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

HUỲNH HOÀNG TIẾN - 51704111

SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

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

HUỲNH HOÀNG TIẾN - 51704111

SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

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 tôi muốn bày tỏ lòng biết ơn sâu sắc đến **PGS.TS. Lê Anh Cường**, Khoa Công Nghệ Thông Tin, và Trường Đại Học Tôn Đức Thắng vì sự hỗ trợ và đóng góp quan trọng trong quá trình thực hiện báo cáo này. Thầy Cường không chỉ là nguồn động viên lớn, mà còn là người hướng dẫn chi tiết, giúp chúng tôi nắm vững kiến thức của môn "Nhập môn Học máy".

Khoa Công Nghệ Thông Tin mang lại môi trường học tập năng động và sáng tạo, nơi chúng tôi có cơ hội tiếp xúc với những kiến thức và công nghệ mới nhất. Đặc biệt, chúng tôi muốn bày tỏ lòng biết ơn với Trường Đại Học Tôn Đức Thắng vì môi trường học thuận lợi, giúp chúng tôi không ngừng khám phá và phát triển.

Cuối cùng, chúng tôi chân thành cảm ơn tất cả những người đã đóng góp vào hành trình học tập và nghiên cứu của chúng tôi. Sự giúp đỡ và khuyến khích của mọi người đã là động lực quan trọng, giúp chúng tôi vượt qua những thách thức và đạt được những thành công nhỏ trên con đường này. Xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày ... tháng ... năm 20..
Tác giả
(Ký tên và ghi rõ họ tên)

Huỳnh Hoàng Tiến

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 ... tháng ... năm 20..

Tác giả

(Ký tên và ghi rõ họ tên)

Huỳnh Hoàng Tiến

TÓM TẮT

1. *Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy:*

Trong nghiên cứu này, em đã tìm hiểu và so sánh các phương pháp Optimizer phổ biến trong huấn luyện mô hình học máy. Các phương pháp mà em đã nghiên cứu bao gồm Gradient Descent, Stochastic Gradient Descent (SGD), Momentum, RMSProp và Adam.

Em đã tìm hiểu cách mỗi phương pháp hoạt động, cách tính toán gradient và cách cập nhật trọng số trong quá trình huấn luyện. Em đã nghiên cứu cả ưu điểm và nhược điểm của từng phương pháp để hiểu rõ hơn về hiệu suất và khả năng hội tụ của chúng.

Sau đó, em tiến hành thực hiện các thí nghiệm để so sánh hiệu suất của các phương pháp Optimizer này trên các tập dữ liệu khác nhau. Em đã đánh giá hiệu suất dựa trên các tiêu chí như tốc độ hội tụ, độ chính xác và khả năng tránh trạng thái hố đen (local minima).

Kết quả của nghiên cứu cho thấy mỗi phương pháp Optimizer có những đặc điểm riêng, phù hợp với các bài toán và tập dữ liệu khác nhau. Việc lựa chọn phương pháp Optimizer phù hợp có thể giúp tăng tốc quá trình huấn luyện và đạt được kết quả tốt hơn.

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:*

Trong phần này của nghiên cứu, em đã tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán cụ thể.

Em đã tìm hiểu về khái niệm Continual Learning (học liên tục), đây là một phương pháp học máy mà mô hình được huấn luyện liên tục trên các tập dữ liệu mới mà không quên đi kiến thức cũ. Em đã nghiên cứu các phương pháp và kỹ thuật Continual Learning như Elastic Weight Consolidation (EWC), Incremental Moment Matching (IMM) và Experience Replay để hiểu cách chúng hoạt động và ứng dụng trong việc giải quyết các bài toán phức tạp.

Em cũng đã tìm hiểu về Test Production, đây là quá trình đánh giá hiệu suất và độ chính xác của mô hình học máy đã được xây dựng. Em đã nghiên cứu các phương pháp Test Production như Cross-Validation, Holdout Validation và Bootstrap để hiểu cách thực hiện kiểm tra hiệu suất mô hình và đánh giá tính khả thi của nó.

Kết quả của nghiên cứu cho thấy Continual Learning và Test Production là hai khía cạnh quan trọng trong xây dựng giải pháp học máy. Hiểu rõ về các phương pháp và kỹ thuật trong cả hai lĩnh vực này có thể giúp nâng cao hiệu suất và độ chính xác của mô hình học máy và đảm bảo tính liên tục và khả năng kiểm tra của nó trong quá trình triển khai và sử dụng.

ABSTRACT

1. *Understanding and comparing Optimizer methods in machine learning model training:*

In this research, I have studied and compared popular Optimizer methods used in training machine learning models. The methods that I have researched include Gradient Descent, Stochastic Gradient Descent (SGD), Momentum, RMSProp, and Adam.

I have learned how each method works, how gradients are computed, and how weight updates are performed during training. I have also studied the advantages and disadvantages of each method to understand their performance and convergence capabilities.

Next, I conducted experiments to compare the performance of these Optimizer methods on different datasets. I evaluated the performance based on criteria such as convergence speed, accuracy, and the ability to avoid local minima.

The results of the research showed that each Optimizer method has its own characteristics and is suitable for different problems and datasets. Choosing the appropriate Optimizer method can help accelerate the training process and achieve better results.

2. *Understanding Continual Learning and Test Production when building a machine learning solution for a specific problem:*

In this part of the research, I have studied Continual Learning and Test Production when building a machine learning solution for a specific problem.

I have learned about the concept of Continual Learning, which is a machine learning approach where the model is continuously trained on new datasets without

forgetting previous knowledge. I have studied Continual Learning methods and techniques such as Elastic Weight Consolidation (EWC), Incremental Moment Matching (IMM), and Experience Replay to understand how they work and their applications in solving complex problems.

I have also studied Test Production, which is the process of evaluating the performance and accuracy of the built machine learning model. I have researched Test Production methods such as Cross-Validation, Holdout Validation, and Bootstrap to understand how to perform model evaluation and assess its feasibility.

The results of the research show that Continual Learning and Test Production are important aspects in building a machine learning solution. Having a clear understanding of methods and techniques in both areas can improve the performance and accuracy of the machine learning model and ensure its continuity and testability in deployment and usage.

MỤC LỤC

DANH MỤC CÁC CHỮ VIẾT TẮT

GD	Gradient Descent
SGD	Stochastic Gradient Descent
RMSProp	Root Mean Square Propagation
Adam	Adaptive Moment Estimation

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

1.1 Mô tả cụ thể về cách mỗi phương pháp hoạt động:

1.1.1 Gradient Descent (GD)

Gradient Descent (GD) là một phương pháp tối ưu hóa được sử dụng trong quá trình huấn luyện các mô hình máy học. Phương pháp này được sử dụng để tìm giá trị nhỏ nhất (hoặc lớn nhất) của một hàm số thông qua việc điều chỉnh các tham số của mô hình.

