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

HOÀNG ĐỨC MINH – 52100912

NGHIÊN CỨU VỀ PHƯƠNG PHÁP OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION

DỰ ÁN CUỐI KỲ MÔN MACHINE LEARNING

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

HOÀNG ĐỨC MINH – 52100912

NGHIÊN CỨU VỀ PHƯƠNG PHÁP OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION

DỰ ÁN CUỐI KỲ MÔN MACHINE LEARNING

Người hướng dẫn
Thầy Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Nhóm em xin gửi lời cảm ơn đến thầy Lê Anh Cường - Giảng viên khoa Công Nghệ Thông Tin - Trường Đại học Tôn Đức Thắng đã giúp đỡ nhóm em trong việc hoàn thành bài dự án cuối kỳ. Thầy đã hết sức tận tình trong việc giảng dạy và truyền đạt kiến thức trong suốt quá trình học tập. Tạo động lực thúc đẩy chúng em có thể hoàn thành bài báo cáo một cách suôn sẻ và tốt nhất.

Trong suốt quá trình làm bài, có một số thiếu sót nhỏ mà chúng em không phát hiện ra mong nhận được sự đóng góp của Thầy để giúp nhóm em hoàn thiện hơn.

Xin chân thành cảm ơn Thầy!

TP. Hồ Chí Minh, ngày tháng năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Hoàng Đức Minh

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

Em xin cam đoan đây là công trình nghiên cứu của riêng em và được sự hướng dẫn khoa học của Thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra trong báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu có bất kỳ sự gian lận nào em 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 em gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Hoàng Đức Minh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

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

1. Khái niệm về Optimizer

- Optimizer (hay còn được gọi là bộ tối ưu hóa) là một thành phần quan trọng của quá trình huấn luyện mô hình máy học. Nhiệm vụ chính của optimizer là tối ưu hóa các tham số của mô hình, chẳng hạn như trọng số (weights) và độ lệch (bias), để mô hình đạt được hiệu suất tối ưu trên dữ liệu huấn luyện. Quá trình này được thực hiện dựa trên việc giảm thiểu hàm mất mát (loss function).
- Mục tiêu cuối cùng của quá trình tối ưu hóa là đạt được một bộ tham số mô hình mà khi áp dụng vào dữ liệu mới, mô hình sẽ có khả năng dự đoán chính xác. Để đạt được điều này, optimizer thường sử dụng các phương pháp tối ưu hóa, chẳng hạn như gradient descent, để điều chỉnh giá trị của các tham số sao cho hàm mất mát giảm dần.
- Các thuật toán tối ưu hóa khác nhau có thể có ảnh hưởng lớn đến tốc độ và độ chính xác của quá trình huấn luyện. Việc lựa chọn một optimizer phù hợp có thể phụ thuộc vào đặc tính của dữ liệu và cấu trúc của mô hình.
- Bộ tối ưu hóa (optimizer) trong học máy hoạt động nhằm điều chỉnh các tham số của mô hình để giảm thiểu giá trị của hàm loss. Mục tiêu cuối cùng là tìm ra bộ tham số mô hình sao cho mô hình có khả năng dự đoán chính xác trên dữ liệu mới. Dưới dây là một số optimizer phổ biến trong học máy:
    - Gradient Descent: sử dụng đạo hàm của hàm loss với các tham số mô hình để xác định hướng và mức độ điều chỉnh của chúng.
    - Stochastic Gradient Descent: chỉ sử dụng một mẫu ngẫu nhiên trong mỗi lần cập nhật thay vì phải cập nhật tham số dựa trên toàn bộ dữ liệu huấn luyện.

- Adam (Adaptive Moment Estimation): là một optimizer kết hợp với các ưu điểm của AdaGrad và RMSProp. Sử dụng thông tin lịch sử của Gradient để điều chỉnh learning rate cho từng tham số.
- RMSProp (Root Mean Square Propagation): cũng như Adam, RMSProp vừa giảm learning rate của tham số có gradient lớn vừa tăng learning rate cho các tham số có gradient nhỏ.
- Learning Rate Schedules: một số optimizer dùng Learning Rate Schedules để điều chỉnh learning rate theo thời gian và số lần cập nhật tham số.

