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

CAO HOÀNG OANH - 52100917

BÁO CÁO CUỐI KÌ
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

CAO HOÀNG OANH - 52100917

BÁO CÁO CUỐI KỲ
NHẬP MÔN HỌC MÁY

Người hướng dẫn
TS LÊ ANH CƯỜNG

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Trước tiên, tôi xin gửi lời cảm ơn chân thành đến thầy Lê Anh Cường, thầy đã luôn hỗ trợ cho tôi trong thời gian học tập trên lớp và cuối cùng là báo cáo cuối kì này. Thầy luôn giải đáp tất cả các thắc mắc cũng như hướng dẫn tận tình trong suốt thời gian học vừa qua.

Tiếp theo, tôi xin gửi lời cảm ơn đến khoa Công nghệ thông tin trường Đại học Tôn Đức Thắng, khoa đã tạo điều kiện cho tôi có các tiếp xúc thực tế với môn học bằng việc làm các báo cáo nghiên cứu như thế này, giúp tôi có thêm nhiều kinh nghiệm cho các vị trí việc làm trong tương lai cũng như một lần tổng hợp lại kiến thức đã học được trong thời gian qua.

Cuối cùng, nhóm tôi xin chúc thầy cô giảng viên và khoa Công nghệ thông tin có thật nhiều sức khỏe, ngày càng phát triển để có thật nhiều thành tích cống hiến cho trường Đại học Tôn Đức Thắng cũng như trong lĩnh vực công nghệ nói chung.

TP. Hồ Chí Minh, ngày 15 tháng 12 năm 2023

Tác giả

Cao Hoàng Oanh

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

Tôi xin cam đoan đây là công trình nghiên cứu của tôi và được sự hướng dẫn khoa học của 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 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 15 tháng 12 năm 2023

Tác giả

Cao Hoàng Oanh

TÓM TẮT

Bài báo cáo này là quá trình tìm hiểu, so sánh các phương pháp Optimizer, các vấn đề về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

ABSTRACT

This report is the process of researching and comparing various optimizer methods, issues related to Continual Learning, and Test Production when building a machine learning solution to address a specific problem.

MỤC LỤC

DANH SÁCH HÌNH VẼ

PHẦN 1. OPTIMIZER

1.1 Giới thiệu Machine Learning

Machine learning (ML) là một nhánh của trí tuệ nhân tạo (AI) cho phép máy tính "tự học" từ dữ liệu huấn luyện và cải thiện theo thời gian mà không cần được lập trình một cách rõ ràng. Các thuật toán machine learning có khả năng phát hiện ra các mẫu trong dữ liệu và học từ chúng, nhằm tạo ra dự đoán riêng của mình. Nói một cách đơn giản, các thuật toán và mô hình machine learning học thông qua trải nghiệm.

Thay vì lập trình các thuật toán machine learning để thực hiện các nhiệm vụ, bạn có thể cung cấp cho chúng các ví dụ của dữ liệu đã được gán nhãn (gọi là dữ liệu huấn luyện), giúp chúng thực hiện các tính toán, xử lý dữ liệu và tự động nhận diện các mẫu.

Machine learning có thể được áp dụng vào lượng lớn dữ liệu và có thể thực hiện chính xác hơn nhiều so với con người. Nó có thể giúp bạn tiết kiệm thời gian và tiền bạc cho các nhiệm vụ và phân tích, như giải quyết các vấn đề gây khó chịu cho khách hàng để nâng cao sự hài lòng của khách hàng, tự động hóa các yêu cầu hỗ trợ, và khai thác dữ liệu từ các nguồn nội bộ.

1.2 Giới thiệu Continous Learning

Continuous learning (học liên tục), còn được biết đến là học máy liên tục (CML), là quá trình mà một mô hình học từ các luồng dữ liệu mới mà không cần được huấn luyện lại.

Khác với các phương pháp truyền thống, trong đó các mô hình được huấn luyện trên một bộ dữ liệu tĩnh, triển khai, và được huấn luyện lại định kỳ, các mô hình học liên tục liên tục cập nhật các tham số của chúng để phản ánh phân phối mới trong dữ liệu.

Trong quá trình sau, mô hình cải thiện chính mình bằng cách học từ các phiên bản mới nhất và cập nhật kiến thức của mình khi dữ liệu mới trở nên có sẵn. Vòng

đời của mô hình học liên tục cho phép các mô hình duy trì tính liên quan qua thời gian do tính động của chúng từ bản chất.

PHẦN 2. TÌM HIỂU VÀ SO SÁNH OPTIMIZER

2.1 Tổng quan về Optimizer

2.1.1 Khái niệm

Optimizer là các thuật toán điều chỉnh các tham số của mô hình trong quá trình huấn luyện để giảm thiểu một hàm mất mát. Chúng cho phép mạng neural học từ dữ liệu bằng cách cập nhật trọng số và độ lệch một cách lặp đi lặp lại. Mỗi bộ tối ưu hóa có các quy tắc cụ thể để cập nhật, tỷ lệ học và đà để tìm ra các tham số mô hình tối ưu hóa để cải thiện hiệu suất.

Các bộ tối ưu hóa phổ biến bao gồm:

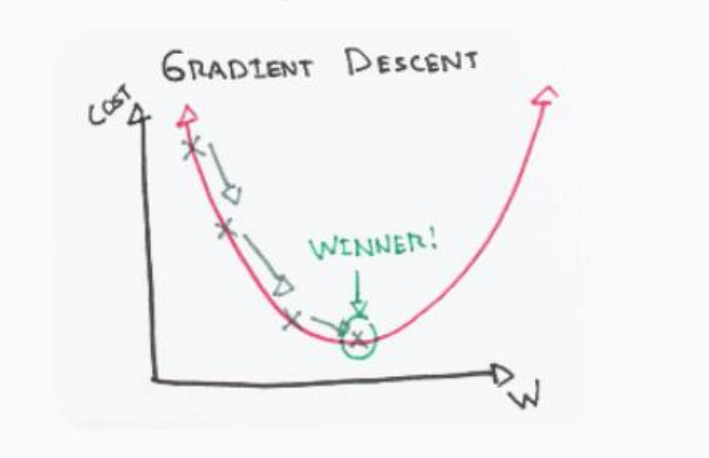
- Gradient Descent (SGD).
- Stochastic Gradient Descent.
- RMSprop.
- Adam.