Cách Gradient Descent hoạt động như sau:

- Khởi tạo ngẫu nhiên các giá trị ban đầu cho các tham số của mô hình. Điểm khởi tạo này được gọi là điểm xuất phát.
- Tính đạo hàm riêng của hàm mục tiêu (thường là hàm mất mát) theo từng tham số của mô hình. Đạo hàm này cho biết hướng tăng nhanh nhất của hàm mục tiêu tại một điểm cụ thể.
- Dùng đạo hàm đã tính được để xác định hướng di chuyển. Với mục tiêu tìm giá trị nhỏ nhất, ta di chuyển theo hướng âm của đạo hàm. Với mục tiêu tìm giá trị lớn nhất, ta di chuyển theo hướng dương của đạo hàm.
- Đặt một bước di chuyển (learning rate) để xác định khoảng cách di chuyển theo hướng đó. Bước di chuyển này quyết định tốc độ và độ lớn của sự điều chỉnh của các tham số.

- Cập nhật các tham số của mô hình bằng cách di chuyển từ điểm hiện tại đến điểm mới theo hướng và khoảng cách đã xác định. Việc này được thực hiện bằng cách trừ đi bước di chuyển nhân với đạo hàm.
  Công thức cập nhật: $\theta_{i+1} =: \theta_i - \alpha \nabla J(\theta_i)$, với θ là vector trọng số, α là learning rate.

- Lặp lại các bước 2-5 cho đến khi một điều kiện dừng được thỏa mãn, ví dụ như đạt được một số lượng lặp tối đa hoặc hàm mục tiêu không thay đổi đáng kể.

Qua mỗi lần lặp, GD tiếp tục cập nhật các tham số của mô hình để di chuyển đến điểm có giá trị hàm mục tiêu thấp hơn. Điều này giúp mô hình tiếp cận với giá trị tối ưu hóa và tìm ra các tham số tốt nhất cho mô hình máy học. Tuy nhiên, việc sử dụng GD cần lưu ý đến bước di chuyển và các vấn đề liên quan để tránh vấn đề như hội tụ chậm hoặc rơi vào điểm cực tiểu cục bộ.

1.1.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của Gradient Descent (GD) được sử dụng rộng rãi trong quá trình huấn luyện mô hình máy học. Khác với GD, SGD không tính toán đạo hàm trên toàn bộ tập dữ liệu huấn luyện mỗi lần cập nhật tham số. Thay vào đó, nó chỉ tính đạo hàm trên một mẫu dữ liệu ngẫu nhiên được chọn trong mỗi lần cập nhật.

Cách Stochastic Gradient Descent hoạt động như sau:

- Khởi tạo ngẫu nhiên các giá trị ban đầu cho các tham số của mô hình. Điểm khởi tạo này được gọi là điểm xuất phát.
- Lặp lại các bước sau cho đến khi một điều kiện dừng được thỏa mãn:
    - Chọn ngẫu nhiên một mẫu dữ liệu từ tập dữ liệu huấn luyện.
    - Tính đạo hàm riêng của hàm mục tiêu (thường là hàm mất mát) theo từng tham số của mô hình, sử dụng mẫu dữ liệu được chọn.
    - Dùng đạo hàm đã tính được để cập nhật các tham số của mô hình theo hướng và khoảng cách đã xác định. Việc này được thực hiện bằng cách trừ đi bước di chuyển nhân với đạo hàm.

- Lặp lại các bước 2 cho đến khi một điều kiện dừng được thỏa mãn, ví dụ như đạt được một số lượng lặp tối đa hoặc hàm mục tiêu không thay đổi đáng kể.

SGD giúp giảm thiểu thời gian tính toán vì nó chỉ tính toán đạo hàm trên một mẫu dữ liệu thay vì toàn bộ tập dữ liệu. Điều này đặc biệt hữu ích khi làm việc với tập dữ liệu lớn. Tuy nhiên, SGD có thể làm cho quá trình hội tụ chậm hơn và có thể tạo ra sự dao động trong quá trình cập nhật tham số do tính ngẫu nhiên trong việc chọn mẫu dữ liệu. Để giải quyết vấn đề này, có thể sử dụng các biến thể khác của SGD như Mini-batch Gradient Descent để tính đạo hàm trên một số lượng nhỏ các mẫu dữ liệu.

1.1.3 Momentum

Momentum là một phương pháp tối ưu hóa được sử dụng trong quá trình huấn luyện mô hình máy học. Nó giúp tăng tốc quá trình hội tụ và giảm hiện tượng dao động khi cập nhật các tham số của mô hình.

Cách Momentum hoạt động như sau:

- Khởi tạo một vector độ lớn bằng 0, gọi là biến momentum, có kích thước tương tự với các tham số của mô hình.
- Lặp lại các bước sau cho đến khi một điều kiện dừng được thỏa mãn:
    - Tính toán đạo hàm riêng của hàm mục tiêu (thường là hàm mất mát) theo từng tham số của mô hình, sử dụng mẫu dữ liệu được chọn.
    - Cập nhật giá trị của biến momentum theo hướng và khoảng cách đã xác định. Việc này được thực hiện bằng cách cộng tổng của một hệ số quán tính (thường được ký hiệu là "γ") nhân với giá trị của biến momentum trước đó và tổng của một hệ số học tập (thường được ký hiệu là "η") nhân với đạo hàm.

    - Cập nhật các tham số của mô hình bằng cách trừ đi giá trị của biến momentum đã được cập nhật.
- Lặp lại các bước 2 cho đến khi một điều kiện dừng được thỏa mãn, ví dụ như đạt được một số lượng lặp tối đa hoặc hàm mục tiêu không thay đổi đáng kể.

Momentum giúp giảm thiểu hiện tượng dao động trong quá trình cập nhật tham số bằng cách tích lũy thông tin từ các bước cập nhật trước đó. Nếu gradient trên một hướng nhất định giữ nguyên hướng và giá trị lớn, thì biến momentum sẽ tăng dần theo hướng đó. Điều này giúp tăng tốc độ hội tụ và tránh bị mắc kẹt ở điểm tối ưu cục bộ. Đồng thời, nếu gradient thay đổi hướng hoặc giá trị nhỏ, biến momentum sẽ giảm dần để tránh quá đi xa khỏi điểm tối ưu dự kiến.

Momentum là một phương pháp quan trọng trong tối ưu hóa mô hình máy học và được sử dụng rộng rãi trong các thuật toán như Gradient Descent với Momentum (GD+Momentum) và Adaptive Moment Estimation (Adam).

1.1.4 RMSProp (Root Mean Square Propagation)

RMSProp (Root Mean Square Propagation) là một thuật toán tối ưu hóa đường dẫn truy ngược (backpropagation) được sử dụng trong quá trình huấn luyện các mạng nơ-ron. Nó giúp điều chỉnh tỉ lệ học tập cho từng tham số một cách tự động và hiệu quả.

Cách RMSProp hoạt động như sau:

- Khởi tạo một vector độ lớn bằng 0, gọi là biến sử dụng được lưu trữ, có kích thước tương tự với các tham số của mô hình.
- Lặp lại các bước sau cho từng lượt huấn luyện:

- Tính toán đạo hàm riêng của hàm mục tiêu (thường là hàm mất mát) theo từng tham số của mô hình, sử dụng mẫu dữ liệu được chọn.
- Cập nhật giá trị của biến sử dụng được lưu trữ bằng cách tính toán trung bình bình phương của đạo hàm theo từng tham số. Cụ thể, ta tính toán bình phương của đạo hàm, sau đó lấy trung bình trọng số với giá trị đã lưu trữ trước đó, sử dụng một hệ số giảm dần (thường được ký hiệu là "ρ").
- Cập nhật các tham số của mô hình bằng cách trừ đi giá trị của hệ số học tập nhân với đạo hàm chia cho căn bậc hai của giá trị được tính toán từ biến sử dụng được lưu trữ.

