CHƯƠNG 1. CÂU 1

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

1.1 Optimizer

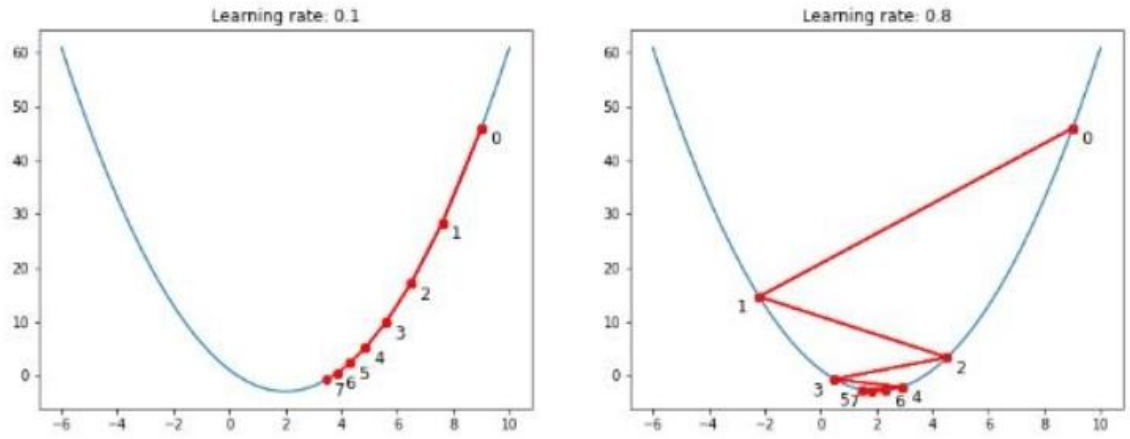
Trong quá trình huấn luyện mô hình học máy, optimizer là một phần quan trọng của quá trình tối ưu hóa. Nó là một thuật toán được sử dụng để điều chỉnh các tham số của mô hình (như trọng số và bias) dựa trên dữ liệu huấn luyện để mục tiêu làm giảm thiểu hàm mất mát (loss function).

Khi một mô hình học máy được huấn luyện, nó cố gắng điều chỉnh các tham số của mình để dự đoán đầu ra chính xác hơn. Optimizer là công cụ giúp mô hình thực hiện việc này bằng cách sử dụng gradient của hàm mất mát đối với các tham số để điều chỉnh chúng theo hướng giảm thiểu mất mát.

Các loại optimizer khác nhau như Gradient Descent, Stochastic Gradient Descent (SGD), Adam, RMSprop, và nhiều thuật toán tối ưu hóa khác có cách tiếp cận khác nhau để cập nhật các tham số của mô hình. Mục tiêu chung của tất cả các optimizer là học mô hình sao cho nó có thể dự đoán chính xác hơn trên dữ liệu mới mà nó chưa từng thấy.

1.2 Gradient descent (GB)

Đây là một trong những phương pháp tối ưu hoá cơ bản và phổ biến trong học máy. Được sử dụng để cập nhật các tham số của mô hình dựa trên gradient của hàm mất mát theo các tham số đó. Mục tiêu là di chuyển từ điểm khởi đầu trên không gian tham số đến điểm tối ưu, thường là điểm tối thiểu của hàm mất mát



Hình 1.1: Gradient Descent Optimizer

1.2.1 Hoạt động

Tính toán Gradient: gradient của hàm mất mát được tính bằng cách lấy đạo hàm của hàm mất mát theo từng tham số trong mô hình

$$\nabla J(\theta_i) = \frac{\partial J}{\partial \theta_i}$$

Trong đó:

- $\nabla J(\theta_i)$ là gradient của hàm mất mát J theo trọng số θ_i
- $\frac{\partial J}{\partial \theta_i}$ là đạo hàm riêng của hàm mất mát J theo trọng số θ_i

Cập nhật tham số: thực hiện cập nhất các tham số của mô hình bằng cách di chuyển ngược chiều của gradient với một tỉ lệ học (learning rate). Bằng cách này, các tham số được điều chỉnh để giảm độ lớn của gradient, mô hình tiến gần đến điểm tối ưu

$$\theta_{t+1} = \theta_t - \eta \nabla \mathrm{J}(\theta_t)$$

Trong đó:

