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

BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN

NHẬP MÔN HỌC MÁY

…tên đề tài…

Người hướng dẫn: **PGS.TS Lê Anh Cường**
Người thực hiện: **Trịnh Ngọc Trung Trực**
Lớp **: 21050301**
Khoá **: 25**

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

BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN
NHẬP MÔN HỌC MÁY

…tên đề tài…

Người hướng dẫn: **PGS.TS Lê Anh Cường**
Người thực hiện: **Trịnh Ngọc Trung Trực**
Lớp **: 21050301**
Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

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

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A;. 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 đồ 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 đồ 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

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Văn B

Trần Văn C

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)

TÓM TẮT

Trình bày các kiến thức cơ bản về optimizer, các phương pháp của optimizer, và các ứng dụng của các phương pháp khác nhau trong việc xây dựng mô hình học máy. Bên cạnh đó, tìm hiểu về Continual Learning và Test Production, để tối ưu trong quá trình xây dựng giải pháp học máy.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

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

1.1 Tìm hiểu các phương pháp trong optimizer

1.1.1 Khái quát về optimizer

- Trong học máy, optimizer là một phần quan trọng của quá trình huấn luyện mô hình. Nó đóng vai trò trong việc tối ưu hóa hàm mất mát bằng cách điều chỉnh các trọng số của mô hình để giảm thiểu mức độ mất mát. Dưới đây là một số điểm quan trọng về optimizer trong học máy:
- Optimizer đóng một vai trò quan trọng trong quá trình huấn luyện mô hình học máy, giúp điều chỉnh trọng số mô hình để đạt được hiệu suất tốt trên dữ liệu đào tạo và dữ liệu mới. Sự lựa chọn của optimizer và các siêu tham số liên quan đến nó có thể ảnh hưởng đáng kể đến khả năng học và hiệu suất của mô hình.

1.1.2 Gradient Descent (GD)

- Gradient Descent (GD) là một phương pháp tối ưu hóa được sử dụng để điều chỉnh các tham số của mô hình trong quá trình huấn luyện. Thuật ngữ "Gradient" đề cập đến đạo hàm của hàm mất mát, và "Descent" đề cập đến việc di chuyển theo hướng ngược với gradient để giảm thiểu hàm mất mát.
- Cụ thể, trong mỗi bước của quá trình huấn luyện, GD tính toán gradient của hàm mất mát theo từng tham số của mô hình và sau đó cập nhật các tham số bằng cách di chuyển một khoảng nhất định theo hướng ngược với gradient. Công thức cập nhật thường là:
    - $\text{tham\_so\_moi} = \text{tham\_so\_cu} - \text{learning\_rate} \times \text{gradient}$
- Trong đó:
    - tham_so_moi là giá trị mới của tham số.
    - tham_so_cu là giá trị hiện tại của tham số.

- learning_rate là tốc độ học, là một hyperparameter quyết định độ lớn của bước cập nhật.

- Ưu Điểm:
  - **Đơn Giản và Dễ Triển Khai:** GD là phương pháp đơn giản và dễ triển khai, không đòi hỏi nhiều tài nguyên tính toán.
- Nhược Điểm:
  - **Dễ Rơi vào Điểm Cực Tiểu Địa Phương:** GD có thể dễ rơi vào điểm cực tiểu địa phương thay vì điểm cực tiểu toàn cục, đặc biệt trên các bề mặt lỗi phức tạp.
  - **Yêu Cầu Chọn Learning Rate:** Việc chọn learning rate là một thách thức và ảnh hưởng lớn đến hiệu suất của thuật toán. Learning rate quá lớn có thể làm cho thuật toán không hội tụ hoặc vượt qua điểm tối ưu, trong khi learning rate quá nhỏ có thể làm chậm quá trình học.
- Tổng quan, Gradient Descent là một cơ sở quan trọng và thường được sử dụng trong nhiều bài toán học máy. Tuy nhiên, để khắc phục nhược điểm, nhiều biến thể như Stochastic Gradient Descent (SGD) và các phương pháp tối ưu hóa khác đã được phát triển và sử dụng rộng rãi trong thực tế.

1.1.3 Stochastic Gradient Descent (SGD)

- Stochastic Gradient Descent (SGD) là một biến thể của Gradient Descent (GD) được thiết kế để giảm độ phức tạp tính toán và phù hợp với việc xử lý dữ liệu lớn. Trong SGD, thay vì cập nhật tham số sau mỗi epoch như GD, thì tham số được cập nhật sau mỗi ví dụ đào tạo (mỗi điểm dữ liệu).
- Cách Hoạt Động:
  - **Chọn Ngẫu Nhiên:** Chọn một ví dụ đào tạo ngẫu nhiên từ tập dữ liệu.
  - **Tính Gradient:** Tính gradient của hàm mất mát theo tham số dựa trên ví dụ đào tạo đã chọn.

- **Cập Nhật Tham Số:** Cập nhật tham số bằng cách di chuyển theo hướng ngược với gradient tính được.
- **Lặp Lại:** Lặp lại quá trình trên cho đến khi điều kiện dừng được đạt được (ví dụ: số lần lặp, độ chênh lệch đủ nhỏ).

- Ưu Điểm:
  - **Phù Hợp với Dữ Liệu Lớn:** Đối với tập dữ liệu lớn, việc tính toán trên toàn bộ dữ liệu có thể rất tốn kém. SGD giảm độ phức tạp tính toán bằng cách sử dụng chỉ một ví dụ đào tạo trong mỗi bước.
  - **Giảm Độ Phức Tạp Tính Toán:** Do chỉ sử dụng một ví dụ đào tạo, SGD giảm độ phức tạp tính toán so với GD, làm cho nó phù hợp với các mô hình và tập dữ liệu lớn.
- Nhược Điểm:
  - **Nhiễu Cao:** Do cập nhật tham số sau mỗi ví dụ, SGD có thể gây nhiễu cao và dao động lớn trong quá trình học, đặc biệt nếu dữ liệu không được shuffle trước mỗi epoch.
  - **Điểm Cực Tiểu Không Ổn Định:** SGD có thể dẫn đến việc mô hình không hội tụ đến điểm cực tiểu ổn định, nhưng thay vào đó, nó có thể dao động quanh điểm tối ưu với độ chênh lệch cao.

