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

ĐỒ ÁN CUỐI KÌ

MÔN NHẬP MÔN HỌC MÁY

TRÌNH BÀY NGHIÊN CỨU, ĐÁNH GIÁ PHẦN LÝ THUYẾT

Người hướng dẫn: **GV LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN TUẤN KIỆT - 52100243**

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 NHẬP MÔN HỌC MÁY

TRÌNH BÀY NGHIÊN CỨU, ĐÁNH GIÁ PHẦN LÝ THUYẾT

Người hướng dẫn: **GV LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN TUẤN KIỆT - 52100243**
Lớp **: 21050301**
Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Lời đầu tiên chúng em chân thành gửi lời cảm ơn đến **Trường đại học Tôn Đức Thắng** đã cho chúng em cơ hội được học và tiếp cận môn học này, giúp chúng em định hướng và có mục tiêu rõ ràng trong tương lai.

Chúng em xin chân thành gửi lời cảm ơn đến **thầy Lê Anh Cường** vì sự nhiệt tình của thầy trong việc giảng dạy, bên cạnh đó là những kiến thức và kinh nghiệm thầy truyền đạt giúp em hiểu biết hơn về các thuật toán và kiến thức nền tảng của môn học, tuy chỉ trong vài tháng ngắn ngủi được học tập và làm việc với thầy nhưng những kiến thức thầy truyền đạt là rất quý giá.

Trong quá trình làm bài báo cáo này, với kinh nghiệm và kiến thức còn ít ỏi nên không tránh được những sơ sót và hạn chế, em mong có thể nhận được sự góp ý của thầy để bài báo cáo được hoàn chỉnh hơn, hiệu quả hơn. Cuối cùng chúng em xin gửi lời chúc sức khoẻ đến thầy, chúc thầy luôn vui vẻ, hạnh phúc và thành công trong công cuộc lèo lái con đò đưa trò sang sông.

Tôi xin chân thành cảm ơn!

Trân trọng.

CAM KẾT
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 thầy 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 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 18 tháng 12 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Tuấn Kiệt

MỤC LỤC

CHƯƠNG 1 – TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 Tìm hiểu chung.

Optimizers, trong ngữ cảnh huấn luyện mô hình học máy, là những thuật toán quan trọng giúp tối ưu hóa các tham số (weights và bias) của mạng neural để mô hình có thể học và hiểu các đặc trưng (features) từ dữ liệu đầu vào. Điều quan trọng là làm thế nào để cập nhật các tham số này một cách hiệu quả để giảm thiểu giá trị của hàm mất mát và cải thiện khả năng dự đoán của mô hình.

Việc "học" các tham số như weights và bias trong mạng neural không đơn giản như việc chọn ngẫu nhiên các giá trị và hy vọng rằng mô hình sẽ học được thông tin từ dữ liệu. Thay vào đó, chúng ta cần một phương pháp cụ thể để điều chỉnh các tham số này theo từng bước, dựa trên thông tin về giá trị mất mát.

Thuật toán Optimizer ra đời để giúp ta thực hiện việc này. Chúng tạo ra một cách cụ thể để tính toán sự thay đổi cần thiết cho các tham số để giảm độ lỗi, dựa trên đạo hàm của hàm mất mát. Các biến thể của Optimizer, chẳng hạn như Gradient Descent và các phiên bản cải tiến khác, sử dụng gradient để xác định hướng và khoảng cách cần thay đổi cho từng tham số để mục tiêu tối ưu hóa có thể đạt được nhanh hơn.

1.2 Một số optimizer được sử dụng phổ biến.

1.2.1 Gradient Descent (GD)

Trong lĩnh vực Machine Learning và Toán Tối Ưu, việc tìm giá trị nhỏ nhất hoặc lớn nhất của một hàm số là một phần quan trọng của nhiều bài toán. Ví dụ, trong Linear Regression và K-means Clustering, chúng ta cần tối ưu hóa các hàm mất mát để đạt được mô hình tốt nhất.

