinh được thêm mới thì không cần đạo hàm trên toàn bộ dữ liệu.

Cơ chế hoạt động của Stochastic Gradient Descent:

- Bước 1: KhởI tạo trọng số ban đầu: Trọng số của mô hình được khởi tạo ngẫu nhiên.

- Bước 2: Chọn Mẫu Ngẫu Nhiên: Từ tập dữ liệu huấn luyện, chọn ngẫu nhiên một mẫu dữ liệu. Đây có thể là một điểm dữ liệu đơn hoặc một mini-batch (một lượng nhỏ các điểm dữ liệu).

- Bước 3: Tính Toán Gradient: Tính toán gradient của hàm mất mát tại điểm dữ liệu đã chọn. Gradient này chỉ ra hướng và mức độ thay đổi của hàm mất mát khi thay đổi các tham số của mô hình.

- Bước 4: Cập nhật trọng số: Trọng số được cập nhật theo hướng ngược với đạo hàm (đi ngược theo gradient) để giảm thiểu hàm mất mát:

+ Công thức: $\omega_{\text{new}} = \omega_{\text{old}} - \alpha * \frac{\partial L}{\partial \omega}$

Trong đó:

* ω_{new} là giá trị mớI của trọng số sau khi cập nhật

* ω_{old} là giá trị cũ của trọng số trước khi cập nhật

* α là tốc độ học, quyết định độ lớn của bước cập nhật

* $\frac{\partial L}{\partial \omega}$ là đạo hàm riêng của hàm mất mát L theo từng trọng số ω

- Bước 5: Lặp lại quá trình: Tiếp tục lựa chọn mẫu ngẫu nhiên, tính toán gradient và cập nhật trọng số cho đến khi đạt được điều kiện dừng được đặt ra (ví dụ: số lượng epoch, đạt đủ độ chính xác mong muốn, hoặc hàm mất mát không thay đổi đáng kể nữa).

Ưu điểm:

- Dễ hiểu, giải quyết được các vấn đề đốI vớI dữ liệu lớn mà GD không làm được.

Nhược điểm:

- Còn phụ thuộc vào trọng số được khởI tạo ban đầu.

- Còn phụ thuộc vào learning rate.

1.2.3 Momentum

Momentum trong Gradient Descent là một cách điều chỉnh quá trình cập nhật trọng số của mô hình. Nó không chỉ dựa vào gradient hiện tại mà còn tính đến đà từ các bước trước, giúp mô hình di chuyển mạnh mẽ hơn và tránh được sự dao động không mong muốn trong quá trình tối ưu hóa.

Vai trò của Momentum là tăng tốc độ hội tụ: Momentum giúp mô hình di chuyển nhanh hơn bằng cách giữ lại và kết hợp gradient từ các bước trước, điều này giúp tránh được việc mô hình bị kẹt ở các điểm cực tiểu cục bộ.

Cơ Chế Hoạt Động:

- Bước 1: Khởi Tạo Momentum: Bắt đầu với việc khởi tạo giá trị ban đầu của momentum, thường là 0 hoặc một giá trị nhỏ.

- Bước 2: Tính Toán Gradient: Tính toán gradient của hàm mất mát tại vị trí hiện tại của các tham số mô hình.

- Bước 3: Cập Nhật Momentum: Công thức cập nhật của momentum:

$$v_t = \beta * v_{t-1} + \alpha * \delta(t)$$

Trong đó:

* v_t là giá trị momentum tại bước thời gian t.

* β là gamma hay hệ số momentum, xác định mức độ giữ lại momentum từ các bước trước.

* α là tỷ lệ học tập (learning rate).

* $\delta(t)$ là gradient của hàm mất mát tại điểm hiện tại.

- Bước 4: Cập Nhật Trọng Số Mô Hình: Sử dụng giá trị momentum đã tính được để cập nhật trọng số của mô hình:

$$\theta_t = \theta_{t-1} - v_t$$

- Bước 5: Lặp Lại Quá Trình: Tiếp tục lặp lại quá trình trên với các bước tiếp theo để hội tụ đến điểm tối ưu.

Ưu điểm:

- Tăng Tốc Độ Hội Tụ: Momentum giúp mô hình hội tụ nhanh hơn bằng cách tích hợp đà từ các bước trước, điều này thường làm giảm số lượng epoch cần thiết để đạt được điểm tối ưu.

- Tránh Được Các Điểm Cực Tiểu Cục Bộ: Đà tích lũy giúp mô hình vượt qua các "đồi" nhỏ và tránh rơi vào các điểm cực tiểu cục bộ, do đó giúp tìm được điểm tối ưu toàn cục tốt hơn.

Nhược điểm:

- Khó Điều Chỉnh Tham Số Momentum: Hệ số momentum cần được điều chỉnh cẩn thận để tránh hiện tượng mô hình không hội tụ hoặc hội tụ quá chậm. Việc điều chỉnh tham số này đôi khi có thể khá khó khăn.