- Lặp lại các bước 2 cho đến khi một điều kiện dừng được thỏa mãn, ví dụ như đạt được một số lượng lặp tối đa hoặc hàm mục tiêu không thay đổi đáng kể.

RMSProp giúp điều chỉnh tỉ lệ học tập tự động dựa trên độ lớn của đạo hàm. Nếu đạo hàm lớn đối với một tham số nhất định đã được lưu trữ trong biến sử dụng được lưu trữ, tỉ lệ học tập sẽ được giảm để tránh sự dao động không cần thiết. Ngược lại, nếu đạo hàm nhỏ hoặc gần bằng 0, tỉ lệ học tập sẽ được tăng lên để tăng tốc quá trình hội tụ.

RMSProp là một phương pháp tối ưu hóa quan trọng trong huấn luyện mạng nơ-ron và thường được sử dụng cùng với các thuật toán khác như Gradient Descent với Momentum (GD+Momentum) và Adaptive Moment Estimation (Adam).

1.1.5 Adam (Adaptive Moment Estimation)

Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa đường dẫn truy ngược (backpropagation) được sử dụng trong quá trình huấn luyện các mạng nơ-ron. Nó kết hợp cả mômentum và RMSProp để tối ưu hóa hiệu quả hơn.

Cách Adam hoạt động như sau:

- Khởi tạo một vector độ lớn bằng 0, gọi là biến mômentum, có kích thước tương tự với các tham số của mô hình.
- Khởi tạo một vector độ lớn bằng 0, gọi là biến RMSProp, có kích thước tương tự với các tham số của mô hình.
- Khởi tạo một biến đếm bằng 0.
- Lặp lại các bước sau cho từng lượt huấn luyện:
    - Tính toán đạo hàm riêng của hàm mục tiêu (thường là hàm mất mát) theo từng tham số của mô hình, sử dụng mẫu dữ liệu được chọn.
    - Tăng giá trị của biến đếm lên một.
    - Cập nhật giá trị của biến mômentum bằng cách tính toán trọng số của đạo hàm theo từng tham số. Cụ thể, ta tính toán trọng số của đạo hàm bằng cách lấy trung bình trọng số giữa đạo hàm và giá trị của biến mômentum đã lưu trữ trước đó, sử dụng một hệ số giảm dần (thường được ký hiệu là "β_1").
    - Cập nhật giá trị của biến RMSProp bằng cách tính toán trọng số của đạo hàm bình phương theo từng tham số. Cụ thể, ta tính toán trọng số của đạo hàm bình phương bằng cách lấy trung bình trọng số giữa đạo hàm bình phương và giá trị của biến RMSProp đã lưu trữ trước đó, sử dụng một hệ số giảm dần (thường được ký hiệu là "β_2").
    - Điều chỉnh giá trị của biến mômentum và biến RMSProp bằng cách chia cho 1 trừ đi lũy thừa β_1 hoặc β_2 tương ứng.
    - Cập nhật các tham số của mô hình bằng cách trừ đi giá trị của hệ số học tập nhân với biến mômentum chia cho căn bậc hai của biến RMSProp đã được điều chỉnh, cộng thêm một giá trị rất nhỏ để tránh chia cho 0.

- Lặp lại các bước 4 cho đến khi một điều kiện dừng được thỏa mãn, ví dụ như đạt được một số lượng lặp tối đa hoặc hàm mục tiêu không thay đổi đáng kể.

Adam kết hợp cả mômentum và RMSProp để cung cấp một phương pháp tối ưu hóa linh hoạt và hiệu quả. Mômentum giúp tăng tốc quá trình hội tụ bằng cách lưu trữ lịch sử của các gradient trước đó, trong khi RMSProp giúp điều chỉnh tỉ lệ học tập tự động dựa trên độ lớn của đạo hàm. Adam đã được sử dụng rộng rãi trong huấn luyện mạng nơ-ron và được coi là một trong những thuật toán tối ưu hóa hiệu quả nhất.

1.2 Ưu điểm và nhược điểm chi tiết

1.2.1 Gradient Descent (GD)

1.2.1.1 Ưu điểm

Sự đơn giản: GD là một phương pháp đơn giản và dễ hiểu. Nó chỉ yêu cầu tính toán đạo hàm của hàm mục tiêu và cập nhật các tham số dựa trên đạo hàm đó.

Khả năng áp dụng rộng rãi: GD có thể được áp dụng cho nhiều loại mô hình học máy và hàm mục tiêu khác nhau.

Hiệu quả với dữ liệu nhỏ: Khi dữ liệu huấn luyện có kích thước nhỏ, GD có thể tìm ra một cực tiểu toàn cục tương đối nhanh chóng.

1.2.1.2 Nhược điểm

Tốc độ hội tụ chậm: GD có thể hội tụ chậm do việc cập nhật tham số dựa trên toàn bộ tập dữ liệu huấn luyện. Điều này đặc biệt đáng chú ý khi tập dữ liệu rất lớn.

Mắc kẹt ở các điểm cực tiểu cục bộ: GD có thể bị mắc kẹt ở các điểm cực tiểu cục bộ, khiến nó không thể tìm ra cực tiểu toàn cục của hàm mục tiêu.

Nhạy cảm với khởi tạo ban đầu: Kết quả của GD có thể thay đổi mạnh mẽ dựa trên khởi tạo ban đầu của các tham số. Nếu khởi tạo không tốt, GD có thể không thể hội tụ hoặc hội tụ đến cực tiểu cục bộ.

Để khắc phục nhược điểm của GD, đã phát triển nhiều biến thể của phương pháp này như Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent, và các phương pháp Optimizer khác như Momentum, RMSProp, Adam, v.v.

1.2.2 Stochastic Gradient Descent (SGD)

1.2.2.1 Ưu điểm

Tốc độ hội tụ nhanh: SGD có thể hội tụ nhanh hơn so với Gradient Descent (GD) truyền thống vì nó chỉ cập nhật tham số dựa trên một mẫu dữ liệu hoặc một lượng nhỏ mẫu dữ liệu (mini-batch) thay vì toàn bộ tập dữ liệu huấn luyện.

Tiêu thụ bộ nhớ ít: Vì SGD chỉ sử dụng một mẫu dữ liệu hoặc mini-batch để cập nhật tham số, nên nó tiêu thụ ít bộ nhớ hơn so với GD truyền thống.

Dễ dàng áp dụng cho dữ liệu lớn: Với tập dữ liệu lớn, SGD có thể được áp dụng hiệu quả hơn GD truyền thống vì nó không cần tính toán đạo hàm trên toàn bộ dữ liệu huấn luyện.

1.2.2.2 Nhược điểm

Không ổn định: SGD có thể không ổn định và dao động xung quanh điểm cực tiểu tối ưu. Điều này là do việc cập nhật tham số dựa trên một mẫu dữ liệu hoặc mini-batch ngẫu nhiên có thể dẫn đến sự biến động trong quá trình tối ưu.