2.1.2 Giới thiệu thuật toán

2.1.2.1 Gradient Descent (SGD)

Thuật toán Gradient Descent là thuật toán tối ưu hóa cơ bản nhất nhưng được sử dụng rộng rãi nhất. Nó được sử dụng mạnh mẽ trong hồi quy tuyến tính và các thuật toán phân loại.

Gradient descent là một thuật toán tối ưu hóa bậc một phụ thuộc vào đạo hàm bậc nhất của một hàm mất mát. Nó tính toán hướng mà trọng số cần được điều chỉnh để hàm có thể đạt đến điểm cực tiểu. Thông qua quá trình lan truyền ngược, sự mất mát được chuyển từ một lớp sang lớp khác và các tham số của mô hình, còn được gọi là trọng số, được điều chỉnh dựa trên các mất mát để hàm mất mát có thể được giảm thiểu.



Hình 2.1 Minh họa đồ thị thuật toán Gradient Descent

- Gradient Descent hoạt động như thế nào?
- Đầu tiên, ta khởi tạo ngẫu nhiên giá trị ban đầu cho các tham số.
- Từ điểm bắt đầu đó, chúng ta sẽ tìm đạo hàm (hoặc độ dốc).
- Cập nhật các tham số theo công thức trên để di chuyển đến vị trí mới trên không gian tham số, mục tiêu là làm giảm giá trị của hàm chi phí.

Công thức cơ bản của thuật toán Gradient Descent cho một tham số α trong mô hình là:

$$\alpha\text{new} = \alpha\text{old} - \gamma \frac{\partial J(\alpha)}{\partial \alpha}$$

Trong đó:

α old là giá trị của tham số tại vòng lặp trước đó.

α new là giá trị của tham số được cập nhật ở vòng lặp hiện tại.

γ là hệ số chờ, quyết định tốc độ học của thuật toán.

J(α) là hàm chi phí cần được tối thiểu hóa.

$\frac{\partial J(\alpha)}{\partial \alpha}$ là đạo hàm riêng của hàm chi phí, chỉ hướng và độ lớn của thay đổi mong muốn.

- Lặp lại các bước 2 và 3 cho đến khi hàm chi phí không thay đổi đáng kể hoặc đạt tới một ngưỡng được xác định trước.

Ưu và nhược điểm của Gradient Descent:

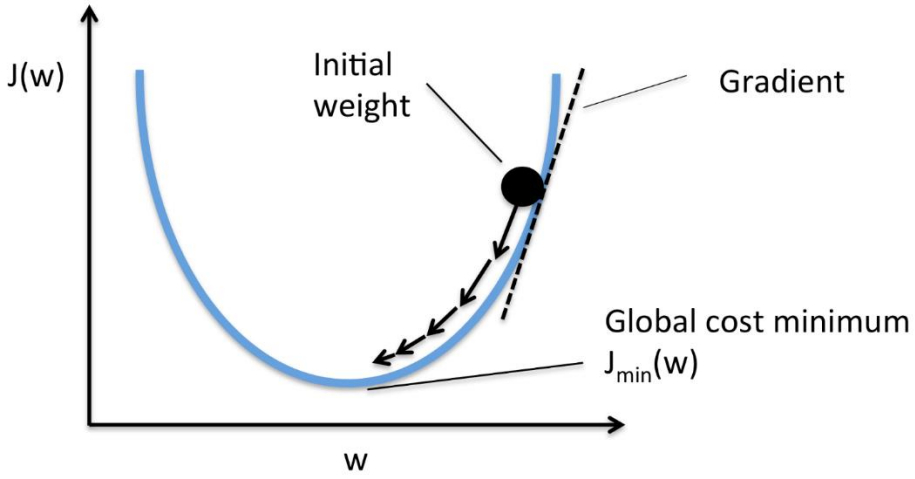
- Ưu điểm:
    - Tính toán dễ dàng.
    - Dễ triển khai.
    - Dễ hiểu.
- Nhược điểm:
    - Có thể rơi vào cực tiểu cục.
    - Trọng số được thay đổi sau khi tính đạo hàm trên toàn bộ tập dữ liệu. Vì vậy, nếu tập dữ liệu quá lớn thì có thể mất nhiều năm để hội tụ đến điểm tối thiểu.
    - Yêu cầu bộ nhớ lớn để tính toán đạo hàm trên toàn bộ tập dữ liệu.

2.1.2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của thuật toán Gradient Descent được sử dụng để tối ưu hóa các mô hình học máy. Nó giải quyết vấn đề không hiệu quả về tính toán của các phương pháp Gradient Descent truyền thống khi xử lý các bộ dữ liệu lớn trong các dự án học máy.

Trong SGD, thay vì sử dụng toàn bộ bộ dữ liệu cho mỗi lần lặp, chỉ một ví dụ huấn luyện ngẫu nhiên (hoặc một lô nhỏ) được chọn để tính toán độ dốc và cập nhật các tham số mô hình. Việc lựa chọn ngẫu nhiên này đưa vào quá trình tối ưu hóa sự ngẫu nhiên, vì vậy cụm từ "stochastic" (ngẫu nhiên) trong "stochastic Gradient Descent".

Lợi ích của việc sử dụng SGD là tính hiệu quả về tính toán, đặc biệt là khi xử lý các bộ dữ liệu lớn. Bằng cách sử dụng một ví dụ duy nhất hoặc một lô nhỏ, chi phí tính toán cho mỗi lần lặp được giảm đáng kể so với các phương pháp Gradient Descent truyền thống đòi hỏi xử lý toàn bộ bộ dữ liệu.



Hình 2.2 *Minh họa thuật toán Stochastic Gradient Descent*

Công thức của thuật toán Stochastic Gradient Descent:

$$w = w - \eta \nabla Qi(w)$$

Trong đó:

w là bộ tham số cần cập nhật.

η là tỷ lệ học (learning rate), đại diện cho bước di chuyển.

$\nabla Qi(w)$ là độ dốc của hàm mất mát.

Stochastic Gradient Descent hoạt động như thế nào?

1. Khởi tạo giá trị ban đầu cho các tham số w và hệ số học tập η
2. Tính toán độ dốc
3. Cập nhật tham số
4. Lặp lại quá trình.