➢ Mỗi thuật toán có những cách tiếp cận và đặc điểm riêng, nhưng chủ yếu đều dựa trên việc điều chỉnh các tham số mô hình dựa trên gradient của tham số mô hình và tỉ lệ learning rate để đạt được tính tối ưu.

2. Các phương pháp Optimizer:

2.1. Gradient Descent

■ Là một trong những phương pháp tối ưu hóa phổ biến nhất. Ý tưởng chính là sử dụng đạo hàm của hàm loss đối với các tham số mô hình để xác định hướng và mức độ điều chỉnh của chúng. Gradient Descent cố gắng giảm giá trị của hàm mất mát bằng cách đi theo hướng ngược với đạo hàm của nó.

■ Nguyên lý hoạt động:

- Bước 1: Khởi tạo tham số mô hình: weights, bias,...
- Bước 2: Tính đạo hàm của hàm loss đối với từng tham số để xác định hướng và mức độ điều chỉnh.
- Bước 3: Cập nhật tham số bằng cách di chuyển ngược với đạo hàm hàm loss để giảm thiểu giá trị mất mát.

■ Công thức tính:

$$\theta_{i+1} = \theta_i - \gamma v_{t-1} . \eta \nabla J(\theta_t)$$

Trong đó,

- θ_i là vector trọng số tại bước thứ *i*
- $\gamma v_{t-1} . \eta \nabla J(\theta)$ là vận tốc di chuyển ngược với đạo hàm hàm loss.
- η là tỉ lệ học (learning rate)

○ Bước 4: Lặp lại các bước cho đến khi đạt điều kiện dừng.

➢ Các phương pháp optimizer như Adam, RMSProp, hay SGD được sử dụng để cải thiện và điều chỉnh quá trình Gradient Descent. Chúng thường kết hợp thông tin về lịch sử của gradient để điều chỉnh tốc độ học và giảm thiểu nhược điểm của Gradient Descent truyền thống, như việc chọn learning rate cố định.

■ Một số bài toán phù hợp:

- Hồi quy tuyến tính: Bài toán dự đoán một biến liên tục dựa trên các biến đầu vào. Gradient Descent thường được sử dụng để tối ưu hóa hàm mất mát.
- Phân loại và hồi quy
- Phân loại Ảnh và đối tượng: Bài toán nhận dạng và phân loại ảnh, cũng như xác định vị trí của đối tượng trong ảnh, thường sử dụng các thuật toán tối ưu hóa để điều chỉnh trọng số của mô hình.
- Dự đoán thời gian: Khi dự đoán giá trị trong chuỗi thời gian, Gradient Descent có thể được sử dụng để tối ưu hóa các mô hình dự đoán.

2.2. Stochastic Gradient Descent

- SGD là một trong những phương pháp tối ưu hóa phổ biến tương tự như Gradient Descent, nhưng thay vì phải cập nhật tham số dựa trên toàn bộ dữ liệu huấn luyện, SGD chỉ sử dụng một mẫu ngẫu nhiên trong mỗi lần cập nhật.
- Nguyên lý hoạt động:
  - Bước 1: Khởi tạo ngẫu nhiên các trọng số, độ lệch của mô hình
  - Bước 2: Dữ liệu huấn luyện được chia thành các mẫu ngẫu nhiên. Với mỗi mẫu dữ liệu, thuật toán tính gradient của hàm loss theo trọng số của mô hình
  - Bước 3: Cập nhật tham số
- Công thức tính:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla J(\theta_t, x_t, y_t)$$

Trong đó,

- θ_i là vector trọng số tại bước thứ *i*
- $\nabla J(\theta_t, x_t, y_t)$ là gradient của hàm loss đối với tham số dựa trên (input, label) là (x, y).
- η là tỉ lệ học (learning rate)

  - Bước 4: Lặp lại quá trình cho đến khi đạt điều kiện dừng.