Đòi hỏi sự điều chỉnh thích hợp: Vì SGD là phương pháp ngẫu nhiên, nó đòi hỏi sự điều chỉnh thích hợp của các siêu tham số như tỷ lệ học (learning rate) để đảm bảo hội tụ và tránh dao động quá mức.

Khó khăn với dữ liệu không đồng nhất: Nếu dữ liệu không đồng nhất về phân phối hoặc quan trọng đối với việc tối ưu, SGD có thể gặp khó khăn trong việc tìm ra điểm cực tiểu tối ưu.

Tuy nhiên, SGD vẫn là một phương pháp phổ biến và mạnh mẽ trong tối ưu hóa mô hình học máy, đặc biệt là khi làm việc với dữ liệu lớn và mô hình phức tạp.

1.2.3 Momentum

1.2.3.1 Ưu điểm:

Tăng tốc độ hội tụ: Momentum giúp tăng tốc độ hội tụ bằng cách tích lũy thông tin về các bước trước đó và sử dụng nó để cập nhật tham số. Điều này giúp vượt qua các vùng địa phương hẹp và tiếp tục tiến gần đến điểm cực tiểu tối ưu.

Giảm độ dao động: Momentum giúp giảm độ dao động của quá trình tối ưu hóa bằng cách giảm ảnh hưởng của các gradient nhỏ và tăng ảnh hưởng của các gradient lớn. Điều này giúp tối ưu hóa ổn định hơn và đạt được kết quả tốt hơn trong quá trình huấn luyện.

Tránh rơi vào các điểm cực tiểu cục bộ: Momentum giúp tránh rơi vào các điểm cực tiểu cục bộ bằng cách giúp mô hình "vượt qua" các vùng không tối ưu và tiếp tục tìm kiếm điểm cực tiểu tối ưu toàn cục.

1.2.3.2 Nhược điểm

Đòi hỏi cân nhắc về tỷ lệ học: Việc sử dụng Momentum đòi hỏi cân nhắc về tỷ lệ học (learning rate). Nếu tỷ lệ học được đặt quá cao, quá trình tối ưu hóa có thể bị dao động mạnh và không hội tụ. Ngược lại, nếu tỷ lệ học được đặt quá thấp, quá trình hội tụ có thể chậm và mất nhiều thời gian hơn để đạt được kết quả tối ưu.

Tuy nhiên, tổng thể, Momentum là một phương pháp mạnh mẽ và phổ biến trong tối ưu hóa mô hình học máy. Nó cung cấp lợi ích vượt trội trong việc tăng tốc độ hội tụ, giảm độ dao động và tránh rơi vào các điểm cực tiểu cục bộ.

1.2.4 RMSProp (Root Mean Square Propagation)

1.2.4.1 Ưu điểm

Tính ổn định: RMSProp giúp làm giảm độ dao động trong quá trình tối ưu hóa bằng cách điều chỉnh tỷ lệ học (learning rate) dựa trên các gradient lịch sử. Điều này giúp tối ưu hóa ổn định hơn và đạt được kết quả tốt hơn trong quá trình huấn luyện.

Tính hiệu quả trong các bài toán thưa: RMSProp hiệu quả trong việc tối ưu hóa các bài toán có ma trận gradient thưa (sparse gradient), nơi chỉ một số lượng nhỏ các phần tử trong gradient khác 0. Phương pháp này giúp tập trung vào các phần tử quan trọng và giảm ảnh hưởng của các phần tử không quan trọng.

Khả năng tùy chỉnh: RMSProp cho phép tùy chỉnh các tham số quan trọng như tỷ lệ học (learning rate) và hệ số giảm tỷ lệ học (decay rate). Điều này giúp điều chỉnh phương pháp tối ưu hóa cho phù hợp với từng bài toán cụ thể.

1.2.4.2 Nhược điểm

Đòi hỏi cân nhắc về tham số: Việc sử dụng RMSProp đòi hỏi cân nhắc về việc lựa chọn các tham số như tỷ lệ học và hệ số giảm tỷ lệ học. Nếu các tham số này được đặt không đúng cách, quá trình tối ưu hóa có thể không hội tụ hoặc hội tụ quá chậm.

Có thể bị mắc kẹt ở điểm cực tiểu cục bộ: Mặc dù RMSProp giúp giảm độ dao động và tìm được điểm cực tiểu tương đối tốt, nó có thể bị mắc kẹt ở các điểm cực tiểu cục bộ và không đạt được điểm cực tiểu toàn cục. Điều này có thể xảy ra nếu tỷ lệ học được đặt quá cao hoặc quá thấp.

Tuy nhiên, tổng thể, RMSProp là một phương pháp hiệu quả và phổ biến trong tối ưu hóa mô hình học máy. Nó cung cấp lợi ích về tính ổn định, hiệu quả trong các bài toán thưa và khả năng tùy chỉnh.

1.2.5 Adam (Adaptive Moment Estimation)

1.2.5.1 Ưu điểm

Tính ổn định và hiệu quả: Adam kết hợp các ưu điểm của phương pháp RMSProp và phương pháp Momentum. Nó giúp tối ưu hóa ổn định và nhanh chóng trong quá trình huấn luyện mô hình học máy.

Tự động điều chỉnh tỷ lệ học: Adam tự động điều chỉnh tỷ lệ học (learning rate) cho từng tham số dựa trên các thông tin về gradient và lịch sử gradient. Điều này giúp tối ưu hóa tỷ lệ học cho phù hợp với từng tham số và đạt được kết quả tốt hơn.

Hiệu quả trong các bài toán có độ lệch gradient lớn: Adam có khả năng hiệu quả trong việc tối ưu hóa các mô hình có độ lệch gradient lớn. Điều này giúp giảm thời gian huấn luyện và đạt được kết quả tốt hơn trong các bài toán phức tạp.

1.2.5.2 Nhược điểm

Đòi hỏi cân nhắc về tham số: Tương tự như RMSProp, Adam cũng đòi hỏi cân nhắc về việc lựa chọn các tham số như tỷ lệ học và hệ số giảm tỷ lệ học. Nếu các tham số này được đặt không đúng cách, quá trình tối ưu hóa có thể không hội tụ hoặc hội tụ quá chậm.

Có thể bị mắc kẹt ở điểm cực tiểu cục bộ: Tương tự như RMSProp, Adam cũng có thể bị mắc kẹt ở các điểm cực tiểu cục bộ và không đạt được điểm cực tiểu toàn cục. Điều này có thể xảy ra nếu tỷ lệ học được đặt quá cao hoặc quá thấp.

Tóm lại, Adam là một phương pháp tối ưu hóa mạnh mẽ và phổ biến trong mô hình học máy. Nó cung cấp lợi ích về tính ổn định, hiệu quả và tự động điều chỉnh tỷ lệ học. Tuy nhiên, việc cân nhắc tham số và nguy cơ mắc kẹt ở điểm cực tiểu cục bộ là nhược điểm cần được xem xét khi sử dụng Adam.