Ưu và nhược điểm của Stochastic Gradient Descent:

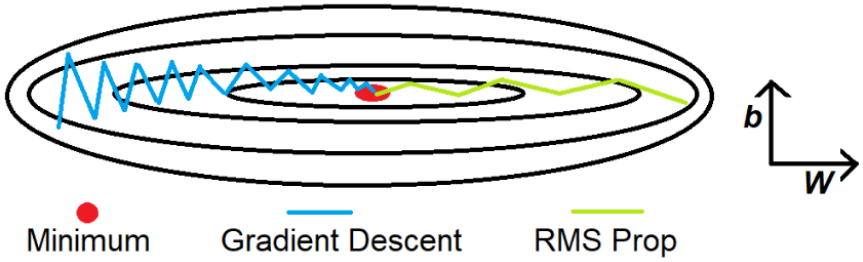
- Ưu điểm:
  - Dễ dàng lưu vào bộ nhớ
  - Tính toán nhanh chóng
  - Hội tụ nhanh hơn cho các tập dữ liệu lớn
  - Vượt qua điểm cực tiểu cục bộ

- Nhược điểm:
  - Các bước di chuyển không ổn định
  - Cần thời gian để hội tụ
  - Chi phí tính toán cao
  - Mất đi ưu điểm của phép toán vector hóa

2.1.2.3 RMSprop

RMSProp, viết tắt của "Root Mean Square Propagation," là một thuật toán tối ưu hóa được sử dụng trong việc huấn luyện mạng neural. Nó giải quyết các vấn đề về việc gradient biến mất hoặc bùng nổ thường gặp trong các kiến trúc mạng neural phức tạp khi huấn luyện.

Đơn giản, RMSProp sử dụng tốc độ học thích nghi thay vì xem xét tốc độ học như một siêu tham số cố định. Điều này có nghĩa là tốc độ học thay đổi theo thời gian. RMSProp sử dụng cùng khái niệm về trung bình trượt theo hình mũi tên với gradient descent có đà, nhưng khác biệt ở cách cập nhật tham số.



Hình 2.3 Minh hoạt thuật toán RMSProp

Công thức của thuật toán RMSProp

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

Trong đó:

$\theta t+1$ là giá trị cập nhật mới của trọng số.

θt là giá trị trọng số hiện tại

gt là gradient của hàm mất mát tại thời điểm t.

$E[g2]t$ là trung bình có trọng số của bình phương của gradient cho mỗi trọng số.

η là tốc độ học (learning rate).

ϵ là một số nhỏ được thêm vào trong mẫu để tránh chia cho 0.

Thuật toán RMSProp hoạt động như thế nào?

1. Khởi tạo một giá trị ban đầu cho trung bình trượt bình phương của gradient
2. Tính toán gradient của hàm mất mát đối với các tham số mô hình tại mỗi vòng lặp.
3. Cập nhật trung bình trượt bình phương của gradient bằng cách tính trung bình trượt có trọng số của gradient hiện tại và trung bình trượt bình phương của gradient trước đó.
4. Sử dụng tham số tỷ lệ học để cập nhật tham số của mô hình
5. Lặp lại các bước trên cho đến khi tiêu chí dừng được đáp ứng hoặc số lần lặp đã đủ.

Ưu và nhược điểm của RMSProp

- Ưu điểm:
  - Tốc độ hội tụ nhanh
  - Tốc độ học ổn định
  - Hiệu suất tốt
- Nhược điểm:
  - Không phải là giải pháp tức thì cho mọi vấn đề
  - Thiếu hỗ trợ lý thuyết

2.1.2.4 Adam (Adaptive Moment Estimation)

Adam Optimizer, viết tắt của "Adaptive Moment Estimation" là một thuật toán tối ưu hóa lặp để giảm thiểu hàm mất mát trong quá trình huấn luyện mạng neural. Adam có thể xem là sự kết hợp giữa RMSprop và Gradient Descent ngẫu nhiên với

Momentum. Nó sử dụng gradient bình phương để điều chỉnh tỷ lệ học giống như RMSprop, và tận dụng **Momentum** bằng cách tính toán trung bình có trọng số của các gradient trước đó, như SGD với Momentum.

Momentum là gì?

Momentum trong ngữ cảnh của thuật toán tối ưu hóa Gradient Descent trong Machine Learning là một phương pháp giúp tăng đà cho việc cập nhật các tham số của mô hình dựa trên gradient. Nó thêm vào một phần của vector cập nhật trước đó vào vector cập nhật hiện tại để giúp thuật toán vượt qua các khe hẹp và tìm được điểm cực tiểu tốt hơn. Trong thuật toán, momentum cũng làm tăng tốc độ cập nhật tham số và giúp tránh được sự dao động không cần thiết, giúp thuật toán hội tụ nhanh hơn.

Công thức cập nhật của momentum trong Gradient Descent có thể được biểu diễn như sau:

$$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla J(\theta_t)$$

Trong đó:

v_t là momentum tại thời điểm tt

γ là hệ số momentum, thường có giá trị từ 0 đến 1

v_{t-1} là momentum tại bước thời gian trước đó t−1

η là tỷ lệ học (learning rate)

$\nabla J(\theta_t)$ là gradient của hàm mất mát J tại tham số θ_t

Công thức cập nhật của thuật toán tối ưu hóa Adam có thể được biểu diễn như sau:

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

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

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

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

$$w_{t+1} = w_t - \frac{\alpha\, \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Trong đó:

m_t là hướng và độ lớn của gradient bậc nhất theo thời gian.

v_t là bình phương của gradient hiện tại gtgt và kết hợp với moment của gradient bậc hai từ các bước trước đó

g_t là gradient tại bước thời gian t

$\hat{m}_t$ và $\hat{v}_t$ là các ước lượng của moment đã được chỉnh sửa để khắc phục bias

α_t và α_{t+1} là các bộ tham số tại các thời điểm t và t+1tương ứng

α là tỷ lệ học (learning rate)

β_1 và β_2 là các hệ số giảm dần (decay rates) cho moment của bậc nhất và bậc hai

ϵ là một hằng số nhỏ để tránh chia cho 0

Các bước trong thuật toán tối ưu hóa Adam:

1. Khởi tạo trung bình di động của moment bậc nhất và bậc hai (m và v) bằng không.
2. Tính toán độ dốc của hàm mất mát đối với các tham số mô hình.
3. Cập nhật trung bình di động sử dụng trung bình giảm dần một cách mũi nhọn. Điều này bao gồm tính toán m_t và v_t như trung bình có trọng số của các khoảnh khắc trước đó và độ dốc hiện tại.
4. Áp dụng sự điều chỉnh sai lệch vào trung bình di động, đặc biệt là trong các vòng lặp ban đầu.
5. Tính toán cập nhật tham số bằng cách chia moment bậc nhất được điều chỉnh sai lệch cho căn bậc hai của moment bậc hai được điều chỉnh sai lệch, với một hằng số nhỏ được thêm vào (epsilon) để đảm bảo tính ổn định số học.
6. Cập nhật các tham số mô hình bằng cách sử dụng các cập nhật đã tính toán.
7. Lặp lại các bước từ 2 đến 6 cho một số lần lặp cụ thể hoặc cho đến khi hội tụ.