- Khả Năng Lạc Hậu (Overshooting): Trong một số trường hợp, Momentum có thể tạo ra việc mô hình "vượt quá" điểm tối ưu mong muốn và tiếp tục di chuyển với đà mạnh mẽ vượt qua điểm cực tiểu, dẫn đến việc không hội tụ

1.2.4 Adagrad

Adagrad là một phương pháp tối ưu hóa adaptive learning rate trong quá trình huấn luyện mô hình. Đặc điểm nổi bật của nó là việc điều chỉnh tỷ lệ học tập (learning rate) tự động cho từng tham số dựa trên lịch sử của gradient cho tham số đó.

Adagrad điều chỉnh learning rate cho mỗi tham số dựa trên tổng bình phương của gradient đã tính toán. Nó ưu tiên việc cập nhật các tham số với gradient lớn bằng cách giảm learning rate và ngược lại, tăng learning rate cho các tham số với gradient nhỏ.

Vai trò của Adagrad tự điều chỉnh learning rate: Adagrad giúp mô hình tự động điều chỉnh learning rate cho từng tham số, làm giảm cần thiết phải chọn learning rate thủ công. Điều này rất hữu ích khi làm việc với dữ liệu có tính chất thay đổi, không đồng nhất, hoặc khi gradient thay đổi lớn giữa các tham số.

Cơ Chế Hoạt Động:

- Bước 1: Điều Chỉnh Learning Rate: Adagrad điều chỉnh tỷ lệ học tập (learning rate) cho từng tham số của mô hình dựa trên lịch sử của gradient tại tham số đó. Nó tự động điều chỉnh learning rate dựa trên việc tích lũy bình phương gradient.

- Bước 2: Bình Phương Gradient: Adagrad lưu trữ tổng bình phương của gradient đã tính toán cho mỗi tham số. Mỗi khi tính toán gradient mới, nó sẽ thêm bình phương của gradient đó vào tổng bình phương.

- Bước 3 Cập Nhật Learning Rate: Learning rate được cập nhật dựa trên căn bậc hai của tổng bình phương của gradient tích lũy. Điều này có nghĩa là các tham số với gradient lớn sẽ có learning rate giảm, trong khi các tham số với gradient nhỏ sẽ có learning rate tăng.

Công thức:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \times \delta(t)$$

Trong đó:

* α : tỷ lệ học tập (learning rate)

* $\delta(t)$: gradient tại thời điểm t

* ϵ : hệ số tránh lỗi (chia cho mẫu bằng 0)

* G_t : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

Ưu Điểm:

- Tự Điều Chỉnh Learning Rate: Adagrad tự động điều chỉnh learning rate cho từng tham số, giúp mô hình học tốt hơn trên các tham số có gradient thay đổi mạnh và giảm được sự nhạy cảm với việc điều chỉnh learning rate thủ công.

Nhược Điểm:

- Yếu điểm của Adagrad là tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

1.2.5 RMSprop

RMSprop là một phương pháp tối ưu hóa trong deep learning, tương tự như Adagrad, nhưng với sự cải tiến để giảm nhược điểm về learning rate giảm quá nhanh.

Tương tự như Adagrad, RMSprop cũng sử dụng adaptive learning rate, nhưng thay vì tính tổng bình phương của toàn bộ gradient, nó chỉ xem xét một khoảng lịch sử gần đây của gradient.

Vai trò của RMSprop là tự điều chỉnh learning rate: RMSprop giúp mô hình tự động điều chỉnh learning rate cho từng tham số dựa trên lịch sử gần đây của gradient, giảm thiểu tác động của gradient cũ và tối ưu hóa quá trình học.

Cơ Chế Hoạt Động:

- Bước 1: Tính Toán Trung Bình Của Bình Phương Gradient: Tại mỗi bước thời gian t, RMSprop tính trung bình động của bình phương gradient với một hệ số giảm dần ρ.

$$E[g^2]_t = \beta \times E[g^2]_{t-1} + (1 - \beta) \times \delta(t)$$

Trong đó:

* $E[g^2]_t$ là giá trị trung bình của bình phương gradient tại bước thời gian t.

* β là hệ số giảm dần (thường là 0.9 hoặc 0.99).

* $E[g^2]_{t-1}$ là giá trị trung bình của bình phương gradient tại bước thời gian trước đó.

* $\delta(t)$ là gradient tại thời điểm t.

- Bước 2: Cập Nhật Learning Rate và Trọng Số: Công thức cập nhật của RMSprop sử dụng giá trị trung bình của bình phương gradient để điều chỉnh learning rate cho từng tham số.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \times \delta(t)$$

Trong đó:

* θ_{t+1} là giá trị cập nhật của tham số tại bước thời gian t+1.

* α là tỷ lệ học tập (learning rate).

* ϵ là một hằng số nhỏ (thường là 10^{-8}) để tránh việc chia cho 0.

* $E[g^2]_t$ là giá trị trung bình của bình phương gradient tại bước thời gian t.

* $\delta(t)$ là gradient tại thời điểm t.

Ưu Điểm:

- Ổn Định Learning Rate: RMSprop giúp ổn định quá trình học bằng cách giảm tác động của gradient cũ, giúp learning rate không giảm quá nhanh như Adagrad.

Nhược Điểm:

- RMSprop cải thiện vấn đề về learning rate giảm quá nhanh của Adagrad và là một trong những phương pháp tối ưu hóa phổ biến trong deep learning. Tuy nhiên, như mọi phương pháp, việc điều chỉnh các tham số cũng là yếu tố quan trọng để đạt hiệu quả tối ưu hóa cao nhất.