- θ_{t+1} là trọng số mới sau bước thời gian t + 1
- θ_t là trọng số tại bước thời gian t
- η là learning rate, quyết định tốc độ học của thuật toán

1.2.2 Ưu điểm

Đơn giản, dễ triển khai

Hiệu suất tốt

Phổi biến và linh hoạt

1.2.3 Nhược điểm:

Dễ rơi vào điểm cực tiểu cục bộ

Khả năng hội tụ chậm

Phụ thuộc vào learning rate

1.3 Stochastic Gradient descent (SGB)

Đây là thuật toán tối ưu hoá cơ bản theo họ gradient, là 1 biến thể của Gradient Descent. SGD tính toán gradient và cập nhật tham số dựa trên từng điểm dữ liệu trong tập huấn luyện. Điều này làm giảm thiểu độ phức tạp tính toán, nhưng có thể dẫn đến các bước di chuyển không ổn định.

1.3.1 Hoạt động

Thay vì sau mỗi epoch chúng ta sẽ cập nhật trọng số (Weight) 1 lần thì trong mỗi epoch có N điểm dữ liệu chúng ta sẽ cập nhật trọng số N lần. SGD sẽ làm giảm đi tốc độ của 1 epoch. Tuy nhiên nhìn theo 1 hướng khác, SGD sẽ hội tụ rất nhanh chỉ sau vài epoch. Công thức SGD cũng tương tự như GD nhưng thực hiện trên từng điểm dữ liệu.

Chọn mẫu ngẫu nhiên: Đầu tiên, một điểm dữ liệu hoặc một mini-batch nhỏ được chọn ngẫu nhiên từ tập dữ liệu huấn luyện.

Tính gradient: Gradient của hàm mất mát được tính dựa trên điểm dữ liệu hoặc mini-batch vừa chọn.

Cập nhật trọng số: Trọng số của mô hình được cập nhật bằng cách di chuyển theo hướng ngược với gradient với một bước theo learning rate (tốc độ học).

$$\theta_{t+1} = \theta_t - \eta \nabla \mathrm{J}(\theta_{t,i})$$

Trong đó:

- θ_{t+1} là trọng số mới sau bước thời gian t + 1

- θ_t là trọng số tại bước thời gian t
- η là learning rate, quyết định tốc độ học của thuật toán
- $\nabla J(\theta_{t,i})$ là gradient của hàm mất mát J theo trọng số $\theta_{t,i}$ dựa trên một mẫu ngẫu nhiên tại thời điểm t

Lặp lại quá trình: Quá trình trên được lặp lại qua nhiều vòng lặp (epochs) hoặc qua toàn bộ tập dữ liệu huấn luyện.

1.3.2 Ưu điểm

Tính linh hoạt: Đặc điểm ngẫu nhiên giúp tránh khỏi các điểm cực tiểu cục bộ và tạo sự đa dạng trong quá trình tối ưu hóa.

Nhanh chóng và hiệu quả với dữ liệu lớn: Do chỉ cần tính toán gradient trên một điểm hoặc một mini-batch nhỏ.

Yêu cầu ít bộ nhớ

1.3.3 Nhược điểm:

Phương sai cao, dễ dẫn đến overfitting

Yêu cầu mô hình thay đổi liên tục (online learning)

1.4 Adagrad

AdaGrad là một phương pháp tối ưu hóa trong học máy, đặc biệt được sử dụng trong quá trình huấn luyện mô hình để cập nhật trọng số theo gradient của hàm mất mát. Phương pháp này có khả năng điều chỉnh tỷ lệ học (learning rate) cho từng tham số của mô hình dựa trên lịch sử của gradient đã tính toán cho tham số đó. Không giống như các thuật toán trước đó thì learning rate hầu như giống nhau trong quá trình training (learning rate là hằng số), Adagrad coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên sau mỗi thời điểm t.

1.4.1 Hoạt động

Tính toán gradient: AdaGrad tính toán gradient của hàm mất mát đối với từng tham số của mô hình.