Ưu và nhược điểm của thuật toán tối ưu Adam:

- Ưu điểm:
  - Hội tụ Nhanh.
  - Tỷ lệ Học Thích Ứng.
  - Sử Dụng Bộ Nhớ Thấp.
  - Độ Ổn Định.
- Nhược điểm:
  - Độ nhạy cảm đối với siêu tham số
  - Hiệu ứng bộ nhớ
  - Có thể tăng chi phí
  - Bảo đảm lý thuyết hạn chế

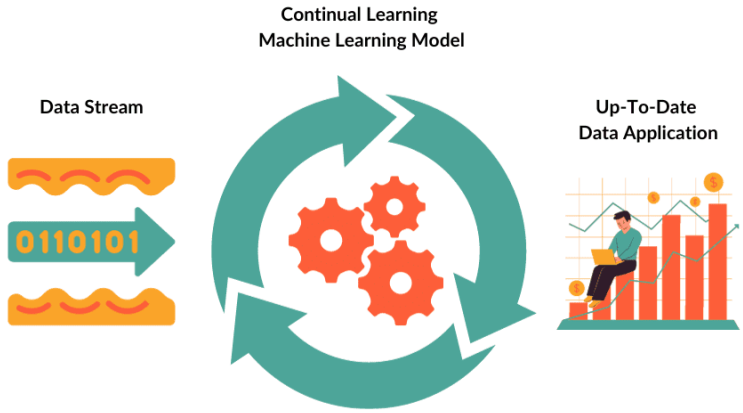
PHẦN 3. CONTINUAL LEARNING & TEST PRODUCTION

3.1 Continual Learning

Học liên tục, hay còn gọi là học máy liên tục (Continuous Machine Learning-CML), là quá trình mà một mô hình học từ dòng dữ liệu mới mà không cần phải được huấn luyện lại.

Ngược lại với các phương pháp truyền thống, trong đó các mô hình được huấn luyện trên một tập dữ liệu tĩnh, triển khai và định kỳ được huấn luyện lại, các mô hình học liên tục lặp đi lặp lại việc cập nhật tham số của mình để phản ánh các phân phối mới trong dữ liệu.

Trong quá trình này, mô hình tự cải thiện bản thân bằng cách học từ phiên bản mới nhất và cập nhật kiến thức của mình khi dữ liệu mới trở nên có sẵn. Vòng đời của mô hình học liên tục giúp mô hình duy trì sự phù hợp theo thời gian do tính động của nó.



Hình 3.1 Giới thiệu về Continual Learning

Một số thách thức liên quan đến Continual Learning:

- Catastrophic Forgetting
- Replay Buffers
- Regularization
- Transfer Learning
- Meta-Learning

3.1.1 Catastrophic Forgetting

3.1.1.1 Định nghĩa

Catastrophic Forgetting là hiện tượng xảy ra trong học máy và trí tuệ nhân tạo, đặc biệt là trong các kịch bản liên quan đến học liên tục hoặc học suốt đời. Nó đề cập đến xu hướng của các mạng neural và các thuật toán học khác để quên thông tin đã học trước đó khi được huấn luyện trên dữ liệu hoặc nhiệm vụ mới không liên quan. Điều này dẫn đến việc mất kiến thức và sự suy giảm hiệu suất trên các nhiệm vụ đã học trước đó.

3.1.1.2 Nguyên nhân

Catastrophic forgetting có thể được gây ra bởi một số nguyên nhân chính:

- Interference between old and new information: Khi một mô hình học thông tin mới, thông tin này có thể ghi đè lên thông tin cũ đã học trước đó. Điều này gây ra sự giao thoa giữa dữ liệu cũ và mới, dẫn đến mất mát hoặc biến đổi không mong muốn trong kỹ năng của mô hình trên các nhiệm vụ cũ.
- Distributional shift: Dữ liệu mới có thể có phân phối khác so với dữ liệu đã được sử dụng để huấn luyện mô hình ban đầu. Nếu mô hình không thích nghi tốt với sự thay đổi này, nó có thể quên đi kiến thức trước đó.
- Limited capacity: Mô hình có giới hạn về khả năng lưu trữ và xử lý thông tin. Khi thông tin mới được học, nó có thể làm đầy bộ nhớ hoặc tài nguyên tính toán của mô hình, dẫn đến việc xóa thông tin cũ.
- Overfitting: Khi mô hình quá tập trung vào dữ liệu huấn luyện cũ mà không thể tổng quát hóa cho dữ liệu mới, nó có thể dẫn đến việc mất đi khả năng thích nghi với thông tin mới mà không ảnh hưởng đến hiệu suất trên dữ liệu cũ.
- Inappropriate training strategies: Các chiến lược huấn luyện không tối ưu hoặc không phù hợp cho việc duy trì kiến thức trước đó khi học thông tin mới cũng có thể góp phần vào hiện tượng quên nhanh.