1.2.6 Adam

Adam (Adaptive Moment Estimation) là một phương pháp tối ưu hóa kết hợp các ưu điểm của RMSprop và Momentum để cải thiện quá trình hội tụ trong huấn luyện mô hình deep learning. Adam kết hợp cả RMSprop để điều chỉnh learning rate cho từng tham số dựa trên lịch sử gần đây của gradient và Momentum để giữ lại đà từ các bước trước.

Cơ Chế Hoạt Động:

- Bước 1: Tính toán Momentum và RMSprop: Tại mỗi bước thời gian t, Adam tính toán Momentum và RMSprop cho từng tham số:

$$M_t = \beta_1 * M_{t-1} + (1 - \beta_1) * \delta(t)$$

$$R_t = \beta_2 * R_{t-1} + (1 - \beta_2) * (\delta(t))^2$$

Trong đó:

* M_t là Momentum tại thời điểm t

* R_t là RMSprop tại thời điểm t

* β_1 là gamma hay hệ số momentum, xác định mức độ giữ lại momentum từ các bước trước

* β_2 là hệ số giảm dần (thường là 0.9 hoặc 0.99)

* $\delta(t)$ là gradient tại thời điểm t

- Bước 2: Tính Bias Correction: Adam sử dụng bias correction để điều chỉnh các giá trị Momentum và RMSprop ước lượng ban đầu:

$$\widehat{M_t} = \frac{M_t}{1 - \beta_1^t}$$

$$\widehat{R_t} = \frac{R_t}{1 - \beta_2^t}$$

Trong đó:

* M_t là Momentum tại thời điểm t

* R_t là RMSprop tại thời điểm t

* β_1^t là gamma hay hệ số momentum tại thời điểm t, xác định mức độ giữ lại momentum từ các bước trước

* β_2^t là hệ số giảm dần tại thời điểm t (thường là 0.9 hoặc 0.99)

* $\delta(t)$ là gradient tại thời điểm t

- Bước 3: Cập Nhật Trọng Số Mô Hình: Adam sử dụng các ước lượng đã được điều chỉnh để cập nhật trọng số của mô hình:

$$\theta_{t+1} = \theta_t - \frac{\alpha \times \widehat{M_t}}{\sqrt{\widehat{R_t}} + \epsilon}$$

Trong đó:

- θ_{t+1} là giá trị cập nhật của tham số tại bước thời gian t+1.
- α là tỷ lệ học tập (learning rate).
- ϵ là một hằng số nhỏ (thường là 10^{-8}) để tránh việc chia cho 0.

Ưu Điểm:

- Kết Hợp Cả RMSprop và Momentum: Adam kết hợp cả hai phương pháp giúp cải thiện quá trình hội tụ và tối ưu hóa mô hình một cách hiệu quả.

- Hiệu Quả và Phổ Biến: Adam thường được sử dụng trong deep learning với hiệu suất tốt trên nhiều loại mô hình và dữ liệu.

Nhược Điểm:

- Cần Điều Chỉnh Tham Số: Adam cũng cần điều chỉnh các tham số như β_1, β_2 và learning rate α để đạt hiệu quả tối ưu hóa cao nhất cho từng bài toán cụ thể.

CHƯƠNG 2 - CONTINUAL LEARNING VÀ TEST PRODUCTION

2. 1 Continual Learning

2.1.1 Sơ lược về Continual Learning

Continual Learning là ý tưởng cập nhật mô hình khi có dữ liệu mới, giúp mô hình thích nghi với sự thay đổi trong phân phối dữ liệu. Phương pháp này không chỉ giúp mô hình duy trì độ chính xác mà còn giải quyết vấn đề của sự thay đổi nhanh chóng trong môi trường dữ liệu.

Nó không phải là lớp đặc biệt của thuật toán Học Máy cho phép cập nhật tăng dần mô hình mỗi khi có một điểm dữ liệu mới. Ví dụ của lớp thuật toán đặc biệt này là việc cập nhật Sequential Bayesian và KNN classifiers. Thuật toán này được gọi là "online learning algorithms".

Khái niệm về Continual Learning có thể áp dụng cho bất kỳ thuật toán Học Máy giám sát nào, không chỉ là một lớp đặc biệt.

Continual Learning không đồng nghĩa với việc bắt đầu một công việc huấn luyện lại mỗi khi có một mẫu dữ liệu mới. Trên thực tế, điều này nguy hiểm vì làm cho mạng nơ-ron dễ bị quên thông tin.

2.1.2 Tại sao Continual Learning cần thiết?

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

- Các trường hợp sử dụng có thể xảy ra những thay đổi nhanh chóng và bất ngờ: Các trường hợp sử dụng như ride-sharing (chia sẻ chuyến đi). Ví dụ: có thể có một buổi hòa nhạc ở một khu vực ngẫu nhiên vào Thứ Hai bất kì và "mô hình học máy dùng để dự đoán buổI hoà nhạc" có thể không được trang bị tốt để xử lý buổi hòa nhạc đó.

