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

NGÔ VĂN DANH - 52100877

PHẦN LÀM BÀI CÁ NHÂN

BÁO CÁO CUỐI KỲ 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

NGÔ VĂN DANH - 52100877

PHẦN LÀM BÀI CÁ NHÂN

BÁO CÁO CUỐI KỲ MÔN

HỌC MÁY

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn THẦY LÊ ANH CƯỜNG. Trong suốt quá trình học tập và rèn luyện, chúng em đã nhận được rất nhiều sự giúp đỡ tận tình, sự quan tâm, chăm sóc của THẦY LÊ ANH CƯỜNG. Ngoài ra, chúng em còn được thầy truyền đạt những kiến thức, phương pháp mới về toán hay ho và thú vị, thầy còn giúp sinh viên có được nhiều niềm vui trong việc học và cảm thấy thoải mái, … Chúng em xin chân thành cảm ơn thầy rất nhiều trong suốt quá trình học tập này!

Bởi lượng kiến thức của chúng em còn hạn hẹp và gặp nhiều vấn đề trong quá trình học nên báo cáo này sẽ còn nhiều thiếu sót và cần được học hỏi thêm. Chúng em rất mong sẽ nhận được sự góp ý của thầy về bài báo cáo này để em rút kinh nghiệm trong những môn học sắp tới. Cuối cùng, chúng em xin chân thành cảm ơn thầy .

TP. Hồ Chí Minh, ngày 11 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Ngô Văn Danh

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi và được sự hướng dẫn 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 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 chúng 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 11 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Ngô Văn Danh

TÓM TẮT

Trong môn học máy, việc lựa chọn phương pháp optimizer là một phần quan trọng trong quá trình huấn luyện mô hình. Có nhiều phương pháp optimizer khác nhau, như Gradient Descent, Stochastic Gradient Descent (SGD), Adam, RMSprop, và gồm nhiều phương pháp khác. Mỗi phương pháp này có ưu điểm và nhược điểm riêng, và lựa chọn phụ thuộc vào loại dữ liệu, cấu trúc mô hình, và yêu cầu của bài toán cụ thể.

Continual Learning đề cập đến khả năng của một mô hình học máy để liên tục học từ dữ liệu mới mà không quên kiến thức đã học trước đó. Điều này quan trọng trong các ứng dụng nơi mô hình phải đối mặt với dữ liệu thay đổi theo thời gian. Trong khi đó, Test Production là quá trình kiểm thử và triển khai mô hình vào môi trường sản xuất, nơi mô hình sẽ xử lý dữ liệu thực tế và cung cấp dự đoán.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

ĐỀ: PHẦN LÀM CÁ NHÂN

Câu 1: Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

1) Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy;
2) Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

BÀI LÀM:

Câu 1.1

Phương pháp Optimizer trong mô hình học máy dùng để tối ưu hóa hàm mất mát của mô hình. Hàm mất mát là một hàm đo lường mức độ sai lệch giữa kết quả dự đoán của mô hình và kết quả thực tế. Phương pháp Optimizer sẽ tìm cách thay đổi các tham số của mô hình để giảm thiểu hàm mất mát.

Có nhiều phương pháp Optimizer khác nhau, mỗi phương pháp có những ưu điểm và nhược điểm riêng, Lựa chọn phương pháp Optimizer phù hợp phụ thuộc vào nhiều yếu tố, bao gồm loại mô hình, loại dữ liệu và mục tiêu của bài toán học máy.

Phương pháp Optimizer đóng một vai trò quan trọng trong việc xác định hiệu suất của mô hình học máy. Một phương pháp Optimizer tốt sẽ giúp mô hình nhanh chóng tìm ra các tham số tối ưu, từ đó cải thiện độ chính xác của mô hình.

-Một số ví dụ về cách sử dụng phương pháp Optimizer trong mô hình học máy:

+Trong các mô hình phân loại, phương pháp Optimizer được sử dụng để tìm các tham số của mô hình sao cho kết quả dự đoán của mô hình càng gần với kết quả thực tế càng tốt.

+Trong các mô hình hồi quy, phương pháp Optimizer được sử dụng để tìm các tham số của mô hình sao cho tổng sai số giữa kết quả dự đoán của mô hình và kết quả thực tế càng nhỏ càng tốt.