3.1.1.3 Cách giải quyết

Phương pháp được đề xuất để giải quyết vấn đề của Catastrophic forgetting trong học máy và mạng nơ-ron:

- Elastic Weight Consolidation (EWC): EWC thêm một thuật ngữ điều chuẩn vào hàm mất mát trong quá trình huấn luyện. Thuật ngữ này trừng phạt sự thay đổi trong trọng số của mạng nơ-ron dựa trên tầm quan trọng của chúng đối với các nhiệm vụ đã học trước đó, giúp mô hình giữ lại kiến thức cần thiết.
- Progressive Neural Networks: Phương pháp này bao gồm việc thêm mạng mới cho mỗi nhiệm vụ mới trong khi vẫn giữ kết nối với các mạng đã được huấn luyện trước đó. Điều này đảm bảo kiến thức cơ bản vẫn được giữ nguyên, và việc học mới được xây dựng dựa trên nó.
- Replay Techniques: Các phương pháp này bao gồm việc giữ lại một số dữ liệu từ các nhiệm vụ trước. Trong quá trình huấn luyện trên các nhiệm vụ mới, mô hình cũng được tiếp xúc với dữ liệu cũ này, ngăn chặn việc quên kiến thức trước đây.
- Meta-learning: Thay vì huấn luyện một mô hình để thực hiện các nhiệm vụ, meta-learning huấn luyện một mô hình để học cách học các nhiệm vụ. Bằng cách hiểu quá trình học tập chính nó, mô hình có thể trở nên linh hoạt hơn và chống lại việc quên thông tin.

3.1.2 Replay Buffers

3.1.2.1 Định nghĩa

Replay buffer là một cơ chế lưu trữ bộ nhớ lưu giữ một tập hợp con đại diện của các trải nghiệm hoặc điểm dữ liệu mà mô hình đã gặp trong quá trình huấn luyện. Các trải nghiệm trước đây này được tái phát hoặc lấy mẫu định kỳ cùng với dữ liệu mới trong quá trình huấn luyện. Điều này giúp mô hình không quên đi những gì đã học được từ các tác vụ trước đó. Replay buffer có thể không gây ra overfitting, nhưng

cách sử dụng của nó trong một số ngữ cảnh có thể góp phần vào việc gây overfitting nếu không được quản lý đúng cách.

3.1.2.2 Tình huống khi replay buffer góp phần vào việc gây overfitting

Hạn chế trong các trải nghiệm được lưu trữ: Nếu replay buffer chỉ chứa một bộ sưu tập hạn chế các trải nghiệm hoặc nó không bao phủ một loạt các kịch bản rộng, mô hình có thể overfitting với những trải nghiệm cụ thể đó. Về lâu dài, mô hình có thể quá phụ thuộc vào những trải nghiệm đã lưu, dẫn đến overfitting với những trường hợp cụ thể đó và thiếu khả năng tổng quát hóa với dữ liệu chưa từng gặp.

Lựa chọn mẫu không đủ đa dạng: Nếu chiến lược lấy mẫu từ replay buffer thiên về các loại trải nghiệm cụ thể hoặc nếu quá trình lấy mẫu bị thiên vị, nó có thể củng cố quá mức một số mẫu cụ thể, dẫn đến overfitting với những mẫu đó.

Lặp lại hoặc không đủ đa dạng trong các mẫu: Việc sử dụng liên tục các trải nghiệm giống nhau hoặc không đủ đa dạng từ buffer trong quá trình huấn luyện có thể làm hạn chế khả năng của mô hình tổng quát hóa với các kịch bản mới, gây ra overfitting với một tập hạn chế các dữ liệu đã được lưu.

3.1.2.3 Cách giảm rủi ro của overfitting khi sử dụng replay buffer

Lấy mẫu ngẫu nhiên: Đảm bảo rằng các trải nghiệm được lấy mẫu ngẫu nhiên từ replay buffer để giới thiệu độ đa dạng vào quá trình học.

Cơ chế ưu tiên hóa: Thực hiện cơ chế ưu tiên hoặc cân đối việc lựa chọn trải nghiệm dựa trên tính quan trọng hoặc liên quan đến nhiệm vụ học, ngăn chặn sự tập trung quá mức vào những trải nghiệm cụ thể.

Áp dụng các kỹ thuật regularization: Áp dụng các phương pháp regularization như dropout hoặc weight decay để ngăn mô hình trở nên quá chuyên sâu vào các trải nghiệm cụ thể.

3.1.3 Regularization

3.1.3.1 Định nghĩa

Regularization là phương pháp khuyến khích mô hình cẩn trọng khi cập nhật các tham số của nó khi có dữ liệu mới, giúp bảo toàn kiến thức từ các nhiệm vụ trước đó và ngăn chặn hiện tượng quên đột ngột. Chúng được thể hiện thông qua các kỹ thuật như: Elastic Weight Consolidation (EWC), Synaptic Intelligence (SI).

3.1.3.2 Một số điểm quan trọng về Regularization

Khi áp dụng các phương pháp điều chỉnh qui luật trong máy học, có một số điểm quan trọng cần lưu ý:

Lựa chọn tham số điều chỉnh: Các phương pháp điều chỉnh qui luật thường có các tham số (ví dụ: hệ số điều chỉnh cho L1, L2, tỉ lệ dropout, etc.) cần phải được điều chỉnh một cách cẩn thận. Quá nhiều hoặc quá ít điều chỉnh có thể ảnh hưởng đến hiệu suất của mô hình.

Ảnh hưởng đến khả năng học: Điều chỉnh qui luật có thể giảm khả năng học của mô hình nếu áp dụng quá mức. Việc quá mức phạt hoặc loại bỏ quá nhiều thông tin cũng có thể làm giảm hiệu suất dự đoán.

Phù hợp với dữ liệu: Các phương pháp điều chỉnh qui luật không phải lúc nào cũng phù hợp với mọi loại dữ liệu. Một phương pháp có thể hoạt động tốt với một loại dữ liệu nhất định nhưng không hiệu quả với loại dữ liệu khác.

Tương tác với kiến trúc mô hình: Một số phương pháp điều chỉnh qui luật có thể tương tác khác nhau với các kiến trúc mô hình. Ví dụ, dropout có thể không phù hợp với một số kiến trúc cụ thể hoặc L1/L2 regularization có thể ảnh hưởng khác nhau đối với mô hình.

Thời gian huấn luyện: Một số phương pháp điều chỉnh qui luật có thể làm tăng thời gian huấn luyện của mô hình do việc tính toán thêm các điều khoản điều chỉnh.

Cân nhắc giữa bias và variance: Điều chỉnh qui luật thường liên quan đến việc cân nhắc giữa việc giảm variance (ngăn chặn overfitting) và việc tăng bias (giảm

khả năng mô hình phù hợp với dữ liệu). Cần phải tìm ra sự cân bằng phù hợp để đạt được hiệu suất tốt nhất.