- Các trường hợp không thể lấy dữ liệu huấn luyện cho một sự kiện cụ thể. Một ví dụ cho điều này là các mô hình thương mại điện tử trong dịp Black Friday hay một số sự kiện sale khác chưa từng được thử trước đây. Rất khó để thu thập dữ liệu lịch sử để dự đoán hành vi của người dùng trong dịp Black Friday, vì vậy mô hình của bạn phải thích ứng cả ngày.

- Các trường hợp sử dụng nhạy cảm với vấn đề cold start. Sự cố này xảy ra khi mô hình của bạn phải đưa ra dự đoán cho người dùng mới (hoặc đã đăng xuất) không có dữ liệu lịch sử (hoặc dữ liệu đã lỗi thời). Nếu bạn không điều chỉnh mô hình của mình ngay khi nhận được một số dữ liệu từ người dùng đó, bạn sẽ không thể đề xuất những điều liên quan cho người dùng đó.

2.1.3 Stateless Retraining và Stateful Retraining

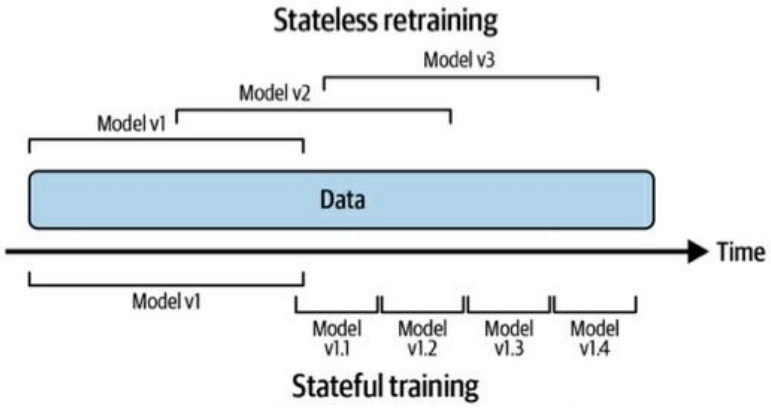


Figure 9-2. Stateless retraining versus stateful training

Hình 3: Hình minh hoạ về Stateless Retraining và Stateful Retraining

2.1.3.1 Stateless Retraining

Stateless Retraining là phương pháp huấn luyện lại mô hình từ đầu, sử dụng trọng số được khởi tạo ngẫu nhiên và dữ liệu mới hơn. Phương pháp này có thể dẫn đến việc một số dữ liệu đã được sử dụng để huấn luyện phiên bản mô hình trước đó.

Trong lĩnh vực Continual Learning, Lúc bắt đầu xây dựng model, họ đã sử dụng stateless retraining như là phương pháp khởi đầu. Phương pháp này đòi hỏi mô hình phải trải qua quá trình huấn luyện toàn diện mỗi khi cập nhật, đảm bảo tích hợp

dữ liệu mới nhất vào quá trình huấn luyện. Mặc dù hiệu quả, một trong những hạn chế của phương pháp này là có thể sử dụng lại một số điểm dữ liệu đã từng được sử dụng để huấn luyện trước đó, có thể gây ra sự trùng lặp trong quá trình học. Tuy nhiên, phương pháp này là một điểm khởi đầu hiệu quả cho việc tiếp cận lĩnh vực Continual Learning.

2.1.3.2 Stateful Retraining

Stateful Retraining (còn được gọi là fine-tuning hoặc incremental learning) là quá trình khởi tạo mô hình với trọng số từ vòng huấn luyện trước đó và tiếp tục huấn luyện bằng dữ liệu mới chưa từng được sử dụng trước đây.

Phương pháp này cho phép mô hình cập nhật với ít dữ liệu hơn đáng kể, giúp mô hình hội tụ nhanh hơn và sử dụng ít tài nguyên tính toán hơn. Sử dụng phương pháp này có thể giảm đến 45% năng lực tính toán khi sử dụng phương pháp này.

Lý thuyết, phương pháp này có thể loại bỏ hoàn toàn việc lưu trữ dữ liệu sau khi dữ liệu đã được sử dụng để huấn luyện (và để lại một khoảng thời gian an toàn). Tuy nhiên, trong thực tế, thường giữ lại tất cả dữ liệu mặc dù không còn cần thiết nữa.

Đôi khi, cần chạy stateless retraining với một lượng lớn dữ liệu để hiệu chuẩn lại mô hình.

Một khi cơ sở hạ tầng được thiết lập đúng cách, việc chuyển từ stateless retraining sang stateful training trở nên đơn giản.

Stateful Retraining chủ yếu được sử dụng để tích hợp dữ liệu mới vào một kiến trúc mô hình hiện có và cố định (tức là cải tiến dữ liệu), nếu bạn muốn thay đổi các đặc trưng hoặc kiến trúc của mô hình, bạn sẽ cần phải thực hiện một quá trình huấn luyện Stateless Retraining ở bước đầu tiên.

Một số nghiên cứu đã được tiến hành về cách chuyển trọng số từ một kiến trúc mô hình sang một kiến trúc mới (ví dụ: Net2Net knowledge transfer, model surgery). Tuy nhiên, hiện chưa có sự áp dụng rộng rãi của các kỹ thuật này.