+Trong các mô hình tạo mẫu, phương pháp Optimizer được sử dụng để tìm các tham số của mô hình sao cho mô hình tạo ra các mẫu càng giống với dữ liệu thực tế càng tốt.

***Sau đây là chi tiết một số phương pháp Optimizer phổ biến:**

A. Gradient Descent:

-Phương pháp tối ưu Gradient Descent (GD) là một trong những phương pháp cơ bản nhất và phổ biến nhất trong huấn luyện mô hình học máy. Nó được sử dụng để tìm kiếm điểm tối ưu của một hàm mất mát thông qua việc điều chỉnh các tham số của mô hình.

-Ý tưởng chính của GD là dựa trên việc cập nhật từng tham số theo hướng ngược với đạo hàm của hàm mất mát. Bằng cách tính toán gradient của hàm mất mát theo từng tham số, ta có thể điều chỉnh các tham số để tiến gần hơn đến điểm tối ưu của hàm mất mát.

-Công thức cập nhật tham số trong GD được biểu diễn như sau:

$$\theta\_new = \theta\_old - learning\_rate * \nabla J(\theta\_old),$$

Trong đó:

+θ_old là giá trị hiện tại của tham số.

+θ_new là giá trị được cập nhật của tham số.

+learning_rate (hay còn gọi là tỷ lệ học) là hằng số dương xác định tốc độ học của mô hình, quyết định khoảng cách mà GD di chuyển trên đồ thị hàm mất mát.

+∇J(θ_old) là gradient của hàm mất mát J theo từng tham số θ_old.

-Tốc độ học là một tham số quan trọng của GD, quyết định tốc độ hội tụ của thuật toán. Nếu tốc độ học quá lớn, thuật toán có thể bị vượt biên (overshoot) và không hội tụ. Nếu tốc độ học quá nhỏ, thuật toán sẽ mất nhiều thời gian để hội tụ. Trong thực tế, người ta thường sử dụng phương pháp giảm tốc độ học dần dần theo thời gian để đảm bảo thuật toán hội tụ.

- Ưu điểm của GD:

+Đơn giản và dễ hiểu

+Có thể áp dụng cho nhiều loại hàm số

+Có thể được sử dụng để tối ưu hóa các mô hình học máy phức tạp

-Nhược điểm của GD:

+Có thể bị vượt biên nếu tốc độ học quá lớn

+Có thể mất nhiều thời gian để hội tụ nếu hàm số có nhiều cục bộ tối ưu

=>Gradient descent là một thuật toán tối ưu quan trọng trong học máy. GD có thể được sử dụng để tối ưu hóa các mô hình học máy phức tạp, nhưng cần chú ý điều chỉnh tốc độ học để đảm bảo thuật toán hội tụ.

B. Stochastic Gradient Descent

-Là một biến thể của gradient descent, trong đó gradient được tính toán dựa trên một điểm dữ liệu ngẫu nhiên trong tập dữ liệu huấn luyện. Ý tưởng của SGD là thay vì tính gradient dựa trên toàn bộ tập dữ liệu huấn luyện, SGD chỉ tính gradient dựa trên một điểm dữ liệu ngẫu nhiên trong tập dữ liệu. Điều này giúp giảm chi phí tính toán của gradient descent, đặc biệt là khi tập dữ liệu huấn luyện lớn.

-Công thức cập nhật tham số trong SGD được biểu diễn như sau:

$\theta\_new = \theta\_old - learning\_rate * \nabla J(\theta\_old, x\_i, y\_i)$,

-Trong đó:

+θ_old là giá trị hiện tại của tham số.

+θ_new là giá trị được cập nhật của tham số.

+learning_rate (tỷ lệ học) là hằng số dương xác định tốc độ học của mô hình, quyết định khoảng cách mà SGD di chuyển trên đồ thị hàm mất mát.

+$\nabla J(\theta\_old, x\_i, y\_i)$ là gradient của hàm mất mát $J(\theta, x\_i, y\_i)$ theo từng tham số θ_old, tính dựa trên điểm dữ liệu đang được sử dụng (x_i, y_i).

