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

ĐỒ ÁN CUỐI KỲ

Nhập môn trí tuệ nhân tạo

Người hướng dẫn: **Giảng viên Lê Anh Cường**

Người thực hiện: **Trần Thành Đại - 52100876**

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

ĐỒ ÁN CUỐI KỲ

Nhập môn trí tuệ nhân tạo

Người hướng dẫn: Giảng viên Lê Anh Cường
Người thực hiện: **Trần Thành Đại – 52100876**
Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy đã trực tiếp giảng dạy, giúp đỡ chúng em hoàn thành báo cáo trong suốt thời gian học vừa qua.

Quá trình thực hiện báo cáo này là một trong những bước đầu để chúng em đi sâu vào chuyên ngành, tìm hiểu để có thể thiết kế ra một website hoàn chỉnh, trên cơ sở những kiến thức đã được các thầy cô giảng dạy và hướng dẫn. Để có thể hoàn thiện báo cáo này, ngoài sự số gắng, tìm tòi, học hỏi, giúp đỡ nhau của các thành viên trong nhóm chúng em, thì còn là sự giúp đỡ, giảng dạy tận tình của các thầy cô khác trong khoa trong quá trình chúng em học tập tại trường đã giúp chúng em có đủ kiến thức, khả năng để thực hiện báo cáo môn học này. Với lòng biết ơn sâu sắc nhất, nhóm chúng em xin được gửi lời cảm ơn chân thành đến các thầy cô trong khoa, đặc biệt là thầy .._ người đã nhiệt tình hướng dẫn nhóm chúng em trong quá trình thực hiện đề tài này.

Vì thời gian và trình độ của chúng em còn hạn chế, nên chắc chắn chúng em sẽ không tránh khỏi những thiếu sót trong bài báo cáo này. Chúng em rất mong nhận được những ý kiến góp ý của các thầy cô để có thể hoàn thiện hơn ở những báo cáo của các môn học khác trong thời gian sắp tới.

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 2023

Tác giả

(ký tên và ghi rõ họ tên)

Đại

Trần Thành Đại

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)

Yêu cầu:

Bài 1 (3 điểm): làm riêng từng người

Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

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

Nộp file .doc hoặc .pdf.
Trình bày trên cả Github

Bài 2: (7 điểm): làm chung trong nhóm

Đưa ra một bài toán dự đoán có thể giải quyết bằng học máy (machine learning) với các yêu cầu sau:

- Số Feature/Attribute gồm nhiều kiểu: categorical và numerical;
- Dữ liệu phải chưa được học, thực tập trên lớp và trong bài tập về nhà;

1) Phân tích thống kê trên dữ liệu, vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu. Tìm hiểu các đặc trưng và đánh gía vai trò của các đặc trưng đối với mục tiêu bài toán;
2) Ứng dụng các mô hình học máy cơ bản để giải quyết bài toán, bao gồm cả các mô hình thuộc Ensemble Learning;
3) Sử dụng Feedforward Neural Network và Recurrent Neural Network (hoặc mô thuộc loại này) để giải quyết bài toán;
4) Áp dụng các kỹ thuật tránh Overfitting trên các mô hình của câu (2) và câu (3) để giải quyết bài toán;
5) Sau khi huấn luyện xong mô hình thì muốn cải thiện độ chính xác, ta sẽ làm gì để giải quyết nó? Phân tích các trường hợp sai, đề ra giải pháp và thực hiện nó, sau đó đánh giá xem có cải tiến so với trước không.

Lưu ý: Tạo project trên Github để làm bài này, chứa code và data;
Nộp file .ipynp và data. Trong file .ipynp ghi rõ tên các thành viên trong nhóm.

I. Optimizer trong Machine Learning

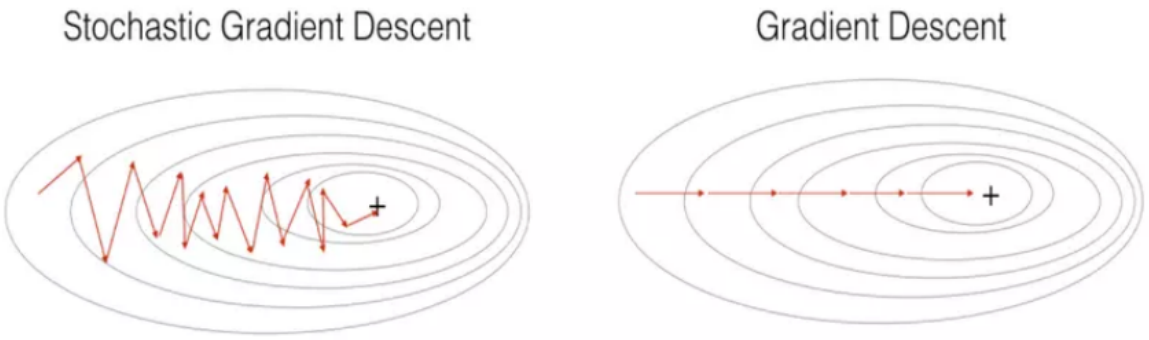
A. Khái niệm: Optimizer là một thuật toán hoặc một hàm có khả năng thích ứng với các thuộc tính của neural network, như learning rate (tốc độ học) và weights (trọng số). Từ đó hỗ trợ cải thiện độ chính xác và giảm thiểu mất mác trong quá trình học.

B. Lựa chọn một optimizer: Lựa chọn một optimizer không phải là một vấn đề đơn giản, vì optimizer ảnh hưởng trực tiếp đến các thông số và trọng số của một mô hình ML. Phải chú ý chọn lựa sao cho phù hợp.

C. Các Optimizer phổ biến:

1. Gradient Descent (GD): Đây là một trong những optimizer cơ bản nhất, được sử dụng trực tiếp từ đạo hàm của loss function (hàm mất mác) và learning rate để giảm mất mác và đạt được cực tiểu lân cận.
   - Công thức: $\theta=\theta-\alpha\cdot\nabla J(\theta)$
     - $\boldsymbol{\theta}$: là thông số cần optimize
     - $\boldsymbol{A}$: là learning rate
     - $\nabla \boldsymbol{J(\theta)}$: là thông số dốc của hàm mất mát, là một vector chỉ tới hướng tăng dốc nhất của thuật toán ở điểm hiện tại.
   - Ưu điểm:
     - Thuật toán GD cơ bản, dễ hiểu. Giải quyết vấn đề tối ưu neural network bằng cách cập nhật trọng số sau mỗi vòng lặp.
   - Nhược điểm:
     - Vì đơn giản nên GD còn nhiều hạn chế như phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.
     - Tốc độ học quá lớn sẽ khiến cho thuật toán không hội tụ, quanh quẩn bên đích vì bước nhảy quá lớn.
2. Stochastic Gradient Descent (SGD): Là một biến thể của Gradient Descent. Thay vì sau mỗi epoch cập nhật trọng số 1 lần, thì nếu có N điểm dữ liệu trong epoch thì ta phải cập nhật trọng số N lần. SGD sẽ giảm tốc độ của 1 epoch, nhưng sẽ hội tụ rất nhanh.

- Công thức: tương tự như Gradient Descent, nhưng lặp lại trên mỗi điểm dữ liệu.



- Nhìn hình ảnh minh họa phía trên, ta có thể thấy đường minh họa cho SGD có hình zíc zắc, độ phức tạp cao hơn nhiều so với Gradient Descent. Vậy tại sao phải áp dụng SGD?
    - Vì SGD hỗ trợ cho những thuật