2.1.4 Cơ chế hoạt động của Continual Learning

Trong Continual Learning có tổng cộng 4 giai đoạn:

- Giai đoạn 1: Manual, Stateless Retraining

+ Các mô hình chỉ được đào tạo lại khi đáp ứng hai điều kiện:

(1) hiệu suất của mô hình đã suy giảm đến mức hiện tại nó gây hại nhiều hơn là có lợi

(2) Có thời gian dư để cập nhật mô hình.

- Giai đoạn 2: Fixed schedule automated stateless retraining

+ Tại giai đoạn này thường xảy ra khi các mô hình chính của một lĩnh vực đã được phát triển và do đó ưu tiên của bạn không còn là tạo ra các mô hình mới, mà là duy trì và cải thiện các mô hình hiện tại. Việc ở lại ở giai đoạn 1 đã trở thành một vấn đề quá lớn để bị bỏ qua.

+ Điểm chuyển giai đoạn từ giai đoạn 1 sang giai đoạn 2 thường là một đoạn mã mà ai đó viết để chạy việc huấn luyện không có trạng thái định kỳ. Việc viết mã này có thể rất dễ dàng hoặc rất khó tùy thuộc vào số lượng phụ thuộc cần được phối hợp để huấn luyện một mô hình.

Các bước của giai đoạn này là:

1. Lấy dữ liệu.
2. Giảm hoặc tăng mẫu dữ liệu nếu cần thiết.
3. Trích xuất đặc trưng.
4. Xử lý và gắn nhãn để tạo dữ liệu huấn luyện.
5. Bắt đầu quá trình huấn luyện.
6. Đánh giá mô hình mới.
7. Triển khai mô hình.

Ngoài ra, cần có thêm 2 phần cơ sở hạ tầng để triển khai đoạn mã này:

1. Lịch trình

2. Một kho lưu trữ mô hình để tự động phiên bản hóa và lưu trữ tất cả các tác phẩm nghệ thuật cần thiết để tạo ra mô hình.

- Giai đoạn 3: Fixed schedule automated stateful training

+ Để đạt được điều này, cần điều chỉnh lại đoạn mã của mình và một cách để theo dõi dữ liệu và dòng thời gian mô hình. Một ví dụ đơn giản về phiên bản của mô hình:

+ V1 so với V2 là hai kiến trúc mô hình khác nhau cho cùng một vấn đề.

+ V1.2 so với V2.3 có nghĩa là kiến trúc mô hình V1 đang ở vòng lặp thứ hai của quá trình huấn luyện không có trạng thái đầy đủ và V2 đang ở vòng lặp thứ ba.

+ V1.2.12 so với V2.3.43 có nghĩa là đã có 12 quá trình huấn luyện có trạng thái được thực hiện trên V1.2 và 43 quá trình trên V2.3.

- Giai đoạn 4: Continual learning

+ Ở giai đoạn này, phần lịch trình cố định của các giai đoạn trước được thay thế bằng một cơ chế kích hoạt việc huấn luyện lại mô hình. Các kích hoạt có thể là:

* Dựa trên thời gian

* Dựa trên hiệu suất: ví dụ như hiệu suất giảm dưới x%.

* Dựa trên khối lượng dữ liệu: Tổng lượng dữ liệu được gắn nhãn tăng thêm 5%.

* Dựa trên sự chuyển động: ví dụ như khi phát hiện sự chuyển đổi "lớn" trong phân phối dữ liệu.

2.1.5 Thách thức của Continual Learning

Có ba thách thức chính trong việc triển khai Continual Learning:

1. Thách thức về truy cập dữ liệu mới:

- Để cập nhật mô hình hàng giờ, cần dữ liệu huấn luyện được gắn nhãn chất lượng hàng giờ.

- Vấn đề: Tốc độ đưa dữ liệu vào các kho dữ liệu.

- Các cách tiếp cận phổ biến bao gồm trích xuất dữ liệu trực tiếp từ luồng thời gian thực và xử lý dữ liệu.

2. Thách thức về tốc độ gắn nhãn:

- Tốc độ gắn nhãn mới thường là điểm bottleneck.

- Các kỹ thuật hỗ trợ gắn nhãn yếu, bán giám sát, hoặc tập trung nhanh gắn nhãn từ cộng đồng có thể được sử dụng.

3. Thách thức đánh giá:

- Áp dụng Continual Learning có rủi ro về thất bại mô hình nghiêm trọng.

- Đòi hỏi kiểm tra mô hình trước khi triển khai rộng rãi.

- Việc kiểm tra mất thời gian, làm giới hạn tần suất cập nhật mô hình nhanh nhất.

Ngoài ra, còn có thêm hai thách thức quan trọng:

- Thách thức về tỉ lệ thu phóng dữ liệu: Tính toán đặc trưng thường yêu cầu thu phóng, đòi hỏi truy cập vào thống kê dữ liệu toàn cầu như min, max, average và variance.

- Thách thức về thuật toán: Sử dụng những thuật toán đòi hỏi truy cập đầy đủ vào bộ dữ liệu để được huấn luyện, không thể huấn luyện tăng dần với dữ liệu mới như mạng neural có thể.

2. 2 Test Production

2.2.1 Sơ lược về Test Production

Để kiểm tra đầy đủ các mô hình của bạn trước khi phổ biến rộng rãi, bạn cần cả đánh giá ngoại tuyến (Offline evaluations) trước khi triển khai và thử nghiệm trong sản xuất. Chỉ đánh giá ngoại tuyến là không đủ.