-Tương tự thuật toán SG tốc độ học là một tham số quan trọng của SGD, quyết định tốc độ hội tụ của thuật toán. Nếu tốc độ học quá lớn, thuật toán có thể bị vượt biên và không hội tụ. Nếu tốc độ học quá nhỏ, thuật toán sẽ mất nhiều thời gian để hội tụ. Trong thực tế, người ta thường sử dụng phương pháp giảm tốc độ học dần dần theo thời gian để đảm bảo thuật toán hội tụ.

-Ưu điểm của SGD:

+Chi phí tính toán thấp, đặc biệt là khi tập dữ liệu huấn luyện lớn

+Có thể được sử dụng để tối ưu hóa các mô hình học máy phức tạp

-Nhược điểm của SGD:

+Có thể bị vượt biên nếu tốc độ học quá lớn

+Có thể bị mắc kẹt trong cục bộ tối ưu

-SGD được ứng dụng rộng rãi trong học máy, đặc biệt là trong các mô hình học sâu, chẳng hạn như mạng nơ-ron nhân tạo. SGD cũng được sử dụng trong các bài toán tối ưu khác, chẳng hạn như trong các bài toán phân loại, hồi quy, và clustering.

=>SGD là một thuật toán tối ưu hiệu quả, đặc biệt là khi tập dữ liệu huấn luyện lớn. Tuy nhiên, cần chú ý điều chỉnh tốc độ học để đảm bảo thuật toán hội tụ và tránh bị mắc kẹt trong cục bộ tối ưu.

C. Adagrad:

-Là một phương pháp tối ưu adaptive được sử dụng trong huấn luyện mô hình học máy. Nó được thiết kế để cải thiện việc điều chỉnh tỷ lệ học (learning rate) theo từng tham số trong quá trình huấn luyện. Adagrad điều chỉnh tỷ lệ học theo cách sao cho các tham số có đạo hàm lớn được cập nhật với tỷ lệ nhỏ hơn, và các tham số có đạo hàm nhỏ được cập nhật với tỷ lệ lớn hơn.

-Ý tưởng chính của Adagrad là sử dụng thông tin từ lịch sử của các gradient đã tính để điều chỉnh tỷ lệ học. Cụ thể, Adagrad tích lũy bình phương của các gradient đã tính từ các lần cập nhật trước đó.

-Công thức cập nhật tham số trong Adagrad được biểu diễn như sau:

$$\theta\_new = \theta\_old - (learning\_rate / (sqrt(G + epsilon))) * \nabla J(\theta\_old),$$

-Trong đó:

+θ_old là giá trị hiện tại của tham số.

+θ_new là giá trị được cập nhật của tham số.

+learning_rate (tỷ lệ học) là hằng số dương xác định tốc độ học của mô hình.

+∇J(θ_old) là gradient của hàm mất mát J theo từng tham số θ_old.

+G là ma trận đường chéo được tính bằng cách tích lũy bình phương của các gradient đã tính từ các lần cập nhật trước đó. Mỗi phần tử G[i, i] tương ứng với tham số θ[i].

+epsilon là một hằng số nhỏ được thêm vào mẫu số để tránh chia cho 0.

-Ưu điểm:

+Tự động điều chỉnh tốc độ học theo từng tham số của mô hình

+Có thể hoạt động tốt với các bài toán có các tham số có độ lệch khác nhau

-Nhược điểm:

+Có thể bị mắc kẹt trong cục bộ tối ưu

+Có thể mất nhiều thời gian để hội tụ nếu hàm số có nhiều cục bộ tối ưu

-Adagrad được ứng dụng rộng rãi trong học máy, đặc biệt là trong các mô hình học sâu, chẳng hạn như mạng nơ-ron nhân tạo. AdaGrad cũng được sử dụng trong các bài toán tối ưu khác, chẳng hạn như trong các bài toán phân loại, hồi quy, và clustering.

=>AdaGrad là một thuật toán tối ưu hiệu quả, đặc biệt là với các bài toán có các tham số có độ lệch khác nhau. Tuy nhiên, cần chú ý điều chỉnh tốc độ học để tránh bị mắc kẹt trong cục bộ tối ưu.