Tuy nhiên, việc tìm kiếm global minimum (cực tiểu toàn cục) của các hàm mất mát trong Machine Learning thường rất phức tạp và đôi khi là không thể. Thay vào đó,

chúng ta thường tìm các điểm local minimum (cực tiểu cục bộ), và trong một số trường hợp, chúng ta coi những điểm này là nghiệm cần tìm của bài toán.

Các điểm local minimum là nghiệm của phương trình đạo hàm của hàm số bằng 0. Một cách tổng quát, ta có thể nghĩ đến việc tìm toàn bộ các điểm cực tiểu, và sau đó chọn ra điểm có giá trị nhỏ nhất cho hàm số. Tuy nhiên, trong thực tế, việc giải phương trình đạo hàm bằng 0 có thể rất khó hoặc thậm chí là không khả thi. Điều này có thể do đạo hàm của hàm số có thể rất phức tạp hoặc do dữ liệu có số chiều lớn, hoặc do số lượng điểm dữ liệu quá lớn.

Do đó, phương pháp tiếp cận phổ biến nhất là bắt đầu từ một điểm khởi đầu gần với nghiệm của bài toán và sử dụng phép toán lặp để tiến dần đến điểm cần tìm. Điều quan trọng là đến một thời điểm nào đó, đạo hàm của hàm số gần với 0. Phương pháp Gradient Descent (GD) và các biến thể của nó là một trong những cách tiếp cận phổ biến được sử dụng rộng rãi.

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta)$$

Trong đó:

- θ là vector của các tham số cần được tối ưu (ví dụ: trọng số trong mô hình học máy).
- $J(\theta)$ là hàm mất mát, đo lường sự sai lệch giữa kết quả dự đoán của mô hình và dữ liệu thực tế.
- $\nabla_\theta J(\theta)$ là gradient của hàm mất mát, tức là một vector chứa đạo hàm riêng của J theo từng tham số trong θ.
- α là tốc độ học (learning rate), quyết định bước nhảy trong không gian tham số.

- Ưu điểm:

Dễ hiểu và dễ triển khai.

Đảm bảo hội tụ tại điểm cực tiểu toàn cục nếu hàm mất mát là lồi.

- Nhược điểm:

Hội tụ chậm và có thể không hiệu quả với các bộ dữ liệu lớn do cần tính toán gradient trên toàn bộ dữ liệu.

1.2.2 Stochastic Gradient Descent (SGD).

Stochastic Gradient Descent (SGD) là một thuật toán tối ưu được sử dụng để giảm thiểu hàm mất mát trong huấn luyện mô hình học máy. SGD cập nhật các tham số của mô hình theo hướng ngược lại của gradient của hàm mất mát, nhưng chỉ dựa trên một mẫu ngẫu nhiên của tập dữ liệu.

SGD là một thuật toán đơn giản và hiệu quả, có thể được áp dụng cho các tập dữ liệu lớn. SGD cũng có thể được sử dụng để huấn luyện các mô hình có nhiều tham số.

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

Trong đó:

- θ là vector của các tham số cần được tối ưu (ví dụ: trọng số trong mô hình học máy).
- $J(\theta; x^{(i)}, y^{(i)})$ là hàm mất mát được tính dựa trên chỉ một cặp quan sát (hoặc một nhóm nhỏ các quan sát) $(x^{(i)}, y^{(i)})$, với $x^{(i)}$ là dữ liệu đầu vào và $y^{(i)}$ là nhãn tương ứng.
- $\nabla_\theta J(\theta; x^{(i)}, y^{(i)})$ là gradient của hàm mất mát, được tính dựa trên cặp quan sát đó.
- α là tốc độ học (learning rate).

Ưu điểm:

- Sử dụng phương pháp mẫu từ tập dữ liệu nên nhanh chóng hơn trong việc cập nhật các tham số.
- SGD khả dụng để giải quyết các bài toán từ điển, đồng thời nó cũng khả dụng trong việc giải quyết các bài toán đặc biệt.

Nhược điểm:

- Kết quả tìm kiếm tham số không ổn định, nhiều khi yêu cầu đi qua nhiều vòng lặp học cho đến khi tham số thay đổi rất ít.
- Nếu dùng cách tiềm năng ngẫu nhiên để lựa chọn mẫu từ tập dữ liệu, kết quả của SGD sẽ không ổn định và phụ thuộc vào quá trình tiềm năng ngẫu nhiên.