- Tổng quan, SGD là một phương pháp quan trọng trong quá trình huấn luyện mô hình, đặc biệt là khi có sẵn dữ liệu lớn và muốn giảm độ phức tạp tính toán của quá trình tối ưu hóa. Để giảm nhiễu, các biến thể khác như Mini-Batch Gradient Descent cũng được sử dụng để kết hợp lợi ích của cả GD và SGD.

1.1.4 Mini-Batch Gradient Descent (MBGD)

- Mini-Batch Gradient Descent là một biến thể của Gradient Descent (GD) được sử dụng rộng rãi trong quá trình huấn luyện mô hình máy học. Trong MBGD, thay vì cập nhật tham số sau mỗi epoch như GD hoặc sau mỗi ví dụ như Stochastic Gradient Descent (SGD), thì tham số được cập nhật sau mỗi mini-batch của dữ liệu.
- Cho một mini-batch có kích thước m và tham số hiện tại là θ, hàm mất mát được ký hiệu là $J(\theta)$, gradient của hàm mất mát theo tham số là $\nabla J(\theta)$, và learning rate là η, quá trình cập nhật tham số được thực hiện như sau:
    - $\theta \text{new} = \theta \text{old} - \eta \cdot \nabla J(\theta \text{old})$
- Trong đó:
    - θnew là giá trị mới của tham số.
    - θold là giá trị hiện tại của tham số.
    - η là learning rate.
    - $\nabla J(\theta \text{old})$ là gradient của hàm mất mát theo tham số, được tính toán trên mini-batch hiện tại.
- Công thức này đại diện cho việc di chuyển từ tham số hiện tại θold đến tham số mới θnew theo hướng đối ngược với gradient và với một bước cập nhật được xác định bởi learning rate η.
- Cách Hoạt Động:
    - **Chia Dữ Liệu Thành Mini-Batches:**Tập dữ liệu huấn luyện được chia thành các mini-batches có kích thước nhỏ. Đây là một số lượng nhỏ các điểm dữ liệu được chọn ngẫu nhiên từ tập dữ liệu gốc.
    - **Lặp Qua Từng Mini-Batch:**
        - Quá trình huấn luyện được thực hiện bằng cách lặp lại qua từng mini-batch.
        - Với mỗi mini-batch, gradient của hàm mất mát được tính toán dựa trên mini-batch đó.

- **Cập Nhật Tham Số:**
    - Tham số của mô hình được cập nhật dựa trên gradient tính toán được từ mini-batch.
    - Công thức cập nhật tương tự như trong GD và SGD, nhưng thay vì tính toán gradient trên toàn bộ tập dữ liệu hoặc mỗi điểm dữ liệu, gradient được tính toán dựa trên một mini-batch.
- **Lặp Lại Quá Trình:**
    - Quá trình này được lặp lại qua tất cả các mini-batches cho đến khi đạt được điều kiện dừng, chẳng hạn như số lần lặp, độ chênh lệch đủ nhỏ, hoặc số epoch đã định trước.

- Ưu Điểm:
    - **Phù Hợp với Dữ Liệu Lớn:**
        - MBGD giúp giảm bớt gánh nặng tính toán so với GD khi áp dụng trên dữ liệu lớn.
        - Cho phép cập nhật tham số thường xuyên hơn so với GD mà không đòi hỏi toàn bộ tập dữ liệu.
    - **Đánh Giá Hiệu Suất Mô Hình Thường Xuyên:**
        - Giúp theo dõi và đánh giá hiệu suất mô hình thường xuyên hơn do cập nhật tham số sau mỗi mini-batch.
    - **Giảm Nhiễu:**
        - Giảm nhiễu so với SGD bằng cách tính toán gradient trên một số lượng lớn các điểm dữ liệu trong mỗi mini-batch.
- Nhược Điểm:
    - **Yêu Cầu Chọn Kích Thước Mini-Batch:**
        - Yêu cầu lựa chọn kích thước mini-batch phù hợp. Kích thước quá lớn có thể làm tăng độ phức tạp tính toán, trong khi kích thước quá nhỏ có thể tăng nhiễu và giảm tốc độ hội tụ.

- **Khả Năng Rơi Vào Điểm Cực Tiểu Địa Phương:**
  - Vẫn tồn tại khả năng rơi vào điểm cực tiểu địa phương, đặc biệt khi sử dụng kích thước mini-batch nhỏ.

- **Tổng quan**: mini-batch gradient descent là một giải pháp linh hoạt và hiệu quả cho việc tối ưu hóa mô hình trong học sâu, cung cấp sự cân bằng giữa tốc độ hội tụ và tính hiệu quả tính toán khi xử lý dữ liệu lớn.

1.1.5 Adagrad

- Adagrad là một phương pháp tối ưu hóa mà có thể tự điều chỉnh learning rate cho từng tham số dựa trên lịch sử của gradient của nó. Ý tưởng chính của Adagrad là tận dụng thông tin từ lịch sử gradient để điều chỉnh learning rate, giúp các tham số được cập nhật với độ lớn thích hợp.
- Cách Hoạt Động:
  - **Tính Gradient:** Tính gradient của hàm mất mát theo từng tham số dựa trên một mini-batch của dữ liệu.
  - **Tính Toán Gradient Squared:** Tính tổng bình phương của gradient cho từng tham số qua thời gian:

$$sum\_squared\_gradient = sum\_squared\_gradient + (gradient)^2$$

  - **Tính Toán Learning Rate Tự Điều Chỉnh:** Tính toán learning rate cho mỗi tham số bằng cách chia learning rate ban đầu cho căn bậc hai của tổng bình phương gradient:

$$adaptive\_learning\_rate = \frac{learning\_rate}{\sqrt{sum_{squared_{gradient}} + \epsilon}}$$

Trong đó, ϵ là một hằng số nhỏ để tránh chia cho 0.

- **Cập Nhật Tham Số:** Cập nhật tham số bằng cách sử dụng learning rate đã tính:

- **Lặp Lại:** Lặp lại quá trình trên cho đến khi điều kiện dừng được đạt được.