Điều chỉnh learning rate: AdaGrad sử dụng một learning rate tự điều chỉnh cho mỗi tham số. Learning rate này phụ thuộc vào lịch sử của gradient đã tính toán cho tham số đó. Cách tính toán learning rate trong AdaGrad là bình phương của tổng bình phương của các gradient đã tính trước đó.

Công thức cập nhật trọng số:

$$\theta_{t+1,i} = \theta_{t,i} - \left(\frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}}\right) . g_{t,i}$$

Trong đó:

- $\theta_{t+1,i}$ là giá trị mới của tham số thứ i sau bước cập nhật.
- $\theta_{t,i}$ là giá trị hiện tại của tham số thứ i.
- η là learning rate ban đầu.
- $g_{t,i}$ là gradient của tham số thứ i tại thời điểm t.
- $G_{t,ii}$ là tổng bình phương của các gradient đã tính trước đó cho tham số thứ i.
- ε là một giá trị nhỏ (như 10^-8) được thêm vào để tránh việc chia cho 0.

1.4.2 Ưu điểm:

Không cần điều chỉnh thủ công tốc độ học tập.

Hội tụ nhanh hơn

Đáng tin cậy hơn

1.4.3 Nhược điểm:

Tỷ lệ học (learning rate) trở nên nhỏ khi mạng neural có độ sâu tăng lên.

Dễ dẫn đến dead neuron

1.5 RMSProp

RMSprop là một phương pháp tối ưu hóa thường được sử dụng trong quá trình huấn luyện mạng neural, đặc biệt là trong các mô hình sử dụng hàm kích hoạt phi tuyến tính như ReLU.

1.5.1 Hoạt động

Exponential Moving Average (EMA) của bình phương gradient: RMSprop theo dõi EMA của bình phương gradient cho mỗi tham số của mạng neural.

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

Trong đó:

- $E[g^2]_t$ là EMA của bình phương gradient ở thời điểm t
- g_t là gradient tại thời điểm t
- β là trọng số giữa các bước gradient trước đó và bước gradient hiện tại (thường trong khoản 0.8 – 0.999)

Thay đổi learning rate: RMSprop chia learning rate cho căn bậc hai của EMA bình phương gradient tương ứng. Điều này giúp làm giảm learning rate cho các tham số mà gradient lớn và tăng learning rate cho các tham số mà gradient nhỏ.

$$learning\_rate_t = \frac{learning\_rate_0}{\sqrt{E[g^2]_t + \epsilon}}$$

Trong đó:

- $learning\_rate_t$ là learning rate ở thời điểm t
- $learning\_rate_0$ là learning rate ở thời điểm ban đầu
- $E[g^2]_t$ là EMA của bình phương gradient ở thời điểm t
- ϵ là một giá trị nhỏ (như 10^-8) được thêm vào để tránh việc chia cho 0.

Tính toán và cập nhật trọng số: RMSprop sử dụng learning rate đã điều chỉnh để cập nhật trọng số của mạng neural theo hướng giảm độ lớn của gradient.

Điều chỉnh adaptive learning rate: Phương pháp này giúp ổn định quá trình tối ưu hóa và giảm vấn đề về learning rate quá lớn hoặc quá nhỏ.

1.5.2 Ưu điểm

Giải quyết được vấn đề tốc độ học giảm dần của Adagrad (vấn đề tốc độ học giảm dần theo thời gian sẽ khiến việc training chậm dần, có thể dẫn tới bị đóng băng)

Hiệu suất trong việc học từ dữ liệu phi tuyến tính: Đặc biệt hiệu quả với các mô hình sử dụng hàm kích hoạt phi tuyến tính như ReLU.

Khả năng tối ưu hóa tốt hơn: RMSprop thường hoạt động tốt trong việc tối ưu hóa mô hình mạng neural và giảm thiểu vấn đề về learning rate không ổn định.

1.5.3 Nhược điểm:

Có thể cho kết quả nghiệm chỉ là cực tiểu cục bộ chứ không đạt được cực tiểu toàn cục

Không cập nhật learning rate theo từng tham số một cách riêng biệt, có thể dẫn đến hiệu suất huấn luyện không tối ưu đối với các mô hình có độ phức tạp cao.