1.2.3 Momentum

Định nghĩa: Momentum là một biến thể của SGD, nó tích lũy gradient của các bước trước đó để đạt được hướng đi "quán tính" và giúp hội tụ nhanh hơn.

1. Cập nhật vector vận tốc (v):
   $v_t = \gamma v_{t-1} + \alpha \nabla_\theta J(\theta)$
2. Cập nhật tham số (θ):
   $\theta = \theta - v_t$

Trong đó:

- v_t là vector vận tốc tại bước thời gian t.
- γ là hệ số Momentum, thường lựa chọn trong khoảng từ 0.9 đến 0.99. Giá trị này định nghĩa mức độ "trọng lượng" của gradient trong quá khứ được bảo toàn.
- α là tốc độ học (learning rate).
- $\nabla_\theta J(\theta)$ là gradient của hàm mất mát tại tham số θ hiện tại.
- θ là vector của các tham số mô hình cần được tối ưu.

- Ưu điểm:

Hội tụ nhanh hơn SGD đơn thuần do "quán tính" giúp vượt qua các thung lũng. Giảm sự biến động của gradient, dẫn đến quá trình học ổn định hơn.

- Nhược điểm:

Có thể gặp khó khăn khi điều chỉnh các hyperparameter như tỷ lệ học và momentum.

1.2.4 Adagrad

Định nghĩa: Adagrad điều chỉnh tỷ lệ học của mỗi tham số dựa trên tần suất của chúng xuất hiện trong tập dữ liệu, thích hợp cho việc xử lý dữ liệu thưa.

1. Tích lũy bình phương gradient:
   $G_t = G_{t-1} + (\nabla_\theta J(\theta))^2$
2. Cập nhật tham số (θ):
   $\theta = \theta - \frac{\alpha}{\sqrt{G_t + \epsilon}} \cdot \nabla_\theta J(\theta)$

Trong đó:

- G_t là ma trận diagonal chứa tổng bình phương của các gradient cho đến bước thời gian t.
- $\nabla_\theta J(\theta)$ là gradient của hàm mất mát tại tham số θ hiện tại.
- α là tốc độ học ban đầu.
- ϵ là một hằng số nhỏ để tránh chia cho 0, thường được đặt là một giá trị nhỏ như $1e-8$.

- Ưu điểm:

Tự động điều chỉnh tỷ lệ học và thích hợp cho các bài toán có dữ liệu thưa thớt.

Loại bỏ phần nào nhu cầu phải điều chỉnh tỷ lệ học thủ công.

- Nhược điểm:

Tỷ lệ học có thể giảm quá nhanh và mô hình có thể ngừng học trước khi đạt được điểm tối ưu.

1.2.5 RMSPROP

Định nghĩa: RMSprop là một biến thể của Adagrad, giải quyết vấn đề giảm nhanh của tỷ lệ học bằng cách sử dụng một trung bình di động của gradient vuông để điều chỉnh tỷ lệ học.

1. Cập nhật trung bình di động trọng số cho bình phương gradient:
   $E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\nabla_\theta J(\theta))^2$
2. Cập nhật tham số (θ):
   $\theta = \theta - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot \nabla_\theta J(\theta)$

Trong đó:

- $E[g^2]_t$ là trung bình di động trọng số của bình phương gradient tại bước thời gian t.
- β là hệ số suy giảm, thường được chọn là một giá trị gần với 1, ví dụ 0.9.
- $\nabla_\theta J(\theta)$ là gradient của hàm mất mát tại tham số θ hiện tại.
- α là tốc độ học.
- ϵ là một hằng số nhỏ để tránh chia cho 0, thường là $1e-8$.

- Ưu điểm:

Tự động điều chỉnh tỷ lệ học và nói chung hoạt động tốt trong nhiều bài toán.

thiểu vấn đề giảm quá nhanh của tỷ lệ học so với Adagrad.

- Nhược điểm:

Vẫn có thể gặp khó khăn trong việc chọn các hyperparameter tốt nhất.

Có thể không hiệu quả như các phương pháp tối ưu hóa tiên tiến khác trong một số trường hợp.