- Ưu Điểm:
  - **Hiệu Quả trên Dữ Liệu Thưa Thớt:** Adagrad thường hoạt động hiệu quả trên các tập dữ liệu có thưa thớt, nơi một số lượng lớn các tham số có độ quan trọng hơn các tham số khác.
  - **Không Cần Thiết Lập Learning Rate Ban Đầu:** Do tự điều chỉnh learning rate, Adagrad giảm gánh nặng của việc lựa chọn một learning rate ban đầu.
- Nhược Điểm:
  - **Learning Rate Giảm Quá Nhanh:** Vì cộng dồn bình phương gradient, có thể dẫn đến learning rate giảm quá nhanh theo thời gian, làm chậm quá trình học sau một số lượng lớn các bước cập nhật.

- Tổng quan, Adagrad là một phương pháp tối ưu hóa hiệu quả, đặc biệt là trên các tập dữ liệu có đặc trưng thưa thớt. Tuy nhiên, với sự giảm nhanh của learning rate, các biến thể như RMSProp và Adam đã được đề xuất để giải quyết vấn đề này và cải thiện hiệu suất trong một số trường hợp.

1.1.6 RMSProp (Root Mean Square Propagation)

- RMSProp (Root Mean Square Propagation) là một phương pháp tối ưu hóa được thiết kế để giảm vấn đề của Adagrad, đặc biệt là vấn đề của việc giảm learning rate quá nhanh khi tích dồn bình phương gradient. RMSProp giữ lại ưu điểm của Adagrad trong việc tự điều chỉnh learning rate cho từng tham số, nhưng giảm độ lớn của learning rate theo thời gian.

- RMSProp giải quyết vấn đề trên bằng cách sử dụng một trung bình chạy của gradient bình phương để chuẩn hóa gradient. Việc chuẩn hóa này cân bằng kích thước bước (momentum), giảm bước cho gradient lớn để tránh sự phát triển đột ngột và tăng bước cho gradient nhỏ để tránh sự biến mất.
- Cách Hoạt Động:
  - **Tính Gradient:** Tính gradient của hàm mất mát theo từng tham số dựa trên một mini-batch của dữ liệu.
  - **Tính Toán Gradient Squared:** Tính toán trung bình cộng bình phương gradient qua thời gian:

    $$\text{moving\_average\_squared\_gradient} = \beta \times \text{moving\_average\_squared\_gradient} + (1 - \beta) \times (\text{gradient})^2$$

    Trong đó, β là một hệ số giảm.
  - **Tính Toán Learning Rate Tự Điều Chỉnh:** Tính toán learning rate cho mỗi tham số bằng cách chia learning rate ban đầu cho căn bậc hai của moving average của bình phương gradient:

    $\text{adaptive\_learning\_rate} = \frac{\text{learning\_rate}}{\sqrt{\text{sum}_{\text{squared}_{\text{gradient}}} + \epsilon}}$,

    ϵ là một hằng số nhỏ để tránh chia cho 0.
  - **Cập Nhật Tham Số:** Cập nhật tham số bằng cách sử dụng learning rate đã tính:

    $$\text{tham\_so\_moi} = \text{tham\_so\_cu} - \text{adaptive\_learning\_rate} \times \text{gradient}$$
  - **Lặp Lại:** Lặp lại quá trình trên cho đến khi điều kiện dừng được đạt được.
- Ưu Điểm:
  - **Hiệu Quả Trên Các Bề Mặt Lỗi Phức Tạp:** RMSProp thường hoạt động hiệu quả trên các bề mặt lỗi có tính chất phức tạp.

- **Tự Điều Chỉnh Learning Rate:** Có khả năng tự điều chỉnh learning rate cho từng tham số, giúp tối ưu hóa quá trình học.

- Nhược Điểm:
  - **Cần Phải Điều Chỉnh Hyperparameter:** Như nhiều phương pháp tối ưu hóa khác, RMSProp cần phải điều chỉnh các hyperparameter như hệ số giảm β để đạt được hiệu suất tốt nhất.

- **Tổng quan:** RMSProp là một trong những phương pháp tối ưu hóa quan trọng trong học máy, và nó đã đóng góp vào sự phát triển của các biến thể khác như Adam, cũng kết hợp lợi ích của Momentum và RMSProp.

1.1.7 Momentum

- Momentum là một phương pháp tối ưu hóa được thiết kế để giúp tăng tốc độ hội tụ của thuật toán Gradient Descent bằng cách tích lũy một phần của gradient trước đó. Ý tưởng chính là giảm độ chói và giúp vượt qua điểm cực tiểu địa phương bằng cách giữ lại đà (momentum) của bước trước đó.
- Cách Hoạt Động:
  - **Tính Gradient:** Tính gradient của hàm mất mát theo từng tham số dựa trên một mini-batch của dữ liệu.
  - **Tích Lũy Momentum:** Tích lũy một phần của gradient trước đó vào momentum:
    - $\text{momentum} = \beta \times \text{momentum} + (1 - \beta) \times \text{gradient}$

      Trong đó, β là hệ số momentum, thường có giá trị gần 1.
  - **Cập Nhật Tham Số:** Cập nhật tham số bằng cách sử dụng momentum tích lũy:

    $$\text{tham\_so\_moi} = \text{tham\_so\_cu} - \text{learning\_rate} \times \text{momentum}$$
  - **Lặp Lại:** Lặp lại quá trình trên cho đến khi điều kiện dừng được đạt được.

- Ưu Điểm:
    - **Giảm Độ Chói:** Momentum giúp giảm độ chói của thuật toán, giúp thuật toán vượt qua điểm cực tiểu địa phương và tiếp tục hướng về điểm tối ưu toàn cục.
    - **Tăng Tốc Độ Hội Tụ:** Giúp tăng tốc độ hội tụ, đặc biệt là trên các bề mặt lỗi phức tạp.
- Nhược Điểm:
    - **Cần Phải Chọn Hệ Số Momentum:** Cần phải chọn giá trị thích hợp cho hệ số momentum. Giá trị quá lớn có thể làm cho thuật toán dao động mạnh và vượt qua điểm tối ưu, trong khi giá trị quá nhỏ có thể làm chậm quá trình học.
- **Tổng quan:** Momentum là một phương pháp tối ưu hóa mạnh mẽ và thường được sử dụng để cải thiện hiệu suất của Gradient Descent, đặc biệt là trong những trường hợp mà bề mặt lỗi có nhiều biến thiên và điểm cực tiểu địa phương.