1.6 Momentum

Tối ưu hóa theo Momentum là một phương pháp tối ưu hóa trong quá trình huấn luyện mạng neural, nhằm cải thiện quá trình hội tụ và tăng tốc độ của việc tối ưu hóa.

1.6.1 Hoạt động

Momentum: Đà đại diện cho độ lớn và hướng của việc cập nhật trọng số. Nó giữ thông tin về hướng di chuyển từ các bước cập nhật trước đó và giúp giảm thiểu độ dao động khi di chuyển đến điểm tối ưu.

Các Bước Tính Toán:

- Tại mỗi bước thời gian trong quá trình huấn luyện:
  + Tính gradient của hàm mất mát theo trọng số hiện tại
  + Cập nhật đà dựa trên gradient hiện tại và đà từ bước trước theo công thức

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

Trong đó:

- v_t là đà tại thời điểm t
- β là hệ số momentum, thường nằm trong khoảng 0.8 – 0.999

- η là learning rate
- $\nabla J(\theta_t)$ là gradient của hàm mất mát J theo trọng số θ tại thời điểm t

Cập nhật trọng số: sau khi tính toán đà, cập nhật trọng số mới bằng công thức

$$\theta_{t+1} = \theta_t - v_t$$

Trong đó:

- θ_{t+1} là trọng số mới sau bước thời gian t+1
- θ_t là trọng số tại bước thời gian t
- v_t là đà tính toán tại thời điểm t

1.6.2 Ưu Điểm:

Hội tụ Nhanh Hơn: Giúp mô hình nhanh chóng hội tụ tới điểm tối ưu hóa.

Ổn Định Hơn: Giúp giảm đi sự dao động không cần thiết trong quá trình huấn luyện.

1.6.3 Nhược Điểm:

Tùy Chỉnh Hệ Số Momentum (β): Sử dụng sai hệ số momentum có thể làm ảnh hưởng đến tốc độ hội tụ và ổn định của quá trình tối ưu hóa.

1.7 Adam

Adam (Adaptive Moment Estimation) là một phương pháp tối ưu hóa thông dụng trong huấn luyện mạng neural. Nó kết hợp cả Momentum và RMSprop, cung cấp sự linh hoạt và hiệu quả trong việc tối ưu hóa hàm mất mát.

1.7.1 Hoạt động

Tính toán Momentum: Adam sử dụng đà (momentum) để tính toán thông tin về hướng di chuyển từ các gradient trước đó, tương tự như Momentum Optimization.

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1)(\nabla J(\theta_t))^2$$

Trong đó:

- v_t là đà tại thời điểm t.

- β_1 là hệ số momentum (thường là 0.9).
- $\nabla J(\theta_t)$ là gradient của hàm mất mát J theo trọng số θ tại thời điểm t

Tính toán RMSprop: Adam cũng sử dụng RMSprop để điều chỉnh learning rate theo từng tham số riêng lẻ, giúp cân bằng giữa việc cập nhật các trọng số lớn và nhỏ.

$$s_t = \beta_2 s_{t-1} (\nabla J(\theta_t))^2$$

Trong đó:

- s_t là squared gradient tại thời điểm t
- β_2 là hệ số giảm giá trị của squared gradient (thường là 0.999)

Kết hợp Momentum và RMSprop: Adam kết hợp cả hai thông tin từ đà và RMSprop để điều chỉnh cả hướng di chuyển và learning rate, giúp quá trình hội tụ nhanh chóng hơn.

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

Trong đó:

- θ_{t+1} là trọng số mới sau bước thời gian t+1
- η là learning rate
- ϵ là một giá trị nhỏ (như 10^-8) được thêm vào để tránh việc chia cho 0.