1.3 Thí nghiệm và kết quả

1.3.1 Thiết lập thí nghiệm

- Bộ dữ liệu: Sử dụng một số bộ dữ liệu phổ biến như MNIST, CIFAR-10, …
- Kiến trúc mô hình: Chọn kiến trúc mô hình thích hợp cho từng bài toán, *Ví dụ:* mạng nơ-ron sâu (DNN) cho MNIST và mạng nơ-ron tích chập (CNN) cho CIFAR-10.
- Thiết lập Optimizer: Sử dụng Gradient Descent, Stochastic Gradient Descent, Momentum, RMSProp, và Adam như các phương pháp Optimizer.
- Siêu tham số: Điều chỉnh các siêu tham số như learning rate, hệ số momentum, và các siêu tham số khác theo từng phương pháp.

1.3.2 Đánh giá tốc độ hội tụ

1.3.2.1 Theo dõi độ mất mát

Sự giảm độ mất mát trên tập huấn luyện: Theo dõi sự giảm của độ mất mát trên tập huấn luyện qua các epoch. Ghi lại biểu đồ hoặc số liệu thể hiện xu hướng giảm này.

Sự giảm độ mất mát trên tập kiểm tra: Theo dõi sự giảm của độ mất mát trên tập kiểm tra qua các epoch. Đánh giá xem mô hình có khả năng tổng quát hóa tốt không.

1.3.2.2 Đánh giá tốc độ hội tụ

Kiểm tra sự hội tụ đầu tiên: Xác định epoch mà độ mất mát trên tập huấn luyện bắt đầu giảm đáng kể (đạt đến điểm hội tụ đầu tiên). So sánh giữa các phương pháp Optimizer.

Xem xét tốc độ hội tụ trên tập kiểm tra: Đánh giá sự hội tụ trên tập kiểm tra bằng cách xem xét độ giảm mất mát qua các epoch. So sánh tốc độ hội tụ giữa các phương pháp.

Phân tích độ dao động: Xem xét độ dao động của độ mất mát trong quá trình huấn luyện. Đánh giá độ ổn định của tốc độ hội tụ.

1.3.2.3 Kết quả dự kiến:

Tốc độ hội tụ nhanh: Phương pháp Adam thường có tốc độ hội tụ nhanh đặc biệt trên các bài toán phức tạp. Momentum cũng thường cho thấy tốc độ hội tụ nhanh.

Sự ổn định của RMSProp: RMSProp có thể mang lại sự ổn định trong quá trình học trên các bề mặt hàm mất mát biến động.

Đánh giá sự hội tụ trên tập kiểm tra: So sánh tốc độ hội tụ giữa các phương pháp trên tập kiểm tra để đánh giá tính tổng quát hóa của mô hình.

1.3.3 Kết quả và nhận xét:

1.3.3.1 Tùy chọn thích hợp cho bài toán:

Stochastic Gradient Descent (SGD) thường được ưa chuộng trên dữ liệu lớn. Tuy nhiên, cũng cần lưu ý rằng SGD có thể có nhược điểm là độ dao động lớn trên bề mặt hàm mất mát không lồi và có thể mất thời gian để hội tụ đến điểm tối ưu. Điều này có thể được giảm bớt bằng cách sử dụng các biến thể như Momentum hoặc sự kiểm soát thông qua việc điều chỉnh learning rate.

Momentum và RMSProp thường được ưa chuộng khi đối mặt với các bài toán có hàm mất mát dao động. Kết hợp cả hai phương pháp có thể mang lại ưu điểm của cả hai, tạo ra phương pháp như Adam. Tuy nhiên, cũng cần lưu ý rằng việc lựa chọn giữa Momentum, RMSProp và Adam còn phụ thuộc vào đặc tính cụ thể của bài toán và dữ liệu, và việc điều chỉnh siêu tham số là quan trọng để đạt được hiệu suất tốt nhất.

Adam thường được coi là lựa chọn tổng quát và hiệu quả cho đa dạng các bài toán trong lĩnh vực học máy. Tuy nhiên, cũng cần lưu ý rằng không có một phương pháp tối ưu hóa nào phù hợp cho mọi tình huống. Việc lựa chọn giữa các phương pháp Optimizer phụ thuộc vào đặc tính cụ thể của bài toán, dữ liệu, và thậm chí là các điều kiện đào tạo cụ thể. Thường xuyên kiểm tra và điều chỉnh siêu tham số là quan trọng để đạt được hiệu suất tốt nhất.

1.3.3.2 Tốc độ hội tụ

Xác định phương pháp nào có tốc độ hội tụ nhanh nhất và giảm thiểu độ mất mát hiệu quả.

Nếu bề mặt hàm mất mát là lồi và không có nhiều dao động, GD có thể là lựa chọn nhanh chóng và hiệu quả.

Đối với các bề mặt phức tạp và dao động, Adam thường là sự kết hợp tốt nhất giữa tốc độ hội tụ và độ ổn định, nhưng cần sự điều chỉnh kỹ lưỡng của các siêu tham số.

1.3.3.3 Độ chính xác

Đánh giá độ chính xác của mỗi phương pháp để xác định khả năng của chúng trong việc tối ưu hóa mô hình.

Độ chính xác của mỗi phương pháp phụ thuộc vào bài toán cụ thể và dữ liệu.

SGD thường hiệu quả trên dữ liệu lớn nhưng có khả năng dao động.

Adam thường được xem là lựa chọn tổng quát và hiệu quả cho đa dạng các bài toán, nhưng yêu cầu điều chỉnh kỹ lưỡng của siêu tham số.

1.4 Xem xét sự ứng dụng trong các lĩnh vực cụ thể

Tùy thuộc vào tính chất của dữ liệu và bài toán cụ thể, mỗi phương pháp tối ưu hóa có thể có ưu điểm hoặc được ứng dụng tốt trong các lĩnh vực khác nhau. Dưới đây là một số ví dụ về cách mỗi phương pháp có thể được áp dụng:

- Gradient Descent (GD): GD thường được sử dụng trong các bài toán hồi quy tuyến tính và tối ưu hóa mô hình học máy truyền thống. Nó thích hợp cho các bài toán có tập dữ liệu nhỏ và không có nhiều nhiễu. GD có thể được sử dụng trong việc tối ưu hóa các hàm mất mát đơn giản và không đòi hỏi nhiều tính toán.
- Stochastic Gradient Descent (SGD): SGD thường được sử dụng trong các bài toán học máy với tập dữ liệu lớn. Vì SGD chỉ cần cập nhật trọng số dựa trên một mẫu dữ liệu, nó giúp tăng tốc quá trình huấn luyện.

SGD có thể được ứng dụng trong việc tối ưu hóa các mô hình có tập dữ liệu lớn và cần tính toán nhanh.