1.1.8 Adam (Adaptive Moment Estimation)

- Adam là một phương pháp tối ưu hóa kết hợp lợi ích của Momentum và RMSProp. Nó sử dụng cả first-order moment (momentum) và second-order moment (độ lớn của gradient) để cập nhật tham số và tự điều chỉnh learning rate cho từng tham số.
- Cách Hoạt Động:
    - **Tính Gradient:** Tính gradient của hàm mất mát theo từng tham số dựa trên một mini-batch của dữ liệu.
    - **Tích Lũy First-order Moment (Momentum):** Tích lũy first-order moment bằng cách tính trung bình cộng của gradient trước đó:
        - $\text{momentum} = \beta 1 \times \text{momentum} + (1 - \beta 1) \times \text{gradient}$
          Trong đó, $\beta 1$ là hệ số momentum thường có giá trị gần 1.

- **Tích Lũy Second-order Moment (RMSProp):** Tích lũy second-order moment bằng cách tính trung bình cộng bình phương gradient qua thời gian:

$$\text{squared}_{\text{gradient}} = \beta 2 \times \text{squared\_gradient} + (1 - \beta 2) \times (\text{gradient})^2$$

Trong đó, $\beta 2$ là hệ số giảm tốc, thường có giá trị gần 1.

- **Hiệu Chỉnh Moment Đầu Tiên và Moment Thứ Hai:** Để giảm bias của first-order moment và second-order moment do bắt đầu từ 0, ta thực hiện bước này:

$$\text{momentum}^{\wedge} = \frac{momentum}{1 - \beta_2^t}$$

$$\text{squared\_gradient}^{\wedge} = \frac{squared\_gradient}{1 - \beta_2^t}$$

Trong đó, *t* là số bước lặp hiện tại.

- **Cập Nhật Tham Số:** Cập nhật tham số bằng cách sử dụng first-order moment và second-order moment đã được hiệu chỉnh:

$$\text{tham}_{\text{so}_{\text{moi}}} = \text{tham}_{\text{so}_{\text{cu}}} - \text{learning}_{\text{rate}} \times \frac{\text{momentum}^{\wedge}}{\sqrt{squared_{gradient}} + \epsilon}$$

Trong đó, ϵ là một hằng số nhỏ để tránh chia cho 0.

- **Lặp Lại:** Lặp lại quá trình trên cho đến khi điều kiện dừng được đạt được.

- Ưu Điểm:
  - **Hiệu Quả và Được Sử Dụng Rộng Rãi:** Adam thường rất hiệu quả và được sử dụng rộng rãi trong nhiều ứng dụng học máy.
  - **Tự Điều Chỉnh Learning Rate:** Adam tự động điều chỉnh learning rate cho từng tham số, giúp tối ưu hóa quá trình học.
- Nhược Điểm:

- **Cần Phải Điều Chỉnh Nhiều Hyperparameter:** Adam cần phải điều chỉnh nhiều hyperparameter như $\beta 1$, $\beta 2$, và ϵ, điều này có thể làm tăng độ phức tạp của quá trình huấn luyện và đôi khi yêu cầu sự thử nghiệm.

- **Tổng quan**: Adam là một phương pháp tối ưu hóa mạnh mẽ và thường là lựa chọn ưa thích trong nhiều tình huống. Tuy nhiên, việc lựa chọn các hyperparameter đòi hỏi sự cân nhắc cẩn thận để đảm bảo hiệu suất tối ưu.

1.1.9 Learning Rate Decay (Giảm Tỷ Lệ Học)

- **Learning Rate (Tỷ Lệ Học):** Là một siêu tham số trong các thuật toán tối ưu hóa, xác định bước kích thước mỗi lần cập nhật trọng số trong quá trình đào tạo mô hình.

- **Learning Rate Decay:** Là một chiến lược giảm dần tỷ lệ học theo thời gian hoặc qua mỗi epoch/training step để điều chỉnh quá trình học của mô hình. Mục tiêu của việc giảm learning rate là giảm bước cập nhật của gradient descent theo thời gian, giúp thuật toán hội tụ một cách ổn định và hiệu quả hơn cho các mô hình Gradient Descent.

- **Các Phương Pháp Learning Rate Decay Phổ Biến:**

  - **Linear Decay (Giảm Tuyến Tính):**
    - **Công Thức:**
      $\text{learning\_rate} = \text{initial\_learning\_rate} \times (1 - \text{decay} \times \text{epoch\_number})$
    - **Ưu Điểm:** Dễ triển khai, giảm tỷ lệ học một cách đều dần.
  - **Exponential Decay (Giảm Mũ):**
    - **Công Thức:**
      $\text{learning\_rate} = \text{initial\_learning\_rate} \times \exp(-\text{decay} \times \text{epoch\_number})$

- **Ưu Điểm:** Đưa ra giảm tỷ lệ học nhanh chóng ở đầu và giảm chậm dần.

- **Step Decay (Giảm Bước):**
  - **Công Thức:**

$$\text{learning\_rate} = \text{initial\_learning\_rate} \times \text{drop}^{\text{epochs\_per\_decay}\text{epoch\_number}+1}$$

  - **Ưu Điểm:** Cung cấp sự linh hoạt bằng cách giảm tỷ lệ học sau mỗi số lượng epoch nhất định.

- **Ưu Điểm:**

  - **Stabilize Training:** Giảm tỷ lệ học giúp ổn định quá trình đào tạo, đặc biệt khi mô hình đang gặp vấn đề về dao động hoặc hội tụ chậm.
  - **Avoid Overshooting:** Ngăn chặn việc tỷ lệ học quá lớn, tránh tình trạng vượt quá điểm tối ưu.
  - **Adaptability:** Cung cấp sự điều chỉnh tự động cho mô hình dựa trên tiến triển của quá trình đào tạo.

- **Nhược Điểm:**

  - **Yêu Cầu Điều Chỉnh Thêm Hyperparameters:** Đôi khi cần phải điều chỉnh thêm hyperparameters như tỷ lệ giảm, số epoch giảm, v.v.
  - **Không Phải Là Giải Pháp Tối Ưu Cho Mọi Vấn Đề:** Phương pháp giảm tỷ lệ học không phải luôn là giải pháp tốt cho mọi loại mô hình hoặc tập dữ liệu.

- **Tổng Kết:**Sự Quan Trọng của Learning Rate Decay là một chiến lược quan trọng để cân nhắc trong quá trình đào tạo mô hình, giúp cải thiện hiệu suất và ổn định quá trình học. Tùy thuộc vào bài toán cụ thể, có thể chọn chiến lược giảm tỷ lệ học phù hợp.