D. **Adam:**

-Phương pháp tối ưu Adam (Adaptive Moment Estimation) là một phương pháp tối ưu adaptive được sử dụng trong huấn luyện mô hình học máy. Nó kết hợp cả phương pháp Momentum và RMSprop để tận dụng lợi thế của cả hai và cải thiện quá trình tối ưu hóa.

- Adam sử dụng một ước lượng độ đo độ dốc thứ nhất (first moment) và độ đo độ dốc thứ hai (second moment) của gradient để điều chỉnh tỷ lệ học cho từng tham số. Công thức cập nhật tham số trong Adam được biểu diễn như sau:

*m_i = β1 * m_i-1 + (1 - β1) * g_i, (cập nhật first moment)*

*v_i = β2 * v_i-1 + (1 - β2) * g_i^2, (cập nhật second moment)*

m^_i = m_i / (1 - β1^t), (sửa lỗi bias first moment)

v^_i = v_i / (1 - β2^t), (sửa lỗi bias second moment)

*θ_new = θ_old - (learning_rate / (sqrt(v^_i) + epsilon)) * m^_i, (cập nhật tham số)*

-Trong đó:

+θ_old là giá trị hiện tại của tham số.

+θ_new là giá trị được cập nhật của tham số.

+learning_rate (tỷ lệ học) là hằng số dương xác định tốc độ học của mô hình.

+g_i là gradient của tham số θ[i] trong lần cập nhật hiện tại.

+m_i là first moment của gradient, là ước lượng trung bình trượt của gradient.

+v_i là second moment của gradient, là ước lượng trung bình trượt của bình phương gradient.

+β1 và β2 (beta1 và beta2) là các hệ số giảm trọng số, thường có giá trị từ 0.9 đến 0.999. Tương tự như RMSprop, chúng xác định mức độ đóng góp của các lần cập nhật trước đó trong trung bình trượt. Giá trị gần 1 sẽ làm cho trung bình trượt dài hơn và trung bình trượt nhanh hơn.

+t là bước lặp hiện tại.

+m^_i và v^_i là sửa lỗi bias first moment và second moment. Do việc khởi tạo ban đầu của m_i và v_i là 0, chúng có xu hướng bị bias về 0 ở các bước đầu tiên. Việc sửa lỗi bias này giúp cải thiện độ tin cậy của ước lượng.

-Adam kết hợp cả first moment và second moment của gradient để điều chỉnh tỷ lệ học. First moment (m_i) tương tự như trong phương pháp Momentum, giúp tích lũy gradient của các bước trước đó để tăng tốc quá trình tối ưu. Second moment (v_i) tương tự như trong phương pháp RMSprop, giúp điều chỉnh tỷ lệ học dựa trên trung bình trượt của bình phương gradient để thích ứng với các tham số có gradient lớn hoặc nhỏ.

-Adam được ứng dụng rộng rãi trong học máy, đặc biệt là trong các mô hình học sâu, chẳng hạn như mạng nơ-ron nhân tạo. Adam cũng được sử dụng trong các bài toán tối ưu khác, chẳng hạn như trong các bài toán phân loại, hồi quy, và clustering.

<u>-Ưu điểm:</u>

+Tự động điều chỉnh tốc độ học theo từng tham số của mô hình

+Có thể hoạt động tốt với các bài toán có các tham số có độ lệch khác nhau

+Có thể hội tụ nhanh hơn RMSprop nếu hàm số có nhiều cục bộ tối ưu

+Ít bị mắc kẹt trong cục bộ tối ưu hơn AdaGrad

<u>-Nhược điểm:</u>

+Có thể yêu cầu điều chỉnh các tham số β1 và β2

=>Adam là một thuật toán tối ưu hiệu quả, đặc biệt là với các bài toán có các tham số có độ lệch khác nhau. Adam cũng có thể hội tụ nhanh hơn RMSprop nếu hàm số có nhiều cục bộ tối ưu. Adam cũng ít bị mắc kẹt trong cục bộ tối ưu hơn AdaGrad. Tuy nhiên, cần chú ý điều chỉnh các tham số β1 và β2 để đảm bảo thuật toán hội tụ hiệu quả.