1.8 So sánh

Optimizer	**Hoạt Động**	**Ưu Điểm**	**Nhược Điểm**
Gradient Descent	Tính Gradient, Cập Nhật Tham Số	Đơn giản, Hiệu Suất Tốt, Phổ Biến	Dễ Rơi vào Cực Tiểu Cục Bộ, Hội Tụ Chậm, Phụ Thuộc vào Learning Rate

SGD	Chọn Mẫu Ngẫu Nhiên, Tính Gradient, Cập Nhật	Tính Linh Hoạt, Nhanh Chóng với Dữ Liệu Lớn	Phương Sai Cao, Yêu Cầu Mô Hình Thay Đổi Liên Tục
Adagrad	Tính Gradient, Điều Chỉnh Learning Rate	Không Cần Điều Chỉnh Learning Rate, Hội Tụ Nhanh	Tỷ Lệ Học Giảm Dần, Dễ Dẫn Đến Dead Neuron
RMSprop	Exponential Moving Average, Thay Đổi Learning Rate	Giải Quyết Tốt Vấn Đề Tốc Độ Học Giảm Dần, Hiệu Suất Với Dữ Liệu Phi Tuyến	Có Thể Đạt Cực Tiểu Cục Bộ, Không Cập Nhật Learning Rate Riêng Biệt
Momentum	Đà, Tính Toán Gradient, Cập Nhật Trọng Số	Hội Tụ Nhanh Hơn, Ổn Định Hơn	Tùy Chỉnh Hệ Số Momentum, Không Ổn Định Nếu Hệ Số Sai
Adam	Tính Toán Momentum và RMSprop, Cập Nhật	Linh Hoạt, Hội Tụ Nhanh, Ổn Định Hơn	Có Thể Đạt Cực Tiểu Cục Bộ, Phức Tạp Trong Điều Chỉnh Tham Số

Bảng 1.1: So sánh các Optimizer

Như vậy, mỗi Optimizer có các ưu điểm và nhược điểm riêng, và việc lựa chọn Optimizer thường phụ thuộc vào loại dữ liệu, cấu trúc mô hình, và yêu cầu cụ thể của bài toán.

CHƯƠNG 2. CÂU 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.

2.1 Continual Learning

2.1.1 Định nghĩa

Continual Learning còn được gọi là học máy liên tục (CML), là một quá trình trong đó một mô hình học từ các luồng dữ liệu mới mà không cần đào tạo lại.

Trái ngược với các phương pháp tiếp cận truyền thống, trong đó các mô hình được đào tạo trên một tập dữ liệu tĩnh, được triển khai và đào tạo lại định kỳ, các mô hình học liên tục cập nhật lặp đi lặp lại các tham số của chúng để phản ánh các phân phối mới trong dữ liệu.

Mô hình tự cải thiện bằng cách học hỏi từ lần lặp mới nhất và cập nhật kiến thức của nó khi có dữ liệu mới. Vòng đời của mô hình học tập liên tục cho phép các mô hình duy trì tính phù hợp theo thời gian nhờ chất lượng năng động vốn có của chúng.

2.1.2 Các loại Continual Learning

Các phương pháp học liên tục bao gồm học tăng dần (incremental learning), chuyển giao học (transfer learning), và học suốt đời (lifelong learning). Ngoài ra còn có các phương pháp khác như kỹ thuật lặp lại kinh nghiệm (experience replay) và kỹ thuật điều chuẩn (regularization).

Học tăng dần (Incremental Learning): Phương pháp này cho phép mô hình học từ dữ liệu mới mà không cần huấn luyện lại trên toàn bộ tập dữ liệu. Thay vào đó, nó tiếp tục học từ các mẫu mới một cách tuần tự và cập nhật mô hình theo từng đợt dữ liệu mới. Điều này giúp giảm thiểu sự rườm rà của việc huấn luyện lại toàn bộ mô hình.

Học chuyển giao (Transfer Learning): Phương pháp này sử dụng kiến thức đã học từ một tác vụ và áp dụng nó vào tác vụ khác. Thay vì bắt đầu từ mô hình không

đổi, chúng ta sử dụng một mô hình đã được huấn luyện trên một nhiệm vụ tương tự để tận dụng kiến thức đã học từ nhiệm vụ đó cho nhiệm vụ mới.

Học suốt đời (Lifelong Learning): Đây là một phương pháp học liên tục tập trung vào việc giữ lại kiến thức từ dữ liệu cũ và sử dụng nó để học từ dữ liệu mới. Mô hình được huấn luyện để liên tục tích lũy và điều chỉnh kiến thức của mình từ thông tin mới.