1.2 So sánh tổng quan các phương pháp optimizer:

Phương pháp	Ưu điểm	Nhược điểm	Ứng dụng
GD	- Đơn giản và dễ triển khai.	- Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate	Phù hợp với dữ liệu nhỏ, bề mặt lỗi đơn giản.
SGD	- Phù hợp với dữ liệu lớn. - Giảm độ phức tạp tính toán.	- Nhiễu cao do cập nhật sau mỗi ví dụ. - Điểm cực tiểu không ổn định.	Thích hợp cho dữ liệu lớn, đòi hỏi tính toán nhẹ.
MBGD	- Giảm nhiễu so với SGD bằng cách tính toán gradient trên một số lượng lớn các điểm dữ liệu trong mỗi mini-batch	- Yêu cầu lựa chọn kích thước mini-batch phù hợp. Kích thước quá lớn có thể làm tăng độ phức tạp tính toán, trong khi kích thước quá nhỏ có	Phổ biến trong học máy, dữ liệu lớn.

		thể tăng nhiễu và giảm tốc độ hội tụ.	
Adagard	- Hoạt động hiệu quả trên các tập dữ liệu có thưa thớt -Tự điều chỉnh learning rate, Adagrad giảm gánh nặng của việc lựa chọn một learning rate ban đầu	- Cộng dồn bình phương gradient, dẫn đến learning rate giảm quá nhanh, àm chậm quá trình học sau một số lượng lớn các bước cập nhật	Dữ liệu thưa thớt, đa biến thiên.
RMSProp	- Hiệu Quả Trên Các Bề Mặt Lỗi Phức Tạp - Giải quyết vấn đề giảm learning rate quá nhanh khi tích dồn bình phương gradient của Adagard	- RMSProp cần phải điều chỉnh các hyperparameter như hệ số giảm β để đạt được hiệu suất tốt nhất	Bề mặt lỗi có biến thiên, dữ liệu không đồng nhất.

Momentum	- Momentum giúp giảm độ chói của thuật toán, giúp thuật toán vượt qua điểm cực tiểu địa phương và tiếp tục hướng về điểm tối ưu toàn cục - Giúp tăng tốc độ hội tụ, đặc biệt là trên các bề mặt lỗi phức tạp.	- Cần phải chọn giá trị thích hợp cho hệ số momentum - Giá trị quá lớn có thể làm cho thuật toán dao động mạnh và vượt qua điểm tối ưu, trong khi giá trị quá nhỏ có thể làm chậm quá trình học	Đa biến thiên, bề mặt lỗi phức tạp.
Adam	- Sử dụng cả first-order moment (momentum) và second-order moment (độ lớn của gradient)	- Adam cần phải điều chỉnh nhiều hyperparameter như $\beta 1$, $\beta 2$, và ϵ, điều này có thể làm tăng độ phức tạp của quá trình huấn luyện và đôi khi yêu cầu sự thử nghiệm.	Đa dạng các ứng dụng, hiệu suất cao.

Tổng quan: mỗi phương pháp tối ưu hóa có những ưu và nhược điểm riêng, và sự lựa chọn giữa chúng thường phụ thuộc vào đặc tính của dữ liệu và bài toán cụ thể. Adam thường được ưa chuộng trong nhiều trường hợp do khả năng kết hợp của nó giữa momentum và độ lớn của gradient. Bên cạnh đó thì Learning rate decay là một kỹ thuật quan trọng để điều chỉnh tốc độ học của thuật toán,nó ảnh hưởng đến các phương pháp trên trong việc điều chỉnh cẩn thận của các siêu tham số liên quan đến learning rate decay là quan trọng để đạt được hiệu suất tối ưu, đặc biệt là khi mô hình gặp khó khăn trong việc hội tụ hoặc khi đối mặt với biến động lớn trong dữ liệu.

CHƯƠNG 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 khi xây dựng một giải pháp học máy

- Các mô hình học máy truyền thống thường được đào tạo trên các bộ dữ liệu tĩnh và kiến thức của chúng cố định sau khi quá trình đào tạo trước đó hoàn thành. Tuy nhiên, thế giới thực là động và liên tục thay đổi. Học liên tục (Continual Learning) giải quyết nhu cầu của các mô hình học máy phải đối mặt với dữ liệu và nhiệm vụ mới theo thời gian, và là một khái niệm quan trọng trong lĩnh vực Trí Tuệ Nhân Tạo đang phát triển.

- Học liên tục là một mô hình hiện đại trong lĩnh vực học máy, với mục tiêu tạo ra các mô hình có khả năng tăng trưởng và biến đổi liên tục. Khác với các phương pháp học máy truyền thống có kiến thức cố định, học liên tục cho phép mô hình thích ứng theo thời gian, thu thập thông tin và kỹ năng mới mà không xóa đi kinh nghiệm trước đó. Điều này tương tự như cách con người học và xây dựng trên cơ sở kiến thức hiện tại của họ. Thách thức chính mà học liên tục giải quyết là việc quên thảm họa, trong đó các mô hình truyền thống thường mất đi khả năng thực hiện nhiệm vụ đã học khi đối mặt với nhiệm vụ mới. Bằng cách giảm thiểu vấn đề này, học liên tục giúp các hệ thống Trí Tuệ Nhân Tạo duy trì tính hiệu quả và linh hoạt trong một thế giới thay đổi liên tục.

- Ứng dụng thực tế của học liên tục rất đa dạng và mở rộng. Trong lĩnh vực hiểu ngôn ngữ tự nhiên, nó cho phép các chatbot và mô hình ngôn ngữ theo kịp với xu hướng ngôn ngữ và tương tác người dùng đang phát triển, đảm bảo các phản hồi chính xác và phù hợp văn cảnh. Trong lĩnh vực thị giác máy tính, nó giúp các hệ thống nhận

thức thích nghi với các đối tượng, môi trường và tiêu chuẩn hình ảnh mới, tăng tính ổn định và linh hoạt. Hơn nữa, trong lĩnh vực robot tự động, học liên tục trang bị máy móc khả năng học từ kinh nghiệm và thích ứng với các nhiệm vụ và môi trường khác nhau, làm cho chúng trở nên độc lập và linh hoạt hơn trong các ứng dụng thực tế.