Kiểm tra và đánh giá: Cần thực hiện kiểm tra kỹ lưỡng và đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra hoặc validation để đảm bảo rằng phương pháp điều chỉnh qui luật không làm giảm hiệu suất trên dữ liệu mới.

3.1.4 Transfer Learning

3.1.4.1 Định nghĩa

Chuyển giao kiến thức (Transfer Learning) là một phương pháp trong học máy, trong đó kiến thức đã học từ một tác vụ hoặc một bài toán được chuyển giao và áp dụng vào một tác vụ hoặc bài toán mới khác. Thay vì huấn luyện một mô hình từ đầu với dữ liệu mới hoàn toàn, chuyển giao kiến thức cho phép mô hình sử dụng những kiến thức đã học từ tác vụ trước để cải thiện hiệu suất trên tác vụ mới.

3.1.4.2 Một số điểm quan trọng về Transfer Learning

Huấn luyện từ các mô hình trước đó: Chuyển giao kiến thức thường bắt đầu từ việc sử dụng mô hình đã được huấn luyện trên tác vụ gốc và sau đó tinh chỉnh nó trên tác vụ mới với ít dữ liệu hơn.

Cách thức chuyển giao: Có hai cách thức chính để chuyển giao kiến thức:

Fine-tuning: Đó là quá trình tiếp tục huấn luyện mô hình từ các lớp trên cùng của mô hình cũ (thường được gọi là các lớp mạng nơ-ron hoặc lớp cuối cùng) trên dữ liệu mới.

Feature extraction: Sử dụng các đặc trưng được học từ mô hình cũ như là đầu vào cho một mô hình mới.

Lợi ích của chuyển giao kiến thức: Chuyển giao kiến thức giúp giảm thiểu nhu cầu về dữ liệu huấn luyện mới, giảm thời gian và chi phí huấn luyện, cải thiện khả năng tổng quát hóa của mô hình trên tác vụ mới, đặc biệt khi tác vụ mới có ít dữ liệu huấn luyện.

Lựa chọn mô hình gốc: Sự thành công của chuyển giao kiến thức phụ thuộc lớn vào sự tương đồng giữa tác vụ gốc và tác vụ mới, cũng như mô hình gốc được chọn để chuyển giao kiến thức.

Ứng dụng rộng rãi: Chuyển giao kiến thức được sử dụng rộng rãi trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, thị giác máy tính, và các bài toán nhận diện hình ảnh.

3.1.4.3 Hạn chế

Mặc dù chuyển giao kiến thức (Transfer Learning) có nhiều ưu điểm và áp dụng rộng rãi, nhưng cũng có một số hạn chế cần xem xét:

Dữ liệu không phù hợp: Chuyển giao kiến thức yêu cầu dữ liệu từ tác vụ gốc và tác vụ mới phải có một mức độ tương đồng đủ cao để chuyển giao kiến thức có hiệu quả. Nếu dữ liệu không tương đồng đủ hoặc quá khác nhau, hiệu suất của việc chuyển giao kiến thức có thể giảm.

Overfitting hoặc Underfitting: Nếu không điều chỉnh các tham số huấn luyện một cách thích hợp, chuyển giao kiến thức có thể dẫn đến overfitting hoặc underfitting trên tác vụ mới. Điều này có thể xảy ra khi mô hình gốc quá phức tạp hoặc quá đơn giản.

Phụ thuộc vào kiến trúc mô hình gốc: Hiệu suất của chuyển giao kiến thức có thể bị hạn chế nếu kiến trúc mô hình gốc không phù hợp hoặc không hiệu quả cho tác vụ mới. Đôi khi, mô hình gốc có thể không chứa đủ thông tin cần thiết cho tác vụ mới.

Khó khăn trong việc tinh chỉnh tham số: Điều chỉnh các tham số khi chuyển giao kiến thức có thể trở nên phức tạp hơn so với việc huấn luyện mô hình từ đầu, đặc biệt nếu không có sẵn nhiều dữ liệu cho tác vụ mới.

Chi phí tính toán: Nếu mô hình gốc rất lớn hoặc phức tạp, việc chuyển giao kiến thức có thể đòi hỏi nhiều tài nguyên tính toán, đặc biệt khi cần phải fine-tuning trên tác vụ mới.

3.2 Test Production

Test production (kiểm thử sản phẩm) là quá trình kiểm tra và đánh giá hiệu suất của một sản phẩm hoặc ứng dụng trước khi nó được triển khai hoặc phát hành chính thức cho người dùng cuối. Trong lĩnh vực học máy và phát triển phần mềm, test production thường xảy ra trong giai đoạn cuối của quy trình phát triển sản phẩm.

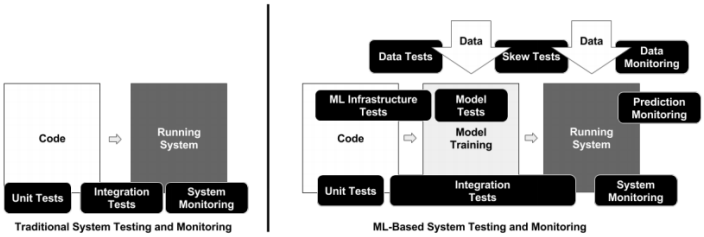
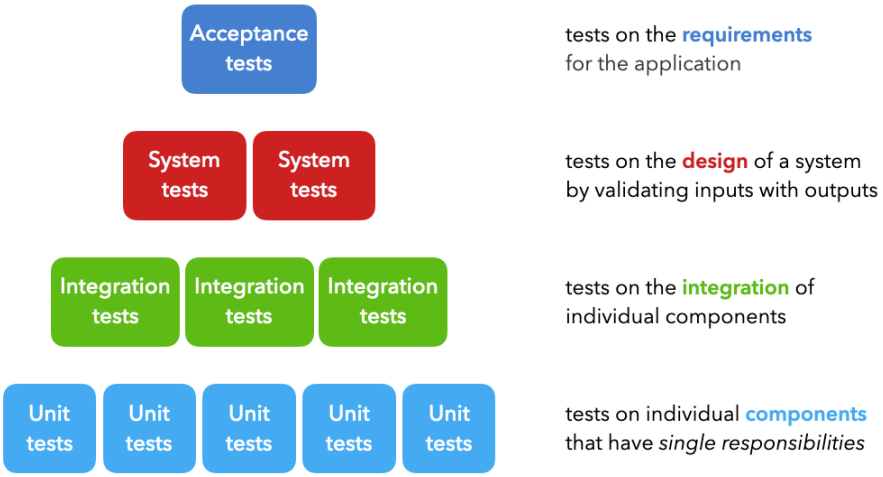


Figure 1. **ML Systems Require Extensive Testing and Monitoring.** The key consideration is that unlike a manually coded system (left), ML-based system behavior is not easily specified in advance. This behavior depends on dynamic qualities of the data, and on various model configuration choices.