1.3 So sánh ưu nhược điểm của các thuật toán

Tính năng	Gradient Descent (GD)	Stochastic Gradient Descent (SGD)	Momentum	Adagrad	RMSPROP
Cập nhật tham số	Sử dụng toàn bộ tập dữ liệu	Sử dụng một mẫu ngẫu nhiên	Kết hợp gradient hiện tại với gradient trước đó	Thích ứng tốc độ học tập theo	Thích ứng tốc độ học tập theo bình phương

				từng tham số	trung bình của gradient
Tốc độ	Chậm đối với tập dữ liệu lớn	Nhanh hơn GD	Nhanh hơn SGD, có thể giúp thoát khỏi điểm cực tiểu cục bộ	Nhanh hơn GD và SGD đối với dữ liệu thưa thớt	Nhanh hơn GD và SGD, ổn định hơn Adagrad
Nhạy cảm với bước nhảy	Có	Có	Ít hơn SGD	Ít	Ít
Khả năng thoát khỏi điểm cực tiểu cục bộ	Khó khăn hơn	Dễ dàng hơn	Dễ dàng hơn SGD	Tốt	Tốt
Phù hợp cho	Tập dữ liệu nhỏ	Tập dữ liệu lớn	Tập dữ liệu lớn, vấn đề điểm cực tiểu cục bộ	Dữ liệu thưa thớt	Tập dữ liệu lớn, vấn đề điểm cực tiểu cục bộ
Độ phức tạp	Thấp	Thấp	Trung bình	Cao	Trung bình

1.4 Kết luận

GD phù hợp cho các tập dữ liệu nhỏ và các bài toán đơn giản.

SGD thường được sử dụng cho các tập dữ liệu lớn và các bài toán phức tạp.

Momentum giúp tăng tốc độ hội tụ và có thể giúp thoát khỏi điểm cực tiểu cục bộ.

Adagrad phù hợp cho các bài toán với dữ liệu thưa thớt.

RMSPROP là một thuật toán mạnh mẽ và ổn định, thường được sử dụng trong thực tế.

CHƯƠNG 2 – TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY.

2.1 Tìm hiểu chung

2.1.1 Continual Learning

Continual Learning (CL) là một tiến trình khám phá để tìm hiểu và giải quyết các vấn đề liên quan đến việc học hỏi liên tục trong máy học. Nó tập trung vào việc cải thiện khả năng học tập của mô hình máy học để có thể chịu đựng các vòng lặp học mới, bất kể kích thước lớn hay nhỏ.

Một số lý do khiến CL quan trọng:

- Nhanh hơn quá trình đào tạo: Trong khi việc học mới về một lĩnh vực được coi là "học đầu tiên" trong vòng đời của một người, quá trình đào tạo kỹ năng cho các bước tiếp theo có thể được tăng cường thông qua CL.
- Cải thiện khả năng áp dụng: CL giúp mô hình máy học tự động điều chỉnh các tham số và mô hình để phù hợp với nhu cầu ứng dụng của người dùng, không chỉ học cách giải quyết vấn đề đầu tiên mà còn nhiều vấn đề liên quan khác.

- Tiết kiệm nguồn lực: CL giúp giảm thiểu sự phụ thuộc vào người học, từ đó tiết kiệm nguồn lực.

Một số kỹ thuật CL phổ biến:

- Incremental Learning: Mô hình được đào tạo trong vòng đời này sẽ nhớ những điều nó đã học trước đó, để cải thiện khả năng học tập của mình trong các bước tiếp theo.
- Lifelong Learning: Mô hình máy học này không bao giờ quên những gì nó đã học trước đó, nó sẽ tự động cập nhật thông tin mới để giảm thiểu sự chênh lệch giữa thực tế và những điều nó đã học.
- Transfer Learning: Mô hình máy học được đào tạo để giải quyết một loại vấn đề rồi được áp dụng để giải quyết một loại vấn đề khác.
- Continual Adaptation: Mô hình máy học này có khả năng điều chỉnh để đáp ứng các thay đổi trong môi trường, khi đó nó sẽ tự động học cách giải quyết các vấn đề mới.