➢ Việc chia dữ liệu thành các mẫu để phân tích giống phương pháp chia để trị trong các giải thuật phổ biến. Việc này giúp nâng cao tốc độ học và giảm độ phức tạp của thuật toán một cách đáng kể. Đồng thời cải thiện tốt thời gian và không gian.

- Một số bài toán phù hợp:

- Phân loại và hồi quy: SGD thường được sử dụng hiệu quả trong bài toán phân loại và hồi quy,
- Không Gian Trạng Thái Lớn: Khi không gian trạng thái của mô hình lớn và không khả dụng để lưu trữ trong bộ nhớ, SGD trở nên hữu ích vì nó chia nhỏ dữ liệu để xử lý.
- Học máy trực tuyến: Trong các tình huống cần phải xử lý dữ liệu đến từ nguồn nguồn đến một cách liên tục, SGD thích hợp trong học máy trực tuyến.
- Dữ Liệu Nhiễu: Khi dữ liệu có nhiễu, SGD có thể giúp tránh các vấn đề về quá mức điều chỉnh và giúp mô hình tự động thích ứng với các biến động ngẫu nhiên.
- Dữ liệu Lớn: SGD thích hợp cho các tập dữ liệu lớn, vì nó chỉ cần sử dụng phương pháp chia nhỏ mỗi lần cập nhật, giảm bớt tải lớn về tài nguyên tính toán so với Gradient Descent thông thường.

2.3. RMSProp (Root Mean Square Propagation)

- RMSprop (Root Mean Square Propagation) là một biến thể của Gradient Descent, vì vậy nó cũng là một phương pháp optimizer phổ biến trong mô hình học máy. Nó được thiết kế để cải thiện hiệu suất tối ưu hóa bằng cách điều chỉnh tỉ lệ học (learning rate) dựa trên thông tin độ lớn của gradient.
- Nguyên lý hoạt động:
  - Bước 1: Khởi tạo tham số ban đầu bao gồm trọng số và độ lệch bias.
  - Bước 2: Khởi tạo biến đặc trưng $E[g^2]$ nhằm theo dõi giá trị trung bình bình phương của các gradient, gán bằng 0.
  - Bước 3: Lặp qua từng mẫu dữ liệu sau đó tính gradient của hàm loss theo trọng số θ, ký hiệu là g_t

- Bước 4: Cập nhật biến đặc trưng bằng công thức:

$$E[g^2]_t = \beta \cdot E[g^2]_{t-1} + (1 - \beta) \cdot (g_t)^2,\ g_t = \nabla J(\theta_t)$$

- Bước 5: Cập nhật trọng số θ bằng cách sử dụng tỷ lệ học (η) chia cho căn bậc hai của trung bình bình phương gradient đã tính:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t}+\epsilon} \cdot g_t$$

- Bước 6: Lặp các bước trên cho mỗi epoch hoặc mini-batch cho đến khi đạt được tiêu chí dừng hoặc số lượng epoch đã đủ.

➢ Phương pháp RMSProp giúp kiểm soát tốc độ học tự điều chỉnh theo giá trị trung bình của bình phương các gradient trước đó. Điều này giúp nhanh chóng thích ứng với mô hình và giảm nguy cơ tốc độ học quá nhanh hoặc quá chậm trong quá trình huấn luyện. RMSProp thường là lựa chọn hiệu quả trong nhiều loại bài toán học máy và học sâu.

■ Một số bài toán phù hợp:

- Phân loại và Hồi quy: RMSprop phù hợp cho các bài toán phân loại và hồi quy, đặc biệt là có sự biến động trong dữ liệu đầu vào.
- Mô Hình Học Sâu (Deep Learning)
- Dữ Liệu Nhiễu: Khi dữ liệu có nhiễu hoặc biến động mạnh, RMSprop có thể giúp ổn định quá trình tối ưu hóa bằng cách điều chỉ tỷ lệ học dựa trên độ lớn của gradient..
- Học máy trực tuyến: RMSprop có thể phù hợp trong quá trình học online, nơi mô hình cần được cập nhật khi có dữ liệu mới được thêm vào.
- Dữ Liệu Lớn: RMSprop thường hoạt động tốt trên các tập dữ liệu lớn.