Hình 3.2 Minh họa về quá trình Testing



Hình 3.3 Minh họa về các loại test trong machine learning

3.2.1 Integration Testing

Kiểm thử tích hợp là một kỹ thuật kiểm thử phần mềm kiểm tra tích hợp giữa các module hoặc thành phần khác nhau của một ứng dụng phần mềm. Trong Machine Learning (ML), kiểm thử tích hợp liên quan đến việc kiểm tra tương tác giữa các thành phần khác nhau của một ML pipeline, như tiền xử lý dữ liệu, huấn luyện mô hình và đánh giá mô hình.

Các kiểm thử tích hợp trong ML giúp xác nhận rằng các thành phần khác nhau của ML pipeline hoạt động cùng nhau một cách chính xác. Ví dụ, một kiểm thử tích hợp có thể kiểm tra rằng dữ liệu đã được tiền xử lý đúng cách trước khi được đưa vào mô hình, hoặc rằng các chỉ số đánh giá được tính toán đúng sau khi mô hình được huấn luyện.

Kiểm thử tích hợp là quan trọng trong ML vì nó giúp đảm bảo rằng toàn bộ ML pipeline hoạt động như mong đợi. Nó cũng giúp xác định các vấn đề tiềm ẩn hoặc chỗ trở ngại trong pipeline mà có thể không rõ ràng khi kiểm thử từng thành phần đơn lẻ một cách riêng biệt.

Kiểm thử tích hợp thường phức tạp hơn so với kiểm thử đơn vị vì nó liên quan đến việc kiểm tra tương tác giữa các thành phần khác nhau. Có thể cần sử dụng dữ liệu mô phỏng hoặc dữ liệu thực tế để kiểm thử pipeline dưới điều kiện thực tế. Tổng cộng, kiểm thử tích hợp là một phần quan trọng trong quá trình phát triển ML để đảm bảo rằng mô hình cuối cùng đáng tin cậy và hoạt động như mong đợi trong sản xuất.

3.2.2 Unit Testing

Kiểm thử đơn vị là một thực hành kỹ thuật phần mềm liên quan đến việc kiểm thử từng đơn vị hoặc thành phần riêng lẻ của một ứng dụng phần mềm độc lập để đảm bảo chúng hoạt động như mong đợi. Trong Machine Learning (ML), kiểm thử đơn vị được sử dụng để xác nhận từng thành phần cá nhân của một mô hình ML, chẳng hạn như tiền xử lý dữ liệu, kiến trúc mô hình và thuật toán huấn luyện.

Kiểm thử đơn vị trong ML rất quan trọng để đảm bảo từng thành phần của pipeline hoạt động như dự kiến. Ví dụ, kiểm thử đơn vị về tiền xử lý dữ liệu có thể kiểm tra xem dữ liệu bị thiếu được xử lý đúng cách hay không, hoặc dữ liệu được chuẩn hóa một cách thích hợp. Kiểm thử đơn vị về kiến trúc mô hình có thể xác minh rằng mô hình được xây dựng chính xác và các định dạng dữ liệu vào và ra đều như mong đợi.

Kiểm thử đơn vị giúp phát hiện lỗi sớm trong quá trình phát triển và giảm nguy cơ triển khai một mô hình có lỗi. Chúng cũng có thể ngăn chặn việc quay trở lại bằng cách đảm bảo rằng các thay đổi trong một phần của mô hình không ảnh hưởng đến

các thành phần khác. Tổng thể, kiểm thử đơn vị là một thực hành quan trọng trong phát triển ML để đảm bảo mô hình chất lượng cao và đáng tin cậy.

3.2.3 Invariance tests

Kiểm thử không biến đổi là một loại kiểm thử trong Machine Learning (ML) kiểm tra xem một mô hình có không biến đổi với các biến đổi hoặc thay đổi cụ thể trong dữ liệu đầu vào hay không. Nói cách khác, một mô hình không biến đổi sẽ tạo ra cùng kết quả đầu ra cho các đầu vào tương tự nhau nhưng đã trải qua một số biến đổi hoặc thay đổi. Thuộc tính này quan trọng trong ML vì nó đảm bảo rằng dự đoán của mô hình là mạnh mẽ đối với các biến đổi trong dữ liệu đầu vào.

Ví dụ, một mô hình phân loại hình ảnh nên không biến đổi với các thay đổi về điều kiện ánh sáng, vị trí của đối tượng trong hình ảnh, hoặc hướng của đối tượng. Một kiểm thử không biến đổi sẽ liên quan đến việc đánh giá hiệu suất của mô hình trên một tập dữ liệu mà các biến đổi này được áp dụng vào các hình ảnh đầu vào. Nếu mô hình không biến đổi với những biến đổi này, nó sẽ hoạt động tốt trên các hình ảnh đã được biến đổi như nó đã làm trên các hình ảnh gốc.

Kiểm thử không biến đổi quan trọng vì nó đảm bảo rằng hiệu suất của mô hình không bị ảnh hưởng bởi các biến đổi cụ thể trong dữ liệu đầu vào. Nó có thể giúp xác định các điểm yếu có thể có trong mô hình và thông tin cho quyết định về tăng cường dữ liệu hoặc các kỹ thuật khác có thể cải thiện tính mạnh mẽ của mô hình.

Tổng quát, kiểm thử không biến đổi là một khía cạnh quan trọng của việc kiểm thử và đánh giá ML, đảm bảo rằng dự đoán của mô hình là mạnh mẽ và đáng tin cậy trong nhiều điều kiện.

3.2.4 Directional Expectation Tests

Kiểm định hướng đối chiếu là một phương pháp kiểm định thống kê được sử dụng trong máy học để xác định xem dự đoán của mô hình có nhất quán với một kỳ vọng trước đó hoặc giả thuyết không. Loại kiểm định này đặc biệt hữu ích trong các tình huống mà kết quả dự kiến của mô hình đã biết trước.