2.1.2 Test Production

Test Production là một phương pháp kiểm tra phần mềm trong môi trường sản xuất. Phương pháp này cho phép các nhà phát triển kiểm tra phần mềm của họ trên dữ liệu và môi trường thực tế, giúp phát hiện các lỗi hoặc vấn đề tiềm ẩn mà có thể không được phát hiện trong môi trường kiểm thử.

Test Production thường được thực hiện bằng cách sử dụng một số phương pháp khác nhau, bao gồm:

- Feature flags: Feature flags là các công cụ cho phép các nhà phát triển bật hoặc tắt các tính năng phần mềm một cách chọn lọc. Feature flags có thể được sử dụng để triển khai các tính năng mới trong sản xuất một cách dần dần, giúp các nhà phát triển theo dõi hiệu suất của các tính năng đó và khắc phục các vấn đề tiềm ẩn trước khi triển khai rộng rãi.

- Canary releases: Canary releases là một phương pháp triển khai phần mềm trong đó một phần nhỏ người dùng nhận được bản cập nhật mới. Điều này cho phép các nhà phát triển theo dõi hiệu suất của bản cập nhật mới trên một số lượng người dùng nhỏ trước khi triển khai rộng rãi.
- A/B testing: A/B testing là một phương pháp so sánh hai phiên bản khác nhau của một tính năng hoặc sản phẩm. A/B testing có thể được sử dụng để đánh giá hiệu suất của các tính năng mới hoặc thay đổi đối với các tính năng hiện có.

Lợi ích của Test Production

- Chất lượng phần mềm được cải thiện: Test Production giúp phát hiện các lỗi hoặc vấn đề tiềm ẩn sớm hơn, giúp giảm thiểu tác động của các lỗi này đối với người dùng.
- Tốc độ triển khai được cải thiện: Test Production cho phép các nhà phát triển triển khai các tính năng mới nhanh hơn mà không cần phải lo lắng về chất lượng phần mềm.
- Trải nghiệm người dùng được cải thiện: Test Production giúp các nhà phát triển hiểu rõ hơn về cách người dùng tương tác với phần mềm của họ, giúp họ cải thiện trải nghiệm người dùng.

Nhược điểm của Test Production

- Rủi ro cao hơn: Test Production có thể làm tăng rủi ro phát hành phần mềm có lỗi hoặc vấn đề.
- Tốn kém hơn: Test Production có thể tốn kém hơn so với kiểm thử trong môi trường kiểm thử.
- Yêu cầu nhiều kỹ năng: Test Production yêu cầu các nhà phát triển có kỹ năng và kinh nghiệm để triển khai và quản lý các tính năng phần mềm trong môi trường sản xuất.

2.2 Xây dựng bài toán với Continual Learning và Test Production

2.2.1 Xây dựng bài toán với Continual Learning

Mô Tả

Giả sử bạn đang phát triển một hệ thống nhận dạng đối tượng dựa trên camera an ninh cho một tòa nhà văn phòng. Hệ thống này cần phát hiện và phân loại các loại đối tượng khác nhau như người, xe cộ, động vật, v.v. Tuy nhiên, môi trường xung quanh tòa nhà thay đổi theo mùa và theo năm, đồng thời xuất hiện các loại đối tượng mới theo thời gian (ví dụ: các mẫu xe mới).

Yêu Cầu

Học Từ Dữ Liệu Liên Tục: Hệ thống cần có khả năng học hỏi từ dữ liệu liên tục mà không cần đào tạo lại từ đầu mỗi khi có dữ liệu mới.

Tránh Quên Mất: Khi hệ thống học cách nhận dạng đối tượng mới, nó không nên quên cách nhận dạng các đối tượng đã học trước đó.

Thích Nghi Với Thay Đổi Môi Trường: Hệ thống cần thích nghi với sự thay đổi của môi trường (như ánh sáng, thời tiết) và sự xuất hiện của các đối tượng mới.

Cách Tiếp Cận

Thu Thập Dữ Liệu: Liên tục thu thập dữ liệu từ camera an ninh, bao gồm cả dữ liệu cũ và mới.