Kỹ thuật lặp lại kinh nghiệm (Experience Replay): Phương pháp này sử dụng các trải nghiệm đã lưu trữ từ quá khứ để huấn luyện mô hình. Thay vì chỉ học từ dữ liệu mới, mô hình còn sử dụng lại trải nghiệm đã có để cải thiện hiệu suất học.

Kỹ thuật điều chuẩn (Regularization Techniques): Đây là một kỹ thuật sử dụng trong việc kiểm soát và tránh overfitting khi huấn luyện mô hình trên dữ liệu mới. Điều này giúp mô hình học từ dữ liệu mới mà không bị quá khớp (overfit) với dữ liệu huấn luyện.

Mỗi phương pháp này có ưu điểm và hạn chế riêng. Sự lựa chọn của phương pháp phụ thuộc vào bản chất của dữ liệu, cấu trúc mô hình, hiệu suất mong muốn, độ phức tạp của nhiệm vụ và tài nguyên tính toán có sẵn. Trong nhiều trường hợp, việc kết hợp các phương pháp này có thể tăng cường khả năng học của mô hình.

2.1.3 Quá trình

Huấn luyện ban đầu (Initial Training): Bước này bao gồm việc huấn luyện mô hình ban đầu trên một tập dữ liệu cụ thể để nắm bắt kiến thức ban đầu từ dữ liệu có sẵn.

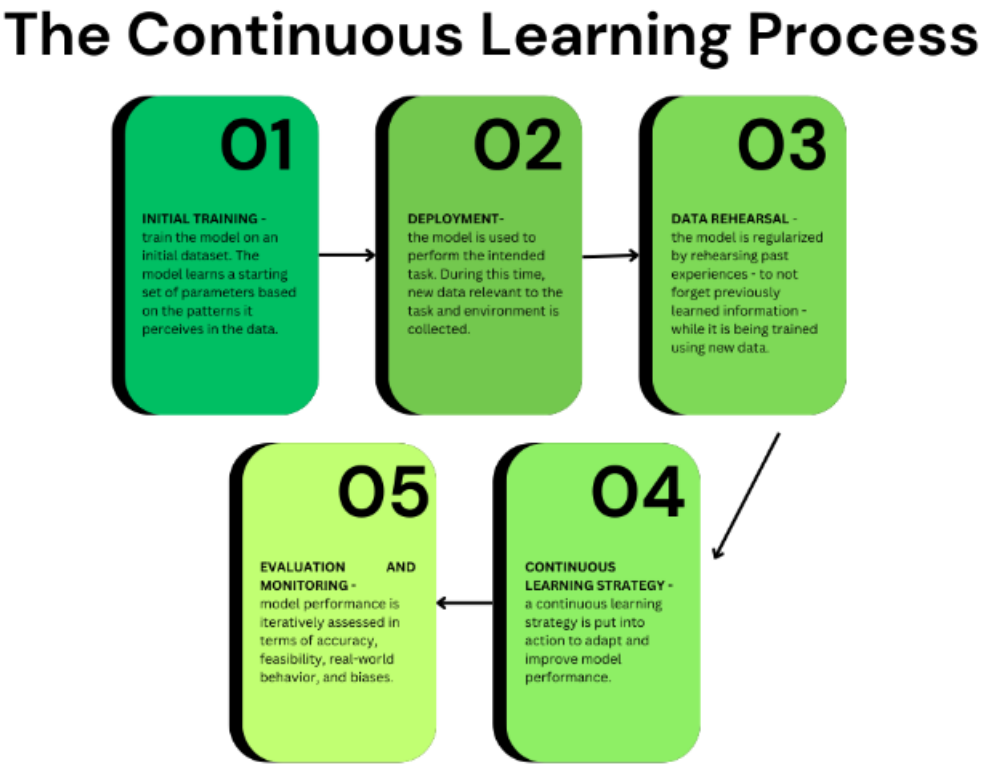
Triển khai (Deployment): Mô hình được triển khai để sử dụng trong môi trường thực tế hoặc trong sản phẩm, dịch vụ.