Lý tưởng nhất là đưa ra một quy trình rõ ràng về cách đánh giá các mô hình: thử nghiệm nào sẽ chạy, ai thực hiện chúng và các ngưỡng áp dụng để thúc đẩy mô hình lên giai đoạn tiếp theo. Tốt nhất là các quy trình đánh giá này được tự động hóa và khởi động khi có bản cập nhật mô hình mới. Việc thăng cấp giai đoạn cần được xem xét tương tự như cách đánh giá CI/CD trong công nghệ phần mềm.

2.2.2 Pre-deployment offline evaluations

Đánh giá ngoại tuyến mô hình, chúng ta đã thảo luận có thể được sử dụng để lựa chọn mô hình. Các đánh giá tương tự có thể được sử dụng lại cho các đánh giá ngoại tuyến trước khi triển khai.

một số kỹ thuật đánh giá phổ biến nhất là

(1) Test splits:Sử dụng phần tách thử nghiệm để so sánh với đường cơ sở.

(2) Backtesting: Chạy thử nghiệm ngược.

Test splits thường là tĩnh để có một thước đo đáng tin cậy để so sánh nhiều mô hình. Điều này cũng có nghĩa rằng hiệu suất tốt trên một phân chia kiểm thử cũ tĩnh không đảm bảo hiệu suất tốt dưới điều kiện phân phối dữ liệu hiện tại trong môi trường sản xuất.

Backtesting (kiểm tra ngược) là ý tưởng sử dụng dữ liệu được gắn nhãn mới nhất mà mô hình chưa nhìn thấy trong quá trình huấn luyện để kiểm tra hiệu suất.

Có thể nói rằng Backtesting là đủ để tránh việc kiểm tra trong sản xuất, nhưng thực tế không phải vậy. Hiệu suất sản xuất cao hơn nhiều so với hiệu suất liên quan đến nhãn. Bạn vẫn cần quan sát những thứ như độ trễ, hành vi của người dùng với mô hình và độ chính xác tích hợp hệ thống để đảm bảo mô hình của bạn an toàn khi triển khai rộng rãi.

2.2.3 Testing in Production Strategies

2.2.3.1 Shadow Deployment

Triển khai đổ bóng (Shadow Deployment) là việc triển khai một mô hình mới song song với một mô hình hiện tại đang hoạt động. Mọi yêu cầu đến được gửi đến cả

hai mô hình nhưng chỉ phục vụ kết quả từ mô hình hiện tại. Các dự đoán từ mô hình thử nghiệm (hay mô hình đổ bóng) được ghi lại để so sánh.

Ưu điểm:

- An toàn: Đây là cách an toàn vì ngay cả khi mô hình mới gặp sự cố hoặc lỗi, dự đoán từ mô hình mới không được phục vụ cho người dùng.

- Đơn giản về mặt khái niệm: Ý tưởng rất đơn giản và dễ hiểu.

- Thu thập dữ liệu: Tất cả các mô hình đều nhận được toàn bộ lưu lượng, giúp thu thập dữ liệu đủ để có ý nghĩa thống kê trong các thử nghiệm.

Nhược điểm:

- Quan sát tương tác: Phương pháp này không hiệu quả khi đo lường tương tác của người dùng với dự đoán. Ví dụ, nếu đó là hệ thống đề xuất, các dự đoán từ mô hình đổ bóng không được phục vụ, gây trở ngại trong việc quan sát hành động của người dùng.

- Tốn kém: Nó làm tăng công việc dự đoán gấp đôi, đòi hỏi tài nguyên tính toán thêm.

- Xử lý các chế độ suy luận:

+ Chênh lệch hiệu suất: Quản lý các tình huống mô hình đổ bóng chậm hơn đáng kể so với mô hình chính trong việc phục vụ dự đoán.

+ Lỗi của mô hình đổ bóng: Quyết định liệu mô hình chính cũng nên gặp sự cố nếu mô hình đổ bóng gặp vấn đề.

Phương pháp này cung cấp một cách an toàn để giới thiệu mô hình mới, nhưng có thể không phù hợp trong các tình huống quan trọng việc quan sát tương tác của người dùng hoặc chi phí. Xử lý chênh lệch hiệu suất và sự cố giữa các mô hình tăng độ phức tạp và đòi hỏi quản lý cẩn thận.

2.2.3.2 A/B Testing

A/B Testing là việc triển khai mô hình thử nghiệm (challenger) song song với mô hình hiện tại (model A - champion) và định tuyến một phần trăm lưu lượng người

dùng tới mô hình thử nghiệm (model B). Dự đoán từ mô hình thử nghiệm được hiển thị cho người dùng. Sử dụng giám sát và phân tích dự đoán trên cả hai mô hình để xác định liệu hiệu suất của mô hình thử nghiệm có thống kê tốt hơn mô hình hiện tại hay không.

Phân chia lưu lượng phải là một thử nghiệm ngẫu nhiên thực sự. Nếu có bất kỳ thiên lệch lựa chọn nào (ví dụ: người dùng truy cập từ desktop nhận mô hình A và di động nhận mô hình B), kết luận của bạn sẽ không chính xác. Thử nghiệm phải chạy đủ lâu để thu thập đủ mẫu để đạt đủ độ tin cậy thống kê về sự khác biệt.