- Momentum: Momentum thường được sử dụng trong các bài toán tối ưu hóa mô hình học máy với độ cong lớn hoặc có nhiễu trong dữ liệu. Nó giúp giảm sự dao động và giúp vượt qua các điểm tối thiểu cục bộ. Momentum có thể được áp dụng trong việc tối ưu hóa các mô hình với hàm mất mát phức tạp và không đồng đều.
- RMSProp: RMSProp thường được sử dụng trong các bài toán tối ưu hóa mô hình học máy với độ lệch gradient lớn. Nó giúp điều chỉnh tỷ lệ học cho từng tham số riêng biệt và giảm độ lớn của gradient. RMSProp có thể được áp dụng trong việc tối ưu hóa các mô hình với hàm mất mát không đều và đòi hỏi sự điều chỉnh tỷ lệ học linh hoạt.
- Adam: Adam là một phương pháp tối ưu hóa linh hoạt và hiệu quả trong nhiều bài toán. Nó kết hợp các ưu điểm của Momentum và RMSProp và thường cho kết quả tốt. Adam có thể được áp dụng trong nhiều lĩnh vực và bài toán, bao gồm cả mô hình học sâu và mô hình học máy.

Tuy nhiên, việc chọn phương pháp tối ưu hóa phụ thuộc vào nhiều yếu tố, bao gồm tính chất của dữ liệu, kích thước tập dữ liệu, độ lớn của gradient và độ phức tạp của mô hình. Đôi khi, việc thử nghiệm và so sánh các phương pháp khác nhau là cần thiết để đạt được kết quả tốt nhất.

CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Giới thiệu

Continual Learning là một khái niệm trong lĩnh vực Machine Learning, nó đề cập đến khả năng của một hệ thống học máy để liên tục học tập và nắm bắt được kiến thức mới mà không quên đi những kiến thức đã học trước đó. Thay vì chỉ học từ một tập dữ liệu cố định, hệ thống Continual Learning có khả năng liên tục học từ các tập dữ liệu mới và cải thiện hiệu suất của mô hình một cách liên tục.

Test Production là quá trình tạo ra các bộ dữ liệu kiểm tra để đánh giá và kiểm tra hiệu suất của mô hình máy học. Bộ dữ liệu kiểm tra được thiết kế để kiểm tra khả năng tổng quát hóa của mô hình, tức là khả năng áp dụng kiến thức đã học từ dữ liệu huấn luyện vào các dữ liệu mới, chưa từng được nhìn thấy trước đó.

Continual Learning và Test Production có ứng dụng quan trọng trong lĩnh vực Machine Learning và phát triển phần mềm:

- Machine Learning: Continual Learning giúp mô hình máy học có khả năng học tập liên tục và cải thiện hiệu suất theo thời gian. Điều này rất hữu ích khi có sự thay đổi trong dữ liệu đầu vào hoặc khi có dữ liệu mới được thêm vào. Test Production giúp đánh giá hiệu suất của mô hình và đảm bảo tính tổng quát hóa của nó trước khi triển khai vào môi trường thực tế.
- Phát triển phần mềm: Continual Learning và Test Production cũng có ý nghĩa trong việc phát triển phần mềm. Continual Learning giúp đảm bảo rằng hệ thống sẽ không quên được kiến thức đã học trước đó và có thể liên tục cải thiện hiệu suất của mô hình theo thời gian. Test Production giúp đảm bảo rằng phần mềm hoạt động đúng và hiệu quả trong mọi trường hợp, từ đó đảm bảo chất lượng và độ tin cậy của phần mềm được phát triển.

Tóm lại, Continual Learning và Test Production đóng vai trò quan trọng trong việc đảm bảo hiệu suất và tính tổng quát hóa của mô hình máy học và phát triển phần mềm. Chúng giúp mô hình học máy cải thiện theo thời gian và đảm bảo rằng phần mềm hoạt động đúng và hiệu quả.

2.2 Continual Learning

2.2.1 Khái niệm và nguyên tắc cơ bản của Continual Learning

Continual Learning là khái niệm trong Machine Learning đề cập đến khả năng của một hệ thống học máy để tiếp tục học tập và nắm bắt kiến thức mới mà không quên đi những kiến thức đã học trước đó. Điều này đặc biệt hữu ích trong các tình huống khi có sự thay đổi trong dữ liệu đầu vào hoặc khi có dữ liệu mới được thêm vào.

Tuy nhiên, việc học liên tục cũng đặt ra một số thách thức. Một trong những thách thức đó là hiện tượng "catastrophic forgetting", tức là khi mô hình học máy học kiến thức mới, nó có thể quên đi kiến thức đã học trước đó. Điều này gây ra sự mất mát thông tin quan trọng và làm giảm hiệu suất của mô hình.

Một thách thức khác của việc học liên tục là việc quản lý bộ nhớ. Với việc học từ dữ liệu mới, kích thước của bộ nhớ tăng lên theo thời gian. Điều này có thể tạo ra áp lực về tài nguyên và ảnh hưởng đến hiệu suất của mô hình.

So sánh với học máy truyền thống, học liên tục tập trung vào việc cải thiện hiệu suất của mô hình theo thời gian và khả năng học từ dữ liệu mới. Trong khi đó, học máy truyền thống thường chỉ học từ một tập dữ liệu cố định và không có khả năng học liên tục.

Để giải quyết các thách thức của học liên tục, có nhiều nguyên tắc cơ bản được đề xuất. Một số trong số đó bao gồm:

- Giữ lại kiến thức quan trọng: Hệ thống cần có khả năng nhớ và giữ lại kiến thức quan trọng đã học trước đó, để tránh hiện tượng quên thông tin quan trọng.
- Quản lý bộ nhớ: Hệ thống cần có khả năng quản lý bộ nhớ và điều chỉnh kích thước của nó để đảm bảo hiệu suất và tối ưu hóa tài nguyên.
- Học từ dữ liệu mới: Hệ thống cần có khả năng học từ dữ liệu mới và tích hợp kiến thức này vào mô hình hiện tại một cách có hiệu quả.

Tóm lại, Continual Learning đặt ra một số thách thức và yêu cầu các nguyên tắc cơ bản để đảm bảo rằng hệ thống học máy có khả năng liên tục học tập và nắm bắt kiến thức mới mà không quên đi những kiến thức đã học trước đó.

2.2.2 Phương pháp và thuật toán trong Continual Learning

Trong Continual Learning, có nhiều phương pháp và thuật toán đã được đề xuất để giải quyết các thách thức liên quan đến việc học liên tục. Dưới đây là một số phương pháp phổ biến và thuật toán:

- Elastic Weight Consolidation (EWC): EWC là một phương pháp đưa ra bởi Kirkpatrick et al. (2017) để giải quyết hiện tượng quên thông tin quan trọng khi học từ dữ liệu mới. EWC đo lường mức độ quan trọng của các trọng số trong mô hình đối với hiệu suất của mô hình trên các tác vụ đã học trước đó. Sau đó, nó áp dụng một hàm mất mát phụ để bảo vệ các trọng số quan trọng đã học trước đó khỏi sự thay đổi quá mức khi học từ dữ liệu mới.
- Synaptic Intelligence: Synaptic Intelligence là một phương pháp đề xuất bởi Zenke et al. (2017) để đo lường mức độ quan trọng của các trọng số trong mô hình dựa trên thay đổi gradient của chúng trong quá

trình học. Phương pháp này cũng áp dụng một hàm mất mát phụ để bảo vệ các trọng số quan trọng đã học trước đó khỏi sự thay đổi quá mức.