E. RMSprop (Root Mean Square Propagation)

-Phương pháp tối ưu RMSprop (Root Mean Square Propagation) là một phương pháp tối ưu adaptive được sử dụng trong huấn luyện mô hình học máy. Nó cải tiến từ phương pháp Adagrad bằng cách giới hạn việc tích lũy bình phương gradient trong quá khứ, nhằm giảm vấn đề giảm tỷ lệ học quá nhanh. RMSprop sử dụng một trung bình trượt của bình phương gradient để điều chỉnh tỷ lệ học theo từng tham số.

-Ý tưởng chính của RMSprop là sử dụng một trung bình trượt của bình phương gradient để điều chỉnh tỷ lệ học. Công thức cập nhật tham số trong RMSprop được biểu diễn như sau:

g_i = ∇J(θ_old), (cập nhật gradient)

*E[g^2] = α * E[g^2] + (1 - α) * g_i^2, (cập nhật trung bình trượt)*

*θ_new = θ_old - (learning_rate / (sqrt(E[g^2]) + epsilon)) * g_i, (cập nhật tham số)*

-Trong đó:

+θ_old là giá trị hiện tại của tham số.

+θ_new là giá trị được cập nhật của tham số.

+learning_rate (tỷ lệ học) là hằng số dương xác định tốc độ học của mô hình.

+∇J(θ_old) là gradient của hàm mất mát J theo từng tham số θ_old.

+g_i là gradient của tham số θ[i] trong lần cập nhật hiện tại.

+E[g^2] là trung bình trượt của bình phương gradient. Ban đầu, nó được khởi tạo là 0 hoặc một giá trị khác nhau tùy thuộc vào triển khai cụ thể.

+α (alpha) là hệ số giảm trọng số, thường có giá trị từ 0.9 đến 0.999. Nó xác định mức độ đóng góp của các lần cập nhật trước đó trong trung bình trượt. Giá trị gần 1 sẽ làm cho trung bình trượt dài hơn và trung bình trượt nhanh hơn.

+epsilon là một hằng số nhỏ được thêm vào mẫu số để tránh chia cho 0 và giúp ổn định tính toán.

-RMSprop giúp điều chỉnh tỷ lệ học theo từng tham số dựa trên trung bình trượt của bình phương gradient. Điều này giúp tối ưu hóa tốt hơn trong việc điều chỉnh tỷ lệ học và thích ứng với các tham số có gradient lớn hoặc nhỏ. Ngoài ra, RMSprop cũng giảm vấn đề giảm tỷ lệ học quá nhanh của phương pháp Adagrad bằng cách giới hạn việc tích lũy bình phương gradient trong quá khứ.

-Ưu điểm:

+Tự động điều chỉnh tốc độ học theo từng tham số của mô hình

+Có thể hoạt động tốt với các bài toán có các tham số có độ lệch khác nhau

+Có thể hội tụ nhanh hơn AdaGrad nếu hàm số có nhiều cục bộ tối ưu

-Nhược điểm:

+Có thể bị mắc kẹt trong cục bộ tối ưu

-RMSprop được ứng dụng rộng rãi trong học máy, đặc biệt là trong các mô hình học sâu, chẳng hạn như mạng nơ-ron nhân tạo. RMSprop cũng được sử dụng trong các bài toán tối ưu khác, chẳng hạn như trong các bài toán phân loại, hồi quy, và clustering.

=>RMSprop là một thuật toán tối ưu hiệu quả, đặc biệt là với các bài toán có các tham số có độ lệch khác nhau. RMSprop cũng có thể hội tụ nhanh hơn AdaGrad nếu hàm số có nhiều cục bộ tối ưu. Tuy nhiên, cần chú ý điều chỉnh tốc độ học để tránh bị mắc kẹt trong cục bộ tối ưu.