Xử Lý Dữ Liệu: Phân loại và gán nhãn cho dữ liệu mới, sử dụng cả dữ liệu đã có và dữ liệu mới thu thập.

Áp Dụng Continual Learning: Sử dụng các kỹ thuật như Elastic Weight Consolidation (EWC), Experience Replay, hoặc các phương pháp kiến trúc để tránh quên mất và thích nghi với dữ liệu mới.

Đánh Giá và Tinh Chỉnh: Thường xuyên đánh giá hiệu suất của mô hình trên cả dữ liệu cũ và mới, và tinh chỉnh mô hình để duy trì khả năng nhận dạng đối tượng một cách chính xác.

Kết Luận

Bài toán này đặt ra thách thức về việc xây dựng một mô hình học máy có khả năng thích nghi và học hỏi liên tục từ dữ liệu không ngừng thay đổi, đồng thời duy trì

kiến thức đã học trước đó. Đây là một ví dụ điển hình về ứng dụng của Continual Learning trong thực tế.

2.2.2 Xây dựng bài toán áp dụng Test Production

Mô Tả

Mục tiêu của bài toán này là phát triển một mô hình học máy có khả năng dự đoán nhu cầu sản phẩm cho một công ty bán lẻ. Mô hình này sẽ được đánh giá và tối ưu hóa trong môi trường thử nghiệm trước khi được triển khai trong môi trường sản xuất thực tế.

Yêu Cầu

Phát Triển Mô Hình: Xây dựng một mô hình học máy dựa trên dữ liệu lịch sử về doanh số bán hàng, xu hướng thị trường, và các yếu tố khác như mùa vụ và sự kiện đặc biệt.

Thử Nghiệm Mô Hình: Kiểm thử mô hình trong một môi trường giả lập hoặc môi trường thử nghiệm để đánh giá hiệu suất và khả năng dự đoán của nó.

Triển khai và Giám Sát trong Môi Trường Sản Xuất: Triển khai mô hình trong môi trường sản xuất thực tế và thiết lập hệ thống để giám sát hiệu suất của nó liên tục.

Tối Ưu Hóa Liên Tục: Cập nhật và tinh chỉnh mô hình dựa trên phản hồi từ môi trường sản xuất để cải thiện hiệu suất và độ chính xác.

Cách Tiếp Cận

Thu Thập và Xử Lý Dữ Liệu: Thu thập dữ liệu từ các nguồn khác nhau và tiền xử lý để chuẩn bị cho việc huấn luyện mô hình.

Huấn Luyện và Tối Ưu Mô Hình: Sử dụng các kỹ thuật học máy và học sâu để xây dựng mô hình, sau đó tối ưu hóa dựa trên hiệu suất trong môi trường thử nghiệm.

Kiểm Thử và Đánh Giá: Thực hiện các bài kiểm thử chéo và đánh giá mô hình dựa trên các chỉ số như sai số dự đoán, độ chính xác, và F1 score.

Triển Khai và Giám Sát: Triển khai mô hình vào môi trường sản xuất và thiết lập các hệ thống giám sát để theo dõi hiệu suất mô hình trong thời gian thực.

Phản Hồi và Tinh Chỉnh Liên Tục: Thu thập phản hồi từ môi trường sản xuất và sử dụng nó để cải thiện và tinh chỉnh mô hình.

Kết Luận

Bài toán này tập trung vào việc phát triển một mô hình học máy có khả năng dự đoán nhu cầu sản phẩm một cách chính xác, và đồng thời xác định cách thức để mô hình này có thể được triển khai và hoạt động hiệu quả trong môi trường sản xuất thực tế. Đây là một ví dụ về cách mà học máy có thể được áp dụng để giải quyết các vấn đề kinh doanh thực tế và cung cấp giá trị thực sự cho các doanh nghiệp.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. https://machinelearningcoban.com/2017/01/16/gradientdescent2/
2. https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8
3. https://jos.husc.edu.vn/backup/upload/vol_18/no_1/668_fulltext_4.%C4%90TVT%20-%20Phuoc%20-%20Vuong%20Quang%20Phuoc.pdf

Tiếng Anh

1. https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0
2. https://www.geeksforgeeks.org/

PHỤ LỤC