Rehearsal Data (Dữ liệu Luyện Tập Lại): Quá trình này bao gồm việc sử dụng dữ liệu mới và thông tin cũ để huấn luyện lại mô hình. Dữ liệu luyện tập lại có thể bao gồm dữ liệu mới, những dữ liệu có tính liên quan hoặc gần giống với dữ liệu đã sử dụng ban đầu.

Chiến lược học liên tục (Continuous Learning Strategy): Đây là bước quan trọng giúp mô hình học từ dữ liệu mới một cách hiệu quả. Chiến lược này có thể bao

gồm các phương pháp cập nhật mô hình, điều chỉnh tham số, tối ưu hóa kiến trúc mô hình hoặc thậm chí là việc chọn lại dữ liệu.

Đánh giá và theo dõi (Evaluation and Monitoring): Bước này liên tục theo dõi hiệu suất của mô hình trên dữ liệu mới và trong môi trường thực tế. Việc đánh giá và theo dõi này giúp xác định xem mô hình có thể cải thiện hay cần điều chỉnh thêm không.



Hình 2.1: Quy trinh Continual Learning

Mỗi bước trong quá trình học liên tục đóng vai trò quan trọng để đảm bảo rằng mô hình không chỉ học từ dữ liệu mới mà còn duy trì và cải thiện tính hiệu quả của nó theo thời gian.

2.1.4 Ưu điểm

Tính tổng quát: Học liên tục giúp mô hình trở nên mạnh mẽ và chính xác hơn khi đối mặt với dữ liệu mới. Nó có khả năng áp dụng kiến thức từ quá khứ vào các tình huống mới, giúp mô hình trở nên linh hoạt hơn và tối ưu hóa dự đoán.

Lưu giữ thông tin: Bằng cách sử dụng chiến lược học liên tục, mô hình xem xét các kiến thức trước đó thu được từ các lần lặp trước đó, từ đó tích luỹ thông tin theo thời gian. Điều này giúp mô hình giữ lại và tận dụng kiến thức đã học để cải thiện hiệu suất.

Tính linh hoạt: Mô hình sử dụng học liên tục có khả năng thích ứng với kiến thức mới, chẳng hạn như sự thay đổi trong dữ liệu hoặc xu hướng mới. Điều này tăng khả năng dự đoán của mô hình trong dài hạn, đồng thời giúp nó phản ứng linh hoạt với sự thay đổi của môi trường.

2.1.5 Nhược điểm

Chi phí: Phương pháp học liên tục, mặc dù hiệu quả, thường phức tạp hơn tính toán so với các phương pháp truyền thống vì mô hình cần điều chỉnh liên tục theo dữ liệu mới. Điều này dẫn đến chi phí kinh tế cao hơn vì yêu cầu nhiều tài nguyên về dữ liệu, nhân lực và tính toán hơn.

Quản lý mô hình: Mỗi khi tham số của mô hình được cập nhật dựa trên dữ liệu mới, một mô hình mới được hình thành. Do đó, phương pháp học liên tục có thể tạo ra nhiều mô hình, làm phức tạp việc xác định mô hình nào hoạt động tốt nhất.

Thay đổi dữ liệu: Đối với phương pháp học liên tục để đạt hiệu quả, chúng ta cần xử lý một lượng lớn dữ liệu mới. Tuy nhiên, mô hình như vậy có nguy cơ mất khả năng dự đoán nếu phân phối đặc trưng thay đổi đột ngột. Điều này được gọi là hiện tượng "thay đổi dữ liệu" (data drift).

2.1.6 Ứng dụng

Thị giác máy tính: Tính động của dữ liệu dựa trên hình ảnh làm cho các phương pháp học liên tục trở nên phổ biến trong việc huấn luyện thuật toán nhận diện và phân loại thông tin hình ảnh. Các ứng dụng thường xuyên được áp dụng trong việc nhận diện khuôn mặt và công nghệ hình ảnh.

Bảo mật mạng: Các phương pháp học liên tục được triển khai để đảm bảo theo dõi liên tục trong cơ sở hạ tầng bảo mật IT. Chúng đóng vai trò quan trọng trong việc

phát hiện lừa đảo, xâm nhập mạng và thư rác, cùng các hoạt động liên quan đến bảo mật khác.