2.4. Adam (Adaptive Moment Estimation)

- Adam là một phương pháp optimizer phổ biến trong machine learning, được thiết kế để cải thiện quá trình tối ưu hóa bằng cách kết hợp cả các ưu điểm của phương pháp AdaGrad và RMSProp. Sử dụng thông tin lịch sử của Gradient để điều chỉnh learning rate cho từng tham số.

- Nguyên lý hoạt động:
  - Bước 1: Khởi tạo các tham số ban đầu bao gồm trọng số θ, các moments m_0 và v_0 ban đầu là 0. Dùng để theo dõi Gradient và Gradient bình phương trung bình.
  - Bước 2: Tính gradient của hàm loss theo trọng số θ, ký hiệu là gr_t
  - Bước 3: Cập nhật các biến đặc trưng moment m và v:

$$gr_t = \nabla J(\theta_t)$$

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . gr_t$$

$$v_t = \beta_2 . v_{t-1} + (1 - \beta_2) . (gr_t)^2$$

Với β_1 và β_2 là các hệ số giảm dần, thường được đặt giá trị mặc định là 0.9 và 0.999

  - Bước 4: Tính ước lượng chính xác của các moments m_t và v_t để không bị mất mát bias khi bắt đầu.

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

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

- Bước 5: Cập nhật trọng số θ bằng cách sử dụng các ước lượng chính xác của m_t và v_t, cũng như tỷ lệ học (η) và một giá trị (ϵ) đủ nhỏ để tránh chia cho 0:

$$\theta_{t+1} = \theta_t - \eta . \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

- Bước 6: Lặp lại các bước trên cho mỗi epoch hoặc mẫu cho đến khi đạt được mục tiêu dừng hoặc số lượng epoch đã đủ

➢ Adam tự điều chỉnh tốc độ học của từng tham số dựa trên lịch sử của gradient, giúp quá trình hội tụ nhanh hơn và tránh những biến động đáng kể trong việc tối ưu hóa. Phương pháp này thường được sử dụng hiệu quả trong huấn luyện mô hình học sâu. Nhờ vào các tính chất này, Adam thường đạt được hiệu suất tốt trên nhiều bài toán và kiến trúc mô hình.

■ Một số bài toán phù hợp:

- Phân loại và Hồi quy: Adam thường được sử dụng hiệu quả cao, đặc biệt là khi có độ phức tạp cao và nhiều tham số.
- Mô Hình Học Sâu (Deep Learning)
- Dữ Liệu Nhiễu: Adam có thể ổn định quá trình học trên dữ liệu nhiễu, đặc biệt là khi dữ liệu chứa đựng độ nhiễu và biến động ngẫu nhiên.
- Học máy trực tuyến: Trong các tình huống cần phải xử lý dữ liệu đến từ nguồn nguồn đến một cách liên tục, Adam thích hợp trong học máy trực tuyến.

- Dữ Liệu Lớn: Adam thường hoạt động tốt trên các tập dữ liệu lớn, nơi sự adaptive của nó trong việc điều chỉnh tỷ lệ học có thể giúp tối ưu hóa mô hình hiệu quả.

CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

1. Continual Learning

★ Continual Learning (CL) là một lĩnh vực quan trọng trong machine learning, nơi mà mô hình phải liên tục học và thích ứng với dữ liệu mới mà không quên đi kiến thức đã học từ trước. Điều này đặt ra một thách thức lớn vì các mô hình máy học thường có xu hướng quên đi kiến thức cũ khi họ được đào tạo lại trên dữ liệu mới.