F. ***Bảng so sánh các phương pháp***

Phương pháp	Mô tả	Ưu điểm	Nhược điểm
Gradient descent (GD)	Cập nhật các tham số của mô hình theo hướng ngược lại của độ dốc của hàm mất mát.	Đơn giản, dễ hiểu.	Có thể gặp phải vấn đề hội tụ trong các vùng có độ dốc nhỏ.
Stochastic gradient descent (SGD)	Cập nhật các tham số của mô hình theo hướng ngược lại của độ dốc của hàm mất mát dựa trên một mẫu ngẫu nhiên từ dữ liệu huấn luyện.	Nhanh hơn GD.	Có thể gặp phải vấn đề hội tụ trong các vùng có độ dốc nhỏ.
Adagrad	Điều chỉnh tỷ lệ học tập theo độ dốc của hàm mất mát.	Có thể giúp mô hình học nhanh hơn trong các vùng có độ dốc thay đổi nhanh.	Có thể dẫn đến tỷ lệ học tập quá nhỏ trong các vùng có độ dốc thay đổi chậm.
RMSprop	Điều chỉnh tỷ lệ học tập theo bình phương của độ dốc của hàm mất mát.	Ổn định hơn Adagrad trong các vùng có độ dốc thay đổi nhanh.	Có thể dẫn đến tỷ lệ học tập quá nhỏ trong các vùng có độ dốc thay đổi chậm.
Adam	Kết hợp các ưu điểm của Adagrad và RMSprop.	Có thể giúp mô hình học nhanh hơn và ổn định hơn trong các vùng có độ dốc thay đổi nhanh hoặc chậm.	Có thể phức tạp hơn các phương pháp khác.

Bảng 1 So sánh các phương pháp Optimizer

Câu 1.2

A. ***Continual Learning (Lifelong Learning)***

-Continual Learning (hay còn gọi là Lifelong Learning hoặc Incremental Learning) là một lĩnh vực quan trọng trong học máy, tập trung vào khả năng của một hệ thống học máy để liên tục học và tích luỹ kiến thức từ các tác vụ/đầu vào mới mà không quên đi kiến thức đã học được từ các tác vụ/đầu vào trước đó. Mục tiêu là xây dựng các mô hình học máy có khả năng học tập liên tục và đảm bảo sự tiến bộ thông qua việc tích hợp các kiến thức cũ và mới.

- Có một số trở ngại chính trong Continual Learning:

+Quên đi kiến thức cũ: Khi học thêm các tác vụ mới, mô hình có thể quên đi kiến thức đã học từ các tác vụ/trạng thái trước đó. Điều này gây ảnh hưởng đến khả năng tổng hợp kiến thức và làm giảm hiệu suất của mô hình.

+Nhiễu kiến thức: Sự chồng chéo giữa các tác vụ hoặc thông tin mâu thuẫn có thể gây ra nhiễu kiến thức, khiến cho mô hình khó khăn trong việc tách biệt thông tin quan trọng và không quan trọng.

+Độ phức tạp tính toán: Khi số lượng tác vụ và dữ liệu tăng lên, việc lưu trữ và tính toán trở nên phức tạp hơn. Mô hình cần có khả năng xử lý dữ liệu lớn và hiệu quả tính toán để đáp ứng yêu cầu của học tập liên tục.

-Có nhiều phương pháp và kỹ thuật khác nhau đã được đề xuất để giải quyết các trở ngại trong Continual Learning. Dưới đây là một số phương pháp phổ biến:

+Regularization-based methods: Các phương pháp này sử dụng các biện pháp như Elastic Weight Consolidation (EWC) hoặc Synaptic Intelligence (SI) để giữ nguyên kiến thức quan trọng từ các tác vụ trước đó trong quá trình học tập mới. Các phương pháp này đặt mức độ ưu tiên cho các trọng số liên quan đến kiến thức cũ và giảm thiểu mất mát kiến thức đã học được.

+Replay-based methods: Các phương pháp này sử dụng các mô-đun sử dụng dữ liệu đã học trước đó để phục tái huấn luyện mô hình trên các tác vụ mới. Dữ liệu đã học trước đó được sử dụng để tạo ra các mẫu giả lập hoặc lưu trữ để tăng cường việc học tập mới.

+Dynamic architectures: Các phương pháp này tập trung vào cải thiện khả năng mở rộng của mô hình bằng cách thay đổi kiến trúc của nó. Một số phương pháp đáng chú ý bao gồm Progressive Neural Networks (PNN) và đa mạng nơ-ron (Multi-Task Learning).