2.1.1 Các yếu tố chính trong học liên tục trong học máy

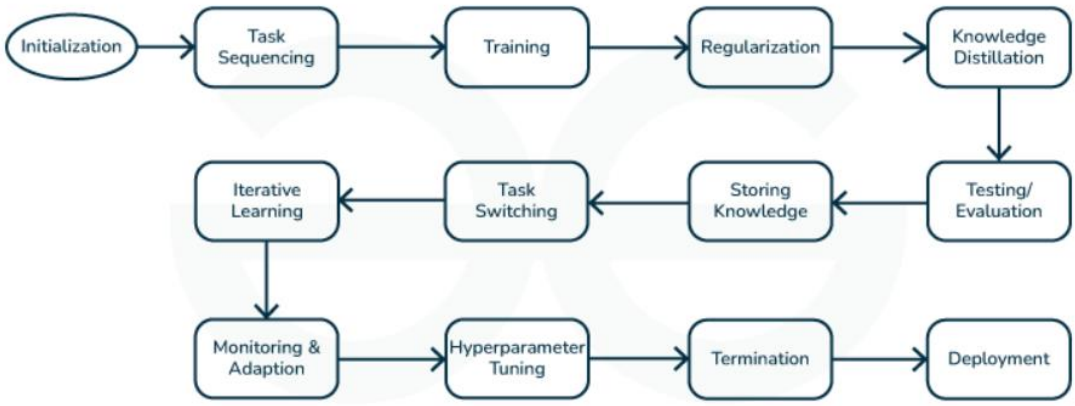
- **Học Bổ Sung (Incremental Learning):** Học liên tục liên quan đến việc đào tạo một mô hình trên dữ liệu mới theo thời gian, thường ở cách tiếp cận tăng dần. Điều này đòi hỏi mô hình phải thích ứng với dữ liệu mới mà không cần đào tạo lại toàn bộ bộ dữ liệu.
- **Bộ Nhớ và Quên (Memory and Forgetting):** Các mô hình trong học liên tục cần cơ chế để nhớ và lưu trữ thông tin quan trọng từ các trải nghiệm trước đó, cũng như cơ chế để tránh việc quên thảm họa, nơi chúng mất hiệu suất trên nhiệm vụ đã học trước đó khi học nhiệm vụ mới.
- **Dãy Nhiệm Vụ (Task Sequences):** Các tình huống học liên tục có thể thay đổi về thứ tự các nhiệm vụ được đặt ra. Một số có thể liên quan đến một thứ tự cố định của các nhiệm vụ, trong khi những cái khác có thể có một thứ tự động hoặc không dự đoán được.
- **Chính Sách và Ổn Định (Regularization and Stabilization):** Các kỹ thuật như Elastic Weight Consolidation (EWC) và synaptic intelligence (SI) được sử dụng để ổn định và bảo dưỡng trọng số mô hình để tránh những thay đổi đột ngột khi học nhiệm vụ mới, giúp duy trì kiến thức trước đó.
- **Tái Tạo và Tái Tạo Trải Nghiệm (Replay and Experience Replay):** Cơ chế tái tạo liên quan đến việc định kỳ quay lại và đào tạo lại trên dữ liệu hoặc trải nghiệm trước đó để cải thiện và củng cố kiến thức đã đạt được trong suốt các nhiệm vụ trước đó.

- **Học Chuyển Giao(Transfer Learning):** Tận dụng kiến thức từ các nhiệm vụ trước để hỗ trợ việc học nhiệm vụ mới là một yếu tố quan trọng của học liên tục. Các kỹ thuật như tái sử dụng đặc trưng và điều chỉnh tinh chỉnh có thể hữu ích.

2.1.2 Các Loại Học Liên Tục

- **Học Theo Nhiệm Vụ (Task-based Continual Learning):** Trong phương pháp này, một mô hình học một chuỗi các nhiệm vụ khác nhau theo thời gian. Mục tiêu của mô hình là thích ứng với mỗi nhiệm vụ mới trong khi vẫn giữ lại kiến thức về các nhiệm vụ đã học trước đó. Các kỹ thuật như Elastic Weight Consolidation (EWC) và Progressive Neural Networks (PNN) thuộc loại này.
- **Học Theo Lớp (Class-incremental Learning):** Học liên tục theo lớp tập trung vào việc quản lý các lớp dữ liệu mới theo thời gian trong khi vẫn giữ lại kiến thức về các lớp đã thấy trước đó. Điều này thường xảy ra trong các ứng dụng như nhận dạng hình ảnh, nơi các lớp đối tượng mới được thêm vào định kỳ. Các phương pháp như iCaRL (Incremental Classifier and Representation Learning) thường được sử dụng trong học liên tục theo lớp.
- **Học Theo Miền (Domain-incremental Learning):** Học liên tục theo miền thích ứng với việc thích ứng với các phân phối hoặc miền dữ liệu mới. Ví dụ, trong robot tự động, robot có thể cần thích ứng với các môi trường khác nhau. Các kỹ thuật cho sự biến đổi miền và học liên tục theo miền được sử dụng để giải quyết tình huống này.

2.1.3 Quá Trình Học Liên Tục



- **Khởi Tạo (Initialization):** Bắt đầu với một phiên bản ban đầu, thường được đào tạo trước trên một bộ dữ liệu lớn để cung cấp kiến thức cơ bản. Phiên bản đã đào tạo trước đó này được sử dụng như một điểm xuất phát cho học liên tục.
- **Xếp Hàng Nhiệm Vụ (Task Sequencing):** Xác định chuỗi nhiệm vụ hoặc luồng dữ liệu mà mô hình sẽ gặp phải. Mỗi nhiệm vụ có thể đại diện cho một thách thức khác nhau, một tập dữ liệu mới hoặc một khía cạnh duy nhất của vấn đề tổng thể.
- **Đào Tạo Trên Một Nhiệm Vụ (Training on a Task):** Đào tạo mô hình trên nhiệm vụ đầu tiên trong chuỗi. Điều này bao gồm cập nhật các tham số của mô hình bằng cách sử dụng dữ liệu cụ thể cho nhiệm vụ hiện tại. Thường sử dụng các kỹ thuật đào tạo tiêu biểu như gradient descent.
- **Chính Sách Bảo Toàn Kiến Thức (Regularization for Knowledge Preservation):** Để ngăn chặn việc quên thảm họa, áp dụng các chiến lược bảo toàn kiến thức. Điều này có thể bao gồm các chiến lược như Elastic Weight Consolidation (EWC) hoặc Synaptic Intelligence (SI) để bảo vệ các tham số quan trọng liên quan đến các nhiệm vụ trước.
- **Truyền Thụ Thống Nhất Kiến Thức (Knowledge Distillation):** Đối với việc học theo lớp hoặc học theo miền, sử dụng phương pháp truyền thụ kiến thức để

chuyển giao thông tin từ phiên bản gốc hoặc mô hình giáo viên sang phiên bản hoặc mô hình hiện tại, giúp nó học được kiến thức của các lớp hoặc miền đã được biết trước.