Ưu điểm:

- Thu thập phản hồi từ người dùng: Vì dự đoán được phục vụ cho người dùng, phương pháp này cho phép bạn thu thập đầy đủ cách người dùng phản ứng với các mô hình khác nhau.

- Dễ hiểu và chi phí thấp: A/B testing dễ hiểu và có rất nhiều thư viện và tài liệu hướng dẫn liên quan đến nó.

- Chi phí thấp: Bởi vì mỗi yêu cầu chỉ có một dự đoán, nên chi phí để thực hiện thử nghiệm này rất ít.

Nhược điểm:

- An toàn thấp hơn so với triển khai đổ bóng: Bạn cần một đảm bảo đánh giá ngoại tuyến mạnh mẽ hơn rằng mô hình của bạn sẽ không gặp sự cố lớn khiến việc xử lý lưu lượng thực tế trở nên không hiệu quả.

- Lựa chọn giữa rủi ro và số liệu: Bạn phải quyết định giữa việc chấp nhận rủi ro hơn (định tuyến nhiều lưu lượng hơn đến mô hình B) và việc thu thập đủ mẫu để phân tích nhanh hơn.

2.2.3.3 Canary Release

Canary Release là việc triển khai mô hình thử nghiệm (challenger) cùng với mô hình hiện tại (champion), nhưng bắt đầu với mô hình thử nghiệm không nhận lưu lượng. Dần dần chuyển lưu lượng từ mô hình hiện tại sang mô hình thử nghiệm (còn

được gọi là canary). Theo dõi các chỉ số hiệu suất của mô hình thử nghiệm, nếu nhìn chung tốt, tiếp tục chuyển toàn bộ lưu lượng đến mô hình thử nghiệm.

Cách thực hiện canary release có thể kết hợp với A/B testing để đo lường sự khác biệt hiệu suất một cách chặt chẽ.

Nếu mô hình thử nghiệm gặp vấn đề, có thể chuyển lưu lượng trở lại cho mô hình hiện tại.

Ưu điểm:

- Dễ hiểu.
- Đơn giản nhất để triển khai nếu bạn đã có cơ sở hạ tầng feature flagging trong công ty.
- Vì dự đoán từ mô hình thử nghiệm được phục vụ, bạn có thể sử dụng phương pháp này với các mô hình yêu cầu tương tác người dùng để thu thập hiệu suất.
- So với triển khai đổ bóng, chi phí thấp hơn. Một dự đoán cho mỗi yêu cầu.
- Nếu kết hợp với A/B testing, cho phép thay đổi động lượng lưu lượng mà mỗi mô hình nhận.

Nhược điểm:

- Có thể dễ dàng không cẩn thận trong việc xác định sự khác biệt hiệu suất.
- Nếu không được giám sát cẩn thận, có thể xảy ra sự cố. Đây có thể là lựa chọn ít an toàn nhưng rất dễ quay trở lại trạng thái trước.

2.2.3.4 Interleaving Experiments

Interleaving khác với A/B testing ở điểm là một người dùng nhận được dự đoán xen kẽ từ cả mô hình A và mô hình B thay vì chỉ nhận từ một trong hai mô hình. Chúng ta theo dõi hiệu suất của mỗi mô hình bằng cách đo lường sự ưa thích của người

dùng với dự đoán từ mỗi mô hình (ví dụ: người dùng nhấp nhiều hơn vào các gợi ý từ mô hình B).

Công việc đề xuất (recommendation tasks) là một trường hợp sử dụng điển hình cho việc Interleaving. Tuy nhiên, không phải mọi công việc đều phù hợp với chiến lược này.

Chúng ta muốn tránh việc một mô hình có lợi thế thiên vị người dùng như luôn chọn lựa chọn hàng đầu từ mô hình A. Nên xác suất chọn lựa chọn đầu tiên từ cả mô hình A và mô hình B phải tương đương. Các vị trí còn lại có thể được điền bằng phương pháp lựa chọn đội (xem hình minh họa).

Ưu điểm:

- Netflix đã thực nghiệm và phát hiện ra rằng Interleaving có thể xác định mô hình tốt nhất một cách đáng tin cậy với số lượng mẫu nhỏ hơn đáng kể so với A/B testing truyền thống.
- Điều này chủ yếu bởi cả hai mô hình đều nhận được toàn bộ lưu lượng.
- Trái ngược với triển khai đổ bóng, chiến lược này cho phép chúng ta thu thập được cách người dùng phản ứng với các dự đoán của chúng ta (vì dự đoán được phục vụ).

Nhược điểm:

- Triển khai phức tạp hơn so với A/B testing.
- Chúng ta cần quan tâm đến các trường hợp biên như làm gì nếu một trong các mô hình trong Interleaving mất quá lâu để phản hồi hoặc gặp sự cố.
- Tăng gấp đôi năng lực tính toán cần thiết vì mỗi yêu cầu nhận dự đoán từ nhiều mô hình.
- Không thể áp dụng cho mọi loại công việc. Ví dụ, nó phù hợp với các công việc xếp hạng/gợi ý nhưng không hợp lý cho các công việc hồi quy.
- Khó mở rộng với một số lượng lớn mô hình thử nghiệm. 2-3 mô hình Interleaving dường như là điểm tốt nhất.