1.1. So sánh giữa Continual Learning và mô hình Machine Learning cơ bản

- Trong lĩnh vực học máy, phần lớn các mô hình thường được huấn luyện trên một tập dữ liệu cố định và sau đó không thể học thêm sau khi quá trình huấn luyện ban đầu kết thúc. Tuy nhiên, trong thực tế, dữ liệu thường xuất hiện dưới dạng luồng không ngừng, đòi hỏi khả năng của mô hình để liên tục thích ứng và học từ các tác vụ mới mà không gây mất mát đáng kể về kiến thức đã học từ các tác vụ trước đó.
- Khả năng này gọi là "học liên tục" (Continual Learning), một khía cạnh quan trọng của học máy trong môi trường động. Trong học liên tục, mô hình không chỉ cần giữ vững thông tin từ dữ liệu mới mà còn phải duy trì và cập nhật kiến thức đã học từ dữ liệu trước đó.
- So với các mô hình machine learning cơ bản, Continual Learning có các điểm khác biệt sau:
    - ➢ *Đối tượng:* Machine learning cơ bản thường áp dụng cho quá trình huấn luyện mô hình trên tập dữ liệu cố định, sau đó đưa ra dự đoán trên cơ sở dữ liệu mới. Trong khi, Continual learning là một phương pháp Machine learning nhưng mô hình này không chỉ học

từ một tập dữ liệu ban đầu, mà nó có khả năng học thêm dữ liệu mới theo thời gian mà không gây ra sự mất mát kiến thức cũ..

- ➢ *Phương pháp học*: đối với Machine Learning cơ bản thì việc học chỉ diễn ra một lần duy nhất và nó không có khả năng học thêm kiến thức mới sau khi đã được huấn luyện ban đầu. Còn đối với Continual learning có thể học liên tục và có tính thích ứng với kiến thức mới và cập nhật, vượt qua thách thức của việc quên đi kiến thức đã được học. Chính vì khả năng này mà Continual Learning có thể thích ứng với dữ liệu có tính liên tục, nó rất phù hợp với khả năng thích nghi và cập nhật dữ liệu mới. đối mặt với hai thách thức chính là quên đi (catastrophic forgetting) và sự xáo trộn kiến thức (knowledge transfer). Trong các mô hình truyền thống, không có cơ chế tự động để xử lý hai thách thức này. Trong khi đó, continual learning tự động giải quyết hai vấn đề này bằng cách duy trì kiến thức quan trọng từ các tác vụ trước và áp dụng kiến thức đó cho các tác vụ mới.
- ➢ *Ứng dụng*: Các mô hình machine learning cơ bản thường sử dụng phương pháp học giám sát và đòi hỏi một lượng lớn dữ liệu được gán nhãn để huấn luyện. Vì vậy nó có thể xử lý dữ liệu ổn định và không thay đổi nhanh chóng theo thời gian. Trong khi đó, Continual learning có thể sử dụng các phương pháp học không giám sát, học bán giám sát hoặc học tăng cường để học từ các tác vụ mới. Vì vậy nó thích hợp cho các tình huống nơi dữ liệu thường xuất hiện dưới dạng luồng không ngừng hoặc thay đổi, và mô hình cần duy trì khả năng học và dự đoán trên các nhiệm vụ mới.

1.2. Thách thức của Continual Learning

★ Catastrophic Forgetting

- Catastrophic forgetting là hiện tượng mô hình machine learning quên đi hoàn toàn kiến thức đã học từ các dữ liệu cũ khi học dữ liệu mới. Nguyên nhân dẫn tới thách thức này là các tham số của mô hình được điều chỉnh để phù hợp với dữ liệu . khi được cập nhật với dữ liệu mới, các trọng số có thể được điều chỉnh một cách quá mức, làm mất mát thông tin quan trọng.
- Catastrophic forgetting có thể gây ra những vấn đề nghiêm trọng trong Continual learning. Nếu mô hình không thể giữ lại kiến thức quan trọng từ các dữ liệu cũ, hiệu suất của nó trên các dữ liệu đó sẽ giảm đi đáng kể.

★ Interference

- Việc học dữ liệu mới có thể gây ra một số tác động đối lập lên dữ liệu đã được học. Điều này có thể dẫn tới hiệu suất tổng thể bị giảm đi. Nguyên nhân là vì sự thay đổi trong mô hình có thể gây tác động đến các dữ liệu và gây ra hiện tượng đối lập nhau.