- **Kiểm Tra và Đánh Giá (Testing and Evaluation):** Sau khi đào tạo trên một nhiệm vụ, so sánh hiệu suất của mô hình trên nhiệm vụ hiện tại để đảm bảo rằng nó đã học đúng. Điều này có thể bao gồm việc sử dụng các phương pháp đánh giá tiêu biểu áp dụng cho nhiệm vụ cụ thể.

- **Lưu Trữ Kiến Thức (Storing Knowledge)**: Tùy thuộc vào phương pháp chọn, bạn có thể lưu trữ thông tin hoặc biểu diễn từ các nhiệm vụ trước đó trong bộ nhớ hoặc bộ đệm. Kiến thức đã lưu trữ này có thể được phát lại hoặc sử dụng để giảm thiểu việc quên khi học nhiệm vụ mới.

- **Chuyển Đổi Nhiệm Vụ (Task Switching)**: Chuyển sang nhiệm vụ tiếp theo trong chuỗi được định sẵn và lặp lại các bước từ 3 đến 7. Mô hình phải thích ứng với dự án mới đồng thời đảm bảo hiệu suất tổng thể không giảm đáng kể trên các nhiệm vụ trước đó.

- **Học Lặp (Iterative Learning)**: Tiếp tục quá trình này lặp lại cho mỗi nhiệm vụ trong chuỗi, duy trì sự cân bằng giữa việc thích ứng với thông tin mới và bảo tồn kiến thức cũ.

- **Theo dõi và Thích Ứng (Monitoring and Adaptation)**: Liên tục theo dõi hiệu suất tổng thể của mô hình và khả năng điều chỉnh. Nếu mô hình có dấu hiệu quên hoặc hiệu suất kém trên các nhiệm vụ trước đó, cân nhắc điều chỉnh các chiến lược chống quên, phát lại, hoặc chưng cất.

- **Điều Chỉnh Tham Số (Hyperparameter Tuning)**: Điều chỉnh các siêu tham số cần thiết để tối ưu hóa sự cân bằng giữa việc thích ứng với nhiệm vụ mới và bảo

tồn kiến thức cũ. Điều này có thể bao gồm việc điều chỉnh tỷ lệ học, độ mạnh của các chiến lược chống quên và các tham số khác.

- **Chấm Dứt hoặc Mở Rộng (Termination or Expansion)**: Xác định điều kiện kết thúc cho quy trình học liên tục, có thể bao gồm một số lượng nhiệm vụ cố định hoặc một phương pháp động cho phép sự biến đổi không giới hạn. Hoặc, mở rộng cấu trúc hoặc khả năng của mô hình để xử lý thêm nhiệm vụ nếu cần thiết.

- **Triển Khai Thực Tế (Real-world Deployment)**: Khi phiên bản đã học từ toàn bộ chuỗi nhiệm vụ, nó có thể được triển khai trong các ứng dụng thực tế, nơi nó có thể thích ứng và tiếp tục học khi gặp phải dữ liệu và nhiệm vụ mới.

2.1.4 Vai trò của Continual Learning trong xây dựng giải pháp học máy

- **Lợi ích của Continual Learning**:

    - **Tính Linh Hoạt** (**Adaptability**): Cho phép mô hình thích ứng và tiến triển theo thời gian, làm cho chúng phù hợp cho các ứng dụng trong môi trường động và thay đổi. Sự linh hoạt này quan trọng trong lĩnh vực như robot tự động và hiểu ngôn ngữ tự nhiên.

    - **Hiệu Quả** (**Efficiency**): Thay vì huấn luyện lại mô hình từ đầu mỗi khi có dữ liệu mới hoặc nhiệm vụ mới xuất hiện, nó cho phép cập nhật theo từng bước, giúp tiết kiệm tài nguyên tính toán và thời gian.

    - **Giữ Kiến Thức** (**Knowledge Retention**): Nó giảm vấn đề của quên thảm họa, cho phép mô hình giữ lại kiến thức từ các nhiệm vụ hoặc kinh nghiệm trước đó. Điều này có giá trị khi đối mặt với việc giữ lại trí nhớ dài hạn trong hệ thống AI.

- **Giảm Lưu Trữ Dữ Liệu** (**Reduced Data Storage**): Các kỹ thuật như tái tạo sáng tạo giảm nhu cầu lưu trữ và quản lý các bộ dữ liệu lịch sử lớn, làm cho việc triển khai học liên tục trở nên khả thi hơn trong các môi trường có tài nguyên hạn chế.

- **Đa Dạng** (**Versatility**): Nó được áp dụng vào nhiều lĩnh vực khác nhau bao gồm xử lý ngôn ngữ tự nhiên, thị giác máy tính, hệ thống gợi ý, làm cho nó trở thành một phương pháp đa năng trong lĩnh vực AI.

- **Hạn Chế và Thách Thức của Continual Learning**:

  - **Catastrophic Forgetting**: Mặc dù đã có những cố gắng để giảm thiểu, mô hình học liên tục vẫn có thể gặp vấn đề quên thảm họa, dẫn đến mất dần hiệu suất trên các nhiệm vụ trước khi học nhiệm vụ mới.

  - **Quá Mức Cùng Cấu Trúc Dữ Liệu Cũ** (**Overfitting to Old Data**): Một số phương pháp học liên tục có thể quá mức cùng cấu trúc với dữ liệu cũ, làm cho mô hình khó khăn khi tổng quát hóa cho nhiệm vụ hoặc lĩnh vực mới.

  - **Phức Tạp** (**Complexity**): Triển khai các kỹ thuật học liên tục có thể phức tạp và yêu cầu điều chỉnh và thiết kế cẩn thận. Sự phức tạp này có thể hạn chế việc áp dụng chúng trong một số ứng dụng.