+Memory-based methods: Các phương pháp này sử dụng bộ nhớ để lưu trữ thông tin quan trọng từ các tác vụ trước đó. Mô hình có thể truy xuất các mẫu đã học trước đó từ bộ nhớ để hỗ trợ việc học tập vào các tác vụ mới.

+Generative replay methods: Các phương pháp này sử dụng mô hình sinh dữ liệu để tạo ra các mẫu dữ liệu mới, giúp mô hình học tập trên các tác vụ/trạng thái cũ và mới.

-CL có nhiều ứng dụng tiềm năng trong thực tế, chẳng hạn như:

+Tự động lái xe: Mô hình CL có thể được sử dụng để học các tình huống lái xe mới, chẳng hạn như tình huống giao thông mới hoặc các điều kiện thời tiết thay đổi.

Ví dụ, một mô hình CL có thể được sử dụng để học cách lái xe an toàn trong điều kiện tuyết rơi. Mô hình này có thể được đào tạo trên dữ liệu từ các tình huống lái xe trong điều kiện bình thường, và sau đó được cập nhật với dữ liệu từ các tình huống lái xe trong điều kiện tuyết rơi. Mô hình CL này có thể giúp xe tự lái tránh được tai nạn trong điều kiện tuyết rơi.

+Chăm sóc sức khỏe: Mô hình CL có thể được sử dụng để học các triệu chứng mới của bệnh tật hoặc các phương pháp điều trị mới.

Ví dụ, một mô hình CL có thể được sử dụng để học cách phát hiện các tế bào ung thư mới. Mô hình này có thể được đào tạo trên dữ liệu từ các tế bào ung thư đã biết, và sau đó được cập nhật với dữ liệu từ các tế bào ung thư mới. Mô hình CL này có thể giúp các bác sĩ chẩn đoán ung thư sớm hơn và chính xác hơn.

+Chế biến ngôn ngữ tự nhiên: Mô hình CL có thể được sử dụng để học các ngôn ngữ mới hoặc các loại dữ liệu văn bản mới.

Ví dụ, một mô hình CL có thể được sử dụng để dịch từ tiếng Anh sang tiếng Trung. Mô hình này có thể được đào tạo trên dữ liệu từ các bản dịch tiếng Anh-Trung đã biết, và sau đó được cập nhật với dữ liệu từ các bản dịch tiếng Anh-Trung mới. Mô hình CL này có thể giúp các doanh nghiệp giao tiếp với khách hàng ở nước ngoài hiệu quả hơn.

-Khi sử dụng CL, cần lưu ý một số điểm sau:

+CL có thể phức tạp và tốn kém hơn so với các phương pháp học máy truyền thống.

+Khả năng hoạt động của CL phụ thuộc vào kỹ thuật CL được sử dụng và đặc điểm của bài toán.

+CL có thể dẫn đến hiện tượng overfitting nếu không được thực hiện cẩn thận.

B. Test Production:

-Test Production là một quá trình kiểm tra và đánh giá giải pháp học máy trong môi trường sản xuất thực tế. Quá trình này nhằm mục đích đảm bảo rằng giải pháp học máy hoạt động chính xác và hiệu quả trong môi trường thực tế.

-Giải pháp học máy thường được đào tạo trên tập dữ liệu được thu thập từ môi trường thực tế. Tuy nhiên, tập dữ liệu này không thể đại diện hoàn toàn cho tất cả các điều kiện mà giải pháp học máy sẽ phải đối mặt trong môi trường sản xuất. Do đó, cần phải thực hiện Test Production để kiểm tra giải pháp học máy trong môi trường sản xuất thực tế.

-Test Production giúp đảm bảo rằng giải pháp học máy hoạt động chính xác và hiệu quả trong môi trường thực tế. Điều này có ý nghĩa quan trọng đối với doanh nghiệp, vì nó giúp doanh nghiệp tránh được các tổn thất tài chính hoặc thiệt hại uy tín do giải pháp học máy hoạt động không hiệu quả.

- Quá trình Test Production thường bao gồm lần lượt các bước sau:

+Chuẩn bị dữ liệu: Dữ liệu được sử dụng để kiểm tra giải pháp học máy phải là dữ liệu thực tế từ môi trường sản xuất. Dữ liệu này phải được thu thập và chuẩn bị cẩn thận để đảm bảo rằng nó đại diện cho các điều kiện thực tế mà giải pháp học máy sẽ phải đối mặt.