- GEM (Gradient Episodic Memory): GEM đề xuất bởi Lopez-Paz và Ranzato (2017) là một phương pháp để học liên tục trong một môi trường không tham gia. GEM sử dụng bộ nhớ episodic để lưu trữ các mẫu dữ liệu đã học trước đó và sử dụng chúng để giới hạn sự thay đổi của các trọng số trong quá trình học mới.
- iCaRL (Incremental Classifier and Representation Learning): iCaRL là một phương pháp đề xuất bởi Rebuffi et al. (2017) để học liên tục trong bài toán phân loại. Phương pháp này kết hợp việc học lại các trọng số của mô hình và cập nhật một bộ nhớ gồm các mẫu đại diện cho các lớp đã học trước đó để tăng cường khả năng phân loại.

Đánh giá hiệu suất và ưu điểm, nhược điểm của từng phương pháp phụ thuộc vào các tình huống cụ thể và bài toán được áp dụng. Tuy nhiên, một số ưu điểm chung của các phương pháp này bao gồm:

- Khả năng giảm thiểu hiện tượng quên thông tin quan trọng khi học liên tục.
- Khả năng học từ dữ liệu mới mà không cần học lại từ đầu.
- Khả năng tối ưu hóa tài nguyên bằng cách quản lý bộ nhớ hoặc sử dụng bộ nhớ episodic.

Tuy nhiên, các phương pháp này cũng có nhược điểm như:

- Đòi hỏi tính toán phức tạp và tài nguyên cao.
- Cần sự điều chỉnh thích hợp của các siêu tham số để đạt hiệu suất tốt.
- Khả năng áp dụng của các phương pháp này có thể bị giới hạn trong một số bài toán cụ thể.

Tóm lại, mỗi phương pháp và thuật toán trong Continual Learning có ưu điểm và nhược điểm riêng, và việc lựa chọn phương pháp phụ thuộc vào yêu cầu và điều kiện cụ thể của bài toán.

2.2.3 Ứng dụng và case studies

Case studies về việc triển khai Continual Learning trong các dự án thực tế chưa được cụ thể đề cập. Tuy nhiên, các công ty công nghệ lớn như Google, Facebook và OpenAI đã tiến hành nghiên cứu và triển khai Continual Learning trong các dự án của họ. Nghiên cứu và ứng dụng Continual Learning trong các dự án thực tế đang tiếp tục được phát triển và nghiên cứu.

Continual Learning có nhiều ứng dụng trong các lĩnh vực như Computer Vision, Natural Language Processing và Robotics. Dưới đây là một số ứng dụng và case studies của Continual Learning.

2.2.3.1 Computer Vision

Phát hiện và nhận dạng đối tượng: Continual Learning có thể được sử dụng để phát triển hệ thống phát hiện và nhận dạng đối tượng liên tục. Hệ thống có thể tiếp tục học từ dữ liệu mới mà không cần học lại từ đầu, đồng thời giữ được khả năng phân loại các đối tượng đã học trước đó.

Phân tích hình ảnh y tế: Continual Learning có thể được áp dụng trong việc phân tích hình ảnh y tế để học liên tục từ dữ liệu mới và cải thiện khả năng chẩn đoán và dự đoán.

2.2.3.2 Natural Language Processing (NLP)

Xử lý ngôn ngữ tự nhiên: Continual Learning có thể được áp dụng trong việc xây dựng mô hình NLP để học từ dữ liệu mới và cải thiện khả năng hiểu và xử lý ngôn ngữ tự nhiên.

Dịch máy: Continual Learning cũng có thể được sử dụng trong việc phát triển hệ thống dịch máy liên tục, cho phép hệ thống học từ các bộ dữ liệu mới và cải thiện khả năng dịch.

2.2.3.3 Robotics:

Điều khiển robot: Continual Learning có thể được áp dụng trong việc phát triển hệ thống điều khiển robot để học liên tục từ môi trường và tác vụ mới. Điều này giúp robot thích nghi và cải thiện khả năng thực hiện các tác vụ trong môi trường đa dạng.

Nhận dạng vật thể và môi trường: Continual Learning cũng có thể được sử dụng để phát triển hệ thống nhận dạng vật thể và môi trường trong robot, cho phép robot học từ các đối tượng và môi trường mới mà không cần học lại từ đầu.

2.3 Test Production

2.3.1 Định nghĩa và quá trình Test Production

Test Production là quá trình triển khai, kiểm thử và đảm bảo chất lượng trong môi trường sản xuất để đảm bảo rằng sản phẩm hoặc dịch vụ đáp ứng các tiêu chuẩn và yêu cầu chất lượng được đặt ra.

Mục tiêu của Test Production là đảm bảo rằng sản phẩm hoặc dịch vụ đáp ứng các yêu cầu chất lượng đã được đặt ra và hoạt động một cách đáng tin cậy trong môi trường thực tế. Test Production cũng giúp xác định và loại bỏ các lỗi, sự cố, và rủi ro có thể xảy ra trong môi trường sản xuất, từ đó nâng cao độ tin cậy và hiệu suất của sản phẩm hoặc dịch vụ.

Phạm vi của Test Production bao gồm các hoạt động sau:

- Triển khai: Đây là quá trình triển khai sản phẩm hoặc dịch vụ trong môi trường sản xuất thực tế. Quá trình này có thể bao gồm việc cài đặt, cấu hình và kết nối các thành phần của hệ thống.
- Kiểm thử: Đây là quá trình kiểm tra và đánh giá tính năng, hiệu suất và chất lượng của sản phẩm hoặc dịch vụ trong môi trường sản xuất. Kiểm thử có thể bao gồm kiểm tra chức năng, kiểm tra bảo mật, kiểm thử hiệu suất và kiểm thử tải.
- Đảm bảo chất lượng: Đây là quá trình đảm bảo rằng sản phẩm hoặc dịch vụ đáp ứng các tiêu chuẩn chất lượng đã được đặt ra. Đảm bảo chất lượng có thể bao gồm việc xác nhận và đánh giá các yêu cầu chất lượng, kiểm tra và giám sát quá trình sản xuất, và đảm bảo rằng quy trình và tiêu chuẩn chất lượng được tuân thủ.

Quá trình triển khai, kiểm thử và đảm bảo chất lượng trong môi trường sản xuất là một quá trình liên tục và phức tạp, yêu cầu sự cộng tác giữa các bộ phận khác nhau trong tổ chức. Nó bao gồm việc lên kế hoạch, thiết kế và thực hiện các bài kiểm

tra, giám sát và phân tích kết quả kiểm thử, và đưa ra các biện pháp cải thiện nếu cần thiết để đảm bảo chất lượng sản phẩm hoặc dịch vụ.

2.3.2 Kiến thức và kỹ thuật liên quan đến Test Production

Continuous Integration/Continuous Deployment (CI/CD) là một phương pháp phát triển phần mềm trong đó các thay đổi trong mã nguồn được tích hợp và triển khai tự động vào môi trường sản xuất một cách liên tục. Quá trình CI/CD thường bao gồm việc sử dụng công cụ và quy trình tự động để kiểm tra, xây dựng, đóng gói và triển khai các phiên bản của phần mềm.