2.2.3.5 Bandits

Bandits là một thuật toán theo dõi hiệu suất hiện tại của từng biến thể mô hình và đưa ra quyết định động trên mỗi yêu cầu về việc sử dụng mô hình hiện có hiệu suất tốt nhất cho đến nay (tức là tận dụng kiến thức hiện tại) hoặc thử nghiệm các mô hình khác để thu thập thêm thông tin về chúng (tức là khám phá trường hợp một trong các mô hình khác có thể tốt hơn).

Tuy nhiên, Bandits không áp dụng được cho tất cả các trường hợp. Để áp dụng Bandits, trường hợp sử dụng của bạn cần:

- Thực hiện dự đoán trực tuyến. Dự đoán tập trung ngoại tuyến không tương thích với Bandits.
- Cần các chu kỳ phản hồi ngắn để xác định xem dự đoán có tốt không và cơ chế để truyền phản hồi đó vào thuật toán Bandits để cập nhật lợi nhuận của từng mô hình.

Ưu điểm:

- Bandits cần ít dữ liệu hơn so với A/B testing để xác định mô hình nào tốt hơn. Một ví dụ đề cập trong sách kể về việc cần 630 nghìn mẫu để có 95% độ tin cậy với A/B testing so với chỉ cần 12 nghìn mẫu với Bandits.
- Bandits hiệu quả hơn về dữ liệu và đồng thời giảm thiểu chi phí cơ hội. Trong nhiều trường hợp, Bandits được xem xét là tối ưu.
- So với A/B testing, Bandits an toàn hơn vì nếu một mô hình thực sự tệ, thuật toán sẽ chọn nó ít hơn. Hơn nữa, việc hội tụ sẽ nhanh hơn, giúp bạn loại bỏ mô hình thử nghiệm tệ nhanh chóng.

Nhược điểm:

- So với các chiến lược khác, Bandits khó triển khai hơn rất nhiều do cần liên tục truyền phản hồi vào thuật toán.
- Bandits chỉ có thể sử dụng cho một số trường hợp sử dụng cụ thể.

- Chúng không an toàn như Triển khai đổ bóng vì các mô hình thử nghiệm nhận lưu lượng thực tế.

2. 3 Tác dụng của Continual Learning và Test Production trong xây dựng và duy trì một giải pháp học máy

2.3.1 Ảnh hưởng của Continual Learning

Duy trì kiến thức cũ khi học từ dữ liệu mới: Continual Learning giúp mô hình học từ dữ liệu mới mà vẫn duy trì được kiến thức cũ, tránh hiện tượng quên thông tin quan trọng đã học trước đó. Điều này rất cần thiết khi dữ liệu thay đổi theo thời gian và mô hình cần cập nhật để thích nghi với dữ liệu mới.

Giảm thiểu tác động tiêu cực của dữ liệu mới: Việc sử dụng các kỹ thuật Continual Learning giúp giảm thiểu tác động tiêu cực của dữ liệu mới lên hiệu suất của mô hình đối với dữ liệu cũ, như sự can thiệp hoặc quên mất thông tin.

2.3.2 Ảnh hưởng của Test Production

Đảm bảo đánh giá chính xác của hiệu suất mô hình: Bộ dữ liệu kiểm thử đáng tin cậy và đa dạng giúp xác định mức độ chính xác của mô hình trên các điều kiện và dữ liệu khác nhau. Điều này quan trọng để biết liệu mô hình có hoạt động tốt trên dữ liệu mới hay không.

Hỗ trợ quá trình cải thiện liên tục: Bằng việc thường xuyên kiểm tra hiệu suất trên các bộ kiểm thử mới, người ta có thể xác định được những điểm yếu của mô hình và cải thiện nó. Điều này làm tăng khả năng mô hình thích nghi với dữ liệu mới và giữ được hiệu suất cao hơn theo thời gian.

2.3.3 Ảnh hưởng chung

Khi kết hợp Continual Learning và Test Production, chúng ta tạo ra một quá trình linh hoạt và đầy đủ để xây dựng và duy trì một giải pháp học máy:

Đảm bảo sự linh hoạt và độ tin cậy của mô hình: Mô hình có thể học từ dữ liệu mới mà vẫn giữ được kiến thức cũ và được đánh giá trên các bộ kiểm thử đa dạng, đáng tin cậy.

Cải thiện liên tục và thích nghi với môi trường thay đổi: Các kỹ thuật Continual Learning kết hợp với Test Production giúp mô hình không chỉ duy trì hiệu suất mà còn cải thiện và thích nghi với dữ liệu mới và yêu cầu thay đổi.

Tóm lại, Continual Learning và Test Production không chỉ cần thiết để xây dựng một giải pháp học máy mà còn để duy trì và cải thiện mô hình trong điều kiện dữ liệu thay đổi liên tục.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. 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

Tiếng Anh

2. Sergio Rodríguez, 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
3. Ayush Gupta (2023), *A Comprehensive Guide on Optimizers in Deep Learning*, https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

PHỤ LỤC