+Thiết lập môi trường kiểm tra: Môi trường kiểm tra phải mô phỏng môi trường sản xuất thực tế. Điều này bao gồm việc sử dụng cùng loại phần cứng, phần mềm và dữ liệu như trong môi trường sản xuất.

+Thực hiện kiểm tra: Giải pháp học máy được kiểm tra bằng cách chạy nó trên dữ liệu kiểm tra. Kết quả của quá trình kiểm tra được thu thập và phân tích để xác định xem giải pháp học máy có hoạt động chính xác và hiệu quả hay không.

+Thay đổi và cải tiến giải pháp: Nếu giải pháp học máy không hoạt động chính xác hoặc hiệu quả, cần thực hiện các thay đổi và cải tiến. Các thay đổi này có thể bao gồm việc điều chỉnh các tham số của mô hình học máy, thu thập thêm dữ liệu đào tạo, hoặc thay đổi môi trường kiểm tra.

-Có nhiều phương pháp kiểm tra khác nhau có thể được sử dụng trong Test Production. Một số phương pháp phổ biến bao gồm:

+Kiểm tra độ chính xác: Phương pháp này xác định xem giải pháp học máy có dự đoán chính xác kết quả hay không.

+Kiểm tra hiệu suất: Phương pháp này xác định xem giải pháp học máy có đáp ứng được các yêu cầu về hiệu suất hay không.

+Kiểm tra độ bền: Phương pháp này xác định xem giải pháp học máy có hoạt động chính xác và hiệu quả trong một thời gian dài hay không.

-Khi thực hiện Test Production, cần lưu ý một số điểm sau:

+Quá trình Test Production nên được thực hiện thường xuyên: Điều này sẽ giúp đảm bảo rằng giải pháp học máy vẫn hoạt động chính xác và hiệu quả trong môi trường thực tế.

+Quá trình Test Production nên được thực hiện bởi các kỹ sư có kinh nghiệm: Các kỹ sư có kinh nghiệm sẽ có thể xác định các vấn đề tiềm ẩn với giải pháp học máy và thực hiện các thay đổi cần thiết.

+Quá trình Test Production nên được kết hợp với các quá trình kiểm soát chất lượng khác: Điều này sẽ giúp đảm bảo rằng giải pháp học máy đáp ứng các yêu cầu chất lượng của doanh nghiệp.

=>Test Production là một quá trình quan trọng trong việc đảm bảo chất lượng của giải pháp học máy. Quá trình này giúp đảm bảo rằng giải pháp học máy hoạt động chính xác và hiệu quả trong môi trường thực tế.

C. Đánh giá chung Continual Learning và Test Production

Continual Learning và Test Production có thể được kết hợp với nhau để tạo ra một giải pháp học máy hiệu quả và bền vững. Trong đó, Continual Learning sẽ giúp mô hình học hỏi và thích nghi với dữ liệu mới trong khi Test Production sẽ giúp cải thiện hiệu suất của mô hình trong thời gian dài.

Ví dụ, trong lĩnh vực cảnh báo sớm, một mô hình Continual Learning có thể được sử dụng để phát hiện các dấu hiệu sớm của các sự kiện bất thường. Mô hình này sẽ được cập nhật thường xuyên với dữ liệu mới từ các nguồn khác nhau, chẳng hạn như dữ liệu thời tiết, dữ liệu giao thông và dữ liệu mạng xã hội. Test Production có thể được sử dụng để triển khai mô hình này trong môi trường sản xuất đồng thời với việc tiếp tục thu thập và huấn luyện mô hình trên dữ liệu mới. Điều này sẽ giúp đảm bảo rằng mô hình có thể phát hiện các dấu hiệu sớm của các sự kiện bất thường một cách chính xác và kịp thời

TÀI LIỆU THAM KHẢO

1.https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/

2.https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

3.https://optimization.cbe.cornell.edu/index.php?title=AdaGrad

4.https://deepchecks.com/how-to-test-machine-learning-models/

5.https://cnvrg.io/how-to-use-continual-learning-to-your-machine-learning-models/