Testing strategies là các chiến lược kiểm thử được sử dụng để đảm bảo chất lượng của sản phẩm hoặc dịch vụ trong quá trình Test Production. Các chiến lược kiểm thử phổ biến bao gồm:

- Unit Testing: Kiểm thử đơn vị tập trung vào việc kiểm tra từng đơn vị (hàm, module) của mã nguồn để đảm bảo rằng chúng hoạt động đúng và đáp ứng yêu cầu chức năng.
- Integration Testing: Kiểm thử tích hợp kiểm tra sự tương tác giữa các thành phần trong hệ thống để đảm bảo rằng chúng hoạt động đúng và liên kết với nhau một cách chính xác.
- End-to-End Testing: Kiểm thử từ đầu đến cuối kiểm tra toàn bộ quy trình hoạt động của hệ thống từ đầu vào đến đầu ra. Nó kiểm tra tính đầy đủ của hệ thống và đảm bảo rằng tất cả các thành phần hoạt động một cách tương tác tốt.
- Automated Testing: Kiểm thử tự động sử dụng các công cụ và kịch bản kiểm thử được tự động hóa. Điều này giúp giảm thời gian và công sức cần thiết cho kiểm thử và đảm bảo tính nhất quán và đáng tin cậy của quá trình kiểm thử.

Các chiến lược kiểm thử này được sử dụng để xác định và loại bỏ các lỗi và rủi ro trong quá trình Test Production, đảm bảo rằng sản phẩm hoặc dịch vụ hoạt động đúng và đáp ứng yêu cầu chất lượng đã đặt ra.

2.3.3 Challenges và best practices trong Test Production

Triển khai và kiểm thử trong môi trường sản xuất đặt ra một số thách thức riêng. Dưới đây là một số thách thức phổ biến và những nguyên tắc và phương pháp tốt nhất để vượt qua chúng:

- Thách thức về môi trường: Môi trường sản xuất thường có sự phức tạp và đa dạng cao, với nhiều thành phần và phụ thuộc. Điều này có thể gây khó khăn trong việc triển khai và kiểm thử. Một nguyên tắc tốt nhất là tạo ra một môi trường kiểm thử tương tự với môi trường sản xuất để đảm bảo tính nhất quán và độ tin cậy của các kiểm thử.
- Thách thức về dữ liệu: Đôi khi, việc tạo dữ liệu kiểm thử phù hợp với môi trường sản xuất có thể là một thách thức. Một phương pháp tốt nhất là sử dụng dữ liệu thực tế hoặc dữ liệu tương tự với dữ liệu thực tế để đảm bảo kiểm thử đầy đủ và chính xác.
- Thách thức về tương tác hệ thống: Trong môi trường sản xuất, hệ thống thường phải tương tác với các thành phần và dịch vụ bên ngoài. Điều này có thể tạo ra khó khăn trong việc kiểm thử và đảm bảo tính nhất quán của hệ thống. Một nguyên tắc tốt nhất là sử dụng các kỹ thuật kiểm thử như stubs, mocks hoặc virtualization để mô phỏng các tương tác bên ngoài và kiểm thử hệ thống một cách độc lập.
- Thách thức về quản lý thay đổi: Trong môi trường sản xuất, sự thay đổi liên tục và thường xuyên xảy ra. Điều này có thể tạo ra khó khăn trong việc duy trì và kiểm thử các phiên bản mới của phần mềm. Một phương pháp tốt nhất là sử dụng quy trình CI/CD để tự động hóa việc triển khai và kiểm thử các thay đổi, đảm bảo rằng các phiên bản mới được triển khai một cách nhanh chóng và đáng tin cậy.

- Thách thức về hiệu suất và tải: Môi trường sản xuất thường phải đối mặt với khối lượng lớn người dùng và tải lớn. Điều này có thể tạo ra khó khăn trong việc kiểm thử hiệu suất và đảm bảo rằng hệ thống hoạt động một cách ổn định dưới tải cao. Một phương pháp tốt nhất là sử dụng công cụ kiểm thử hiệu suất để đo và tối ưu hóa hiệu suất của hệ thống.

Những nguyên tắc và phương pháp tốt nhất này giúp vượt qua những thách thức khi triển khai và kiểm thử trong môi trường sản xuất, đảm bảo tính nhất quán, đáng tin cậy và chất lượng của phần mềm.

2.4 Kết luận

Trên cơ sở các nghiên cứu và hiểu biết hiện tại, dự đoán về xu hướng và phát triển trong tương lai là sự gia tăng của Continual Learning và Test Production trong việc tạo ra và duy trì chất lượng phần mềm. Sự kết hợp của hai lĩnh vực này có thể giúp tăng cường khả năng học và thích nghi của mô hình, đồng thời giảm thiểu lỗi và tăng cường hiệu suất của quá trình phát triển phần mềm.

Tổng hợp lại thì tương lai của Continual Learning (học liên tục) và Test Production (kiểm thử sản xuất) là hai lĩnh vực quan trọng trong phát triển phần mềm. Dưới đây là một số điểm quan trọng và hiểu biết từ nghiên cứu, cũng như dự đoán về xu hướng và phát triển trong tương lai:

2.4.1 Continual Learning

Continual Learning là quá trình học liên tục và không dừng lại sau khi mô hình đã được huấn luyện ban đầu.

Mục tiêu của Continual Learning là giúp mô hình học và thích nghi với dữ liệu mới và giữ được kiến thức đã học trước đó.

Một trong những thách thức của Continual Learning là hiện tượng quên mất kiến thức cũ khi học kiến thức mới. Các phương pháp như Regularization và Replay được sử dụng để giải quyết vấn đề này.

Xu hướng trong tương lai là phát triển các phương pháp Continual Learning mạnh mẽ hơn, đáng tin cậy hơn và hiệu quả hơn để đáp ứng nhu cầu của ngành công nghiệp và ứng dụng thực tế.

2.4.2 Test Production

Test Production là quá trình tạo ra các bộ kiểm thử tự động trong quá trình phát triển phần mềm.

Mục tiêu của Test Production là tăng cường chất lượng phần mềm và giảm thiểu lỗi trong quá trình triển khai.

Một trong những thách thức của Test Production là tạo ra các bộ kiểm thử phù hợp và hiệu quả. Các phương pháp như tự động hóa kiểm thử, phân tích tĩnh và tạo dữ liệu kiểm thử tự động có thể được sử dụng để giải quyết vấn đề này.

Xu hướng trong tương lai là phát triển các công cụ và phương pháp Test Production tiên tiến hơn, đáp ứng nhu cầu của công nghiệp phần mềm ngày càng phức tạp hơn và đòi hỏi chất lượng cao hơn.

TÀI LIỆU THAM KHẢO

1. S. Ruder, "An overview of gradient descent optimization algorithms," 2016.
2. L. Bottou, F. E. Curtis, J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," 2016.
3. D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization," 2014.
4. G. Hinton, "Understanding RMSprop."
5. J. Kirkpatrick et al., "Continual Lifelong Learning with Neural Networks: A Review," 2018.
6. J. Kirkpatrick et al., "Overcoming Catastrophic Forgetting in Neural Networks"
7. A. Antoniou et al., "A Gentle Introduction to Continual Learning," 2019.
8. J. Humble, D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation," 2010.
9. N. Forsgren, J. Humble, G. Kim, "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations," 2018.