★ Shuffle knowledge

- Việc học liên tục các dữ liệu mới và giữ được kiến thức đã học thì quản lý dữ liệu không phải và điều đơn giản. Vì nó phải đối mặt với sự thay đổi dữ liệu một cách đột ngột và liên tục, dẫn đến việc cập nhật cũng phải kịp đáp ứng để duy trì hiệu suất.

★ Resource Management

- Continual Learning phải đối mặt với việc quản lý tài nguyên tính toán khi phải xử lý dữ liệu liên tục. Vì cần phải đảm bảo mô hình không chỉ có thể học liên tục mà còn thích ứng được với việc hạn chế được tài nguyên tính toán dữ liệu.

1.3. Giải pháp cho Continual Learning

➢ Để giải quyết được những thách thức của Continual Learning và đưa mô hình học máy này trở nên phổ biến và hữu dụng hơn, ta có một số giải pháp giúp cho việc hạn chế đi các thách thức của Continual Learning:

- *Rehearsal*: giúp lưu giữ lại một số mẫu dữ liệu cũ để chúng có thể đào tạo lại mô hình khi mô hình bị Catastrophic forgetting. Tuy nhiên nó không hữu dụng cho dữ liệu lớn vì phải tốn không gian lưu trữ.
- *Regularization-based approaches*: Đây là một trong những chiến lược phổ biến trong việc giải quyết thách thức của Continual Learning Các phương pháp này như EWC, SI, PIC giữ lại kiến thức quan trọng từ các tác vụ trước bằng cách giới hạn sự thay đổi của các tham số quan trọng trong quá trình học tác vụ mới. Các tham số đó được xác định dựa trên observed layer và đánh giá mức độ quan trọng đối với các tác vụ trước.
- *Dynamic architecture approaches*: Các phương pháp này tập trung vào việc tạo ra các kiến trúc mô hình linh hoạt có khả năng thay đổi kích thước, số lượng tầng ẩn hoặc số lượng đơn vị. Bằng cách điều chỉnh kiến trúc mô hình theo cách tốt nhất cho mỗi tác vụ, mô hình có khả năng học tác vụ mới mà không gây quên mất kiến thức đã học từ các tác vụ trước.
- *Knowledge distillation*: Phương pháp này nhằm truyền đạt kiến thức từ mô hình phức tạp đã học sang một mô hình đơn giản hơn. Mô hình phức tạp có thể có khả năng đạt được hiệu suất cao trên các tác vụ trước sau đó huấn luyện đơn giản nhằm sao chép từ mô hình phức tạp, ta có thể truyền đạt kiến thức sang sang mô hình đơn giản.
- *Meta-learning*: Phương pháp meta-learning tập trung vào việc học cách học nhanh và hiệu quả từ các tác vụ mới. Mô hình học cách tìm ra các mẫu, quy tắc hoặc cấu trúc chung giữa các tác vụ và áp dụng chúng để học nhanh và hiệu quả trên các tác vụ mới.

2. Test Production

2.1. Test Production là gì?

- Trong quá trình phát triển các mô hình machine learning, Test production là một bước quan trọng trong quy trình kiểm tra và đánh giá hiệu suất của mô hình trước khi triển khai vào môi trường sản xuất.
- Trong quá trình huấn luyện và xác thực dữ liệu, nó được đánh giá trên một tập dữ liệu kiểm tra độc lập. Mục tiêu là đánh giá hiệu suất của mô hình trên tập dữ liệu mà nó chưa từng thấy trước đây, để đảm bảo rằng nó có khả năng tổng quát hóa và dự đoán tốt trên các dữ liệu mới.

2.2. Mục đích của Test Production

- Quá trình kiểm thử đóng vai trò quan trọng để đảm bảo mô hình đã được huấn luyện và triển khai có khả năng hoạt động hiệu quả trong môi trường sản xuất. Giai đoạn này nhằm kiểm tra tính tổng quát hóa của mô hình, đảm bảo khả năng dự đoán chính xác trên dữ liệu mới, và xác minh khả năng xử lý biến thường và tình huống đặc biệt.
- Mục tiêu cuối cùng là đảm bảo hiệu suất và độ tin cậy của mô hình khi triển khai vào môi trường thực tế. Bằng cách thực hiện kiểm thử sản xuất kỹ lưỡng, tổ chức có thể giảm thiểu rủi ro và đảm bảo mô hình làm việc đúng đắn và ổn định trong môi trường sản xuất.
- Dưới đây là những mục đích cụ thể của quy trình:
    - Tính tổng quát hóa: Đảm bảo rằng mô hình có khả năng tổng quát hóa, tức là nó không chỉ hoạt động tốt trên dữ liệu huấn luyện và xác thực, mà còn trên dữ liệu mới mà mô hình chưa từng thấy.
    - Khả năng dự đoán chính xác: Test production giúp đảm bảo rằng mô hình có khả năng tổng quát hóa, đảm bảo rằng mô hình không chỉ hoạt động tốt trên tập dữ liệu huấn luyện và xác thực mà còn trên các dữ liệu mới, không từng được mô hình gặp phải trước đó.

- Xử lý các biến: Đảm bảo rằng mô hình có thể xử lý hiệu quả các biến thường và tình huống đặc biệt trong môi trường sản xuất. Giúp cho mô hình kiểm tra khả năng ổn định và dự đoán đúng trong nhiều tình huống.
- Hiệu suất và đánh giá: Đảm bảo rằng mô hình đáp ứng đúng đắn và có độ tin cậy khi triển khai trong môi trường sản xuất. Đánh giá hiệu suất mô hình trên các mục tiêu đánh giá cụ thể và xác định độ tin cậy của nó

2.3. Các bước của quy trình test production

➢ **Xác định mục tiêu kiểm thử**

- Xác định mục tiêu cụ thể của quá trình kiểm thử, bao gồm cả các yếu tố như tính tổng quát hóa, dự đoán chính xác và xử lý biến.

➢ **Chuẩn bị dữ liệu kiểm thử**

- Cần chuẩn bị một tập dữ liệu kiểm tra độc lập, chưa từng sử dụng trong quá trình huấn luyện. Dữ liệu kiểm tra này nên đại diện cho các điều kiện và tình huống mà mô hình sẽ gặp phải trong môi trường sản xuất.

➢ **Triển khai mô hình trong môi trường kiểm thử**

Cần tạo ra một môi trường tương tự như môi trường thực tế mà mô hình sẽ hoạt động trong đó. Điều này bao gồm cài đặt các thư viện, phần mềm và cấu hình cần thiết để chạy mô hình và triển khai nó.

➢ **Thực hiện kiểm thử**

Mô hình được chạy trên tập dữ liệu kiểm tra và tạo ra các dự đoán. Các đầu ra của mô hình được ghi lại để phân tích và đánh giá hiệu suất sau này.

Bên cạnh đó còn tiến hành kiểm thử các phần nhỏ hơn như function, class trong mã nguồn để đảm bảo kết quả.

Cuối cùng là kiểm tra độ tương thích và khả năng tương tác giữa chúng.

➢ **Đánh giá hiệu suất**

Tiến hành đo lường hiệu suất của mô hình ngay sau khi thực hiện và đánh giá khả năng tổng quát hóa của nó. Kết quả này sẽ cho thấy khả năng dự đoán, độ chính xác và thời gian phản hồi của mô hình trên dữ liệu mới.

➢ **Phân tích kết quả**

Kết quả từ quá trình testing bao gồm các phép đo hiệu suất như độ chính xác, độ phân loại, độ lỗi, đánh giá F1,... Dựa trên các kết quả này, chúng ta có thể đánh giá và so sánh hiệu suất của mô hình và thực hiện các điều chỉnh cần thiết (overfitting,…).

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Machine Learning Cơ bản https://machinelearningcoban.com/
2. Tìm hiểu thêm về Optimizer: https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8