  - **Khả Năng Mở Rộng** (**Scalability**): Khi mô hình tích lũy thêm kiến thức, khả năng mở rộng có thể trở thành một thách thức. Kích thước và yêu cầu tính toán của mô hình có thể tăng đáng kể theo thời gian.

- **Thay Đổi Phân Phối Dữ Liệu (Data Distribution Shifts)**: Khi nhiệm vụ hoặc lĩnh vực mới có phân phối dữ liệu khác biệt đáng kể so với quá khứ, mô hình học liên tục có thể gặp khó khăn trong việc thích ứng hiệu quả.

- **Cân Bằng Kiến Thức Cũ và Mới (Balancing Old and New Knowledge)**: Tìm sự cân bằng phù hợp giữa kiến thức cũ và mới có thể là một thách thức. Mô hình cần quyết định cái nào nên giữ lại từ các nhiệm vụ cũ trong khi chấp nhận kiến thức mới, và việc tìm ra sự cân bằng lý tưởng có thể khó khăn.

- **Tổng quát :** Học liên tục đóng vai trò quan trọng trong việc thu hẹp khoảng cách giữa các mô hình tĩnh truyền thống và nhu cầu của thông tin đang phát triển và các ứng dụng thực tế. Nó mang lại khả năng phát triển các hệ thống AI có khả năng phân tích và thích ứng theo thời gian, tương tự như quá trình học của con người. Đây là một lĩnh vực nghiên cứu và phát triển hấp dẫn và đầy thách thức trong cộng đồng học máy.

2.2 Test Production khi xây khi xây dựng một giải pháp học máy để giải quyết một bài toán

- Quá (Production Testing) đóng một vai trò quan trọng trong việc đảm bảo tính ổn định và hiệu suất của giải pháp học máy khi triển khai để giải quyết một bài toán cụ thể. Dưới đây là một số khía cạnh cần xem xét khi thực hiện Test Production trong quá trình xây dựng một giải pháp học máy:

- **Thiết Kế Kiểm Thử Sản Xuất:**
    - Xác định kịch bản kiểm thử cụ thể cho mỗi phần của hệ thống, từ dữ liệu đầu vào đến đầu ra của mô hình.

- Thiết kế kịch bản kiểm thử để phản ánh một cách chính xác môi trường sản xuất và các tình huống thực tế.

- **Kiểm Thử Dữ Liệu Đầu Vào:**
  - Xác định liệu dữ liệu đầu vào có thể đáp ứng được đa dạng của dữ liệu thực tế hay không.
  - Kiểm tra xử lý dữ liệu và khả năng xử lý các định dạng dữ liệu mới.
- **Kiểm Thử Độ Nhạy Cảm và An Toàn:**
  - Thực hiện kiểm thử với dữ liệu nhạy cảm để đảm bảo tính riêng tư và tuân thủ các quy định về bảo mật.
  - Kiểm tra cơ chế bảo vệ mô hình khỏi các tấn công như tấn công độc lập, thay đổi dữ liệu đầu vào, và lợi dụng lỗ hổng an ninh.
- **Kiểm Thử Hiệu Suất:**
  - Đánh giá hiệu suất của mô hình dưới tải công việc thực tế.
  - Kiểm thử thời gian đáp ứng và tải trọng để đảm bảo rằng mô hình có thể xử lý một lượng lớn các yêu cầu từ người dùng.
- **Kiểm Thử Sự Nhất Quán và Ổn Định:**
  - Kiểm tra sự nhất quán của mô hình qua các phiên bản khác nhau của dữ liệu đầu vào.
  - Đảm bảo ổn định của mô hình dưới điều kiện biến động của môi trường sản xuất.
- **Kiểm Thử Thay Đổi Mô Hình:**
  - Kiểm thử quá trình triển khai mô hình mới và chuyển đổi giữa các phiên bản mô hình.
  - Đảm bảo rằng việc triển khai mô hình mới không ảnh hưởng đến tính liên tục của dịch vụ.
- **Kiểm Thử Tích Hợp Hệ Thống:**

  - Kiểm tra tích hợp giữa mô hình và các thành phần khác của hệ thống, chẳng hạn như cơ sở dữ liệu, giao diện người dùng, và các dịch vụ bên ngoài.
  - Đảm bảo rằng mô hình là một phần hòa nhập và tương tác mạnh mẽ với hệ thống tổng thể.
- **Kiểm Thử Tuân Thủ Quy Định và Chính Sách:**
  - Đảm bảo rằng giải pháp học máy tuân thủ các quy định pháp luật và chính sách liên quan đến loại dữ liệu và mục tiêu ứng dụng.
- **Kiểm Thử Mô Hình Liên Quan:**
  - Nếu có nhiều mô hình liên quan, kiểm thử chúng trong bối cảnh tương tác với nhau.
  - Xác định và giải quyết các tình huống không mong muốn hoặc mâu thuẫn giữa các mô hình.
- **Kiểm Thử Tự Động và Liên Tục:**
  - Thiết lập hệ thống kiểm thử tự động để giảm thời gian kiểm thử và đảm bảo tính liên tục trong quá trình triển khai và cập nhật mô hình.
- **Kiểm Thử Sự Kiên Nhẫn và Độ Tin Cậy:**
  - Kiểm thử khả năng chịu tải và khả năng phục hồi của hệ thống sau lỗi hoặc gián đoạn.
  - Đảm bảo rằng hệ thống có khả năng tự động khôi phục và duy trì sự kiên nhẫn.
- **Kiểm Thử Kịch Bản Khẩn Cấp:**
  - Thiết lập kịch bản kiểm thử khẩn cấp để đảm bảo sự sẵn sàng và khả năng xử lý vấn đề trong trường hợp sự cố.
- **Tổng quan:** Test Production đòi hỏi một chiến lược toàn diện và cẩn thận để đảm bảo rằng giải pháp học máy có thể hoạt động hiệu quả và đáp ứng được các thách thức thực tế khi triển khai trong môi trường sản xuất.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

Tiếng Anh

2. https://www.linkedin.com/pulse/machine-learning-optimization-techniques-bilal-el-jamal
3. https://madewithml.com/courses/mlops/testing/
4. https://www.geeksforgeeks.org/continual-learning-in-machine-learning/