Chăm sóc sức khỏe: Do bản chất tiến hóa của các bệnh tật, các phương pháp học liên tục được sử dụng trong các lĩnh vực chăm sóc sức khỏe khác nhau để tăng cường quy trình chẩn đoán bệnh. Các chuyên ngành như ung thư học và chẩn đoán hình ảnh đã sớm thử nghiệm AI học liên tục để mục đích này.

Robot học: Học liên tục được áp dụng để nâng cao tính linh hoạt và hiệu suất của robot trong các môi trường khác nhau, giúp chúng tối ưu hóa hành động dựa trên trải nghiệm qua quá khứ và mới.

2.2 Test Production

2.2.1 Định nghĩa

Test Production trong việc xây dựng giải pháp học máy là một bước quan trọng để đảm bảo rằng mô hình học máy đã được triển khai hoạt động chính xác và hiệu quả trong môi trường thực tế. Đây là quá trình kiểm tra mô hình trước khi nó được triển khai để đảm bảo rằng nó hoạt động như mong đợi và có khả năng chịu đựng với dữ liệu thực tế. Bước này giúp mô hình hoạt động đáng tin cậy và hiệu quả, đồng thời kiểm tra khả năng chịu đựng và đối phó với dữ liệu mới.

2.2.2 Hoạt động

2.2.2.1 Chuẩn bị mô hình

Huấn luyện và đánh giá: Xây dựng mô hình học máy từ dữ liệu huấn luyện và đánh giá hiệu suất của nó trên dữ liệu kiểm định.

Tối ưu hóa: Điều chỉnh và tối ưu hóa mô hình để đạt được hiệu suất tốt nhất có thể trên tập dữ liệu huấn luyện và kiểm định.

2.2.2.2 Kiểm thử sản xuất (Production Testing)

Triển khai mô hình: Đưa mô hình từ môi trường phát triển sang môi trường sản xuất.

Kiểm tra tính ổn định: Đảm bảo rằng mô hình hoạt động ổn định trong môi trường thực tế và không gặp phải lỗi không mong muốn.

Kiểm tra hiệu suất: Đánh giá lại hiệu suất của mô hình trên dữ liệu thực tế để xác nhận tính chính xác và khả năng dự đoán của nó.

2.2.2.3 Điều chỉnh và tinh chỉnh

Phản hồi từ sản xuất: Dựa trên kết quả kiểm thử, điều chỉnh mô hình để cải thiện hiệu suất và ổn định.

Tối ưu hóa lại: Tinh chỉnh các tham số của mô hình hoặc quá trình huấn luyện để cải thiện hiệu suất dự đoán trong môi trường sản xuất.

2.2.2.4 Triển khai mô hình

Triển khai tổng cục: Đưa mô hình đã được kiểm thử và điều chỉnh vào môi trường sản xuất.

Chuẩn bị tài nguyên: Bao gồm việc chuẩn bị cơ sở hạ tầng, dữ liệu, và quy trình để triển khai mô hình vào hệ thống thực tế.

2.2.2.5 Giám sát và cập nhật

Giám sát liên tục: Theo dõi hoạt động của mô hình trong môi trường thực tế để phát hiện và khắc phục vấn đề nếu có.

Cập nhật mô hình: Dựa trên dữ liệu mới và phản hồi từ môi trường sản xuất, cập nhật mô hình để duy trì hoặc nâng cao hiệu suất và độ ổn định của nó.

2.2.3 Ưu điểm

Tính chính xác: Kiểm thử sản xuất giúp đảm bảo mô hình hoạt động chính xác trong môi trường thực tế.

Tính ổn định: Mô hình được kiểm thử để đảm bảo nó hoạt động ổn định và hiệu suất ở mức cao trong thời gian dài.

Tối ưu hóa liên tục: Cập nhật và tinh chỉnh mô hình để duy trì hoặc nâng cao hiệu suất theo thời gian.

2.2.4 Nhược điểm

Chi phí và thời gian: Quá trình kiểm thử sản xuất có thể tốn kém về thời gian và nguồn lực.

Rủi ro khi triển khai: Mô hình có thể không hoạt động tốt trong môi trường thực tế do các yếu tố không dự đoán được.