Kiểm định hướng đối chiếu liên quan đến việc tính toán một thống kê kiểm định dựa trên sự khác biệt giữa các giá trị quan sát và giá trị dự kiến, và so sánh nó với một giá trị quan trọng dựa trên một mức ý nghĩa được xác định trước. Nếu thống kê kiểm định nằm ngoài giá trị quan trọng, giả thuyết không hợp lệ được bác bỏ, và kết luận rằng dự đoán của mô hình khác biệt so với kết quả dự kiến.

Bằng cách kiểm định này, chúng ta có thể xác định liệu dự đoán của mô hình có phù hợp với kỳ vọng của chúng ta không.

Kiểm định hướng đối chiếu là một kỹ thuật quan trọng trong đánh giá ML vì nó cung cấp một khung thống kê để xác định xem dự đoán của mô hình có nhất quán với kiến thức trước đó hoặc kỳ vọng không. Nó giúp đảm bảo rằng hiệu suất của mô hình tương ứng với kết quả mong muốn và có thể hỗ trợ quyết định về lựa chọn mô hình hoặc điều chỉnh tham số.

3.2.5 Data verification Tests

Data verification Tests (Kiểm thử xác minh dữ liệu) là một loại kiểm thử trong Machine Learning tập trung vào đảm bảo chất lượng và tính toàn vẹn của dữ liệu đầu vào được sử dụng để huấn luyện hoặc đánh giá mô hình, cũng như dữ liệu đầu ra được tạo ra bởi quá trình huấn luyện và suy luận của mô hình. Mục tiêu của kiểm thử xác minh dữ liệu là xác định và giải quyết các vấn đề tiềm ẩn với dữ liệu có thể ảnh hưởng đến độ chính xác hoặc đáng tin cậy của các dự đoán của mô hình.

Kiểm thử xác minh dữ liệu rất quan trọng trong Machine Learning vì chất lượng và tính toàn vẹn của dữ liệu được sử dụng để huấn luyện hoặc đánh giá mô hình có thể ảnh hưởng đáng kể đến hiệu suất và độ chính xác của nó. Bằng việc tiến hành kiểm thử xác minh dữ liệu, có thể xác định và giải quyết các vấn đề tiềm ẩn với dữ liệu trước khi chúng ảnh hưởng đến các dự đoán của mô hình, dẫn đến kết quả đáng tin cậy và chính xác hơn.

3.2.6 System Tests

Kiểm thử hệ thống trong machine learning là việc đánh giá toàn bộ hệ thống hoặc quy trình ML để đảm bảo rằng tất cả các thành phần tích hợp hoạt động một

cách hòa hợp và đáp ứng các yêu cầu được chỉ định. Nó bao gồm việc kiểm tra chức năng và hiệu suất của toàn bộ hệ thống ML thay vì các thành phần cá nhân. Giai đoạn kiểm thử này đánh giá sự tương tác giữa các mô-đun khác nhau (như tiền xử lý dữ liệu, huấn luyện mô hình và suy luận) và xác định xem hệ thống có đáp ứng được mục tiêu và các thông số kỹ thuật đã được xác định không. Mục tiêu là kiểm chứng hành vi, chức năng và hiệu suất của hệ thống trong điều kiện thực tế hoặc mô phỏng trước khi triển khai.

3.2.7 Acceptance Tests

Acceptance Tests trong machine learning (ML) thường được thực hiện để đảm bảo rằng mô hình hoặc hệ thống ML đáp ứng các yêu cầu và tiêu chí chấp nhận từ phía người dùng hoặc khách hàng. Đây là giai đoạn kiểm thử cuối cùng trước khi triển khai, nơi mà mô hình ML được đánh giá dựa trên các tiêu chí đã được xác định trước, bao gồm các yêu cầu chức năng và phi chức năng.

Acceptance Tests trong ML thường tập trung vào việc đánh giá hiệu suất của mô hình trên dữ liệu thực tế hoặc dữ liệu giả mô phỏng để xác định xem mô hình có hoạt động như mong đợi và đáp ứng được nhu cầu sử dụng hay không.

3.2.8 Regression Tests

Regression tests trong machine learning là các bài kiểm thử được thực hiện để đảm bảo rằng sự thay đổi hoặc cập nhật của mô hình học máy không làm ảnh hưởng đến các kết quả hoặc hiệu suất đã được xác định trước. Cụ thể, kiểm thử này đánh giá xem các thay đổi trong mô hình, bao gồm cập nhật dữ liệu, cải thiện hoặc điều chỉnh mô hình, không gây ra những thay đổi không mong muốn hoặc không lường trước trong kết quả dự đoán. Regression tests giúp đảm bảo tính nhất quán và độ tin cậy của mô hình sau mỗi cập nhật, đồng thời tránh được sự biến đổi không mong muốn trong kết quả hoặc hiệu suất của mô hình.

TÀI LIỆU THAM KHẢO

[1] [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.

[2] [Online]. Available: https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/#h-what-is-the-adam-optimizer.

[3] [Online]. Available: https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m.

[4] [Online]. Available: https://builtin.com/data-science/gradient-descent.

[5] [Online]. Available: https://www.javatpoint.com/gradient-descent-in-machine-learning.

[6] [Online]. Available: https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/.

[7] [Online]. Available: https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1.

[8] [Online]. Available: https://www.codingninjas.com/studio/library/rmsprop.

[9] [Online]. Available: https://machinelearning.cards/p/rmsprop.

[10] [Online]. Available: https://deepchecks.com/glossary/rmsprop/.

[11] [Online]. Available: https://spotintelligence.com/2023/10/03/continual-learning/#What_is_continual_learning.

[12] [Online]. Available: https://www.datacamp.com/blog/what-is-continuous-learning.

[13] [Online]. Available: https://www.nightfall.ai/ai-security-101/catastrophic-forgetting.

[14] [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/.

[15] [Online]. Available: https://www.godeltech.com/how-to-automate-the-testing-process-for-machine-learning-systems/.

[16] [Online]. Available: https://madewithml.com/courses/mlops/testing/.

[17] [Online]. Available: https://madewithml.com/courses/mlops/testing/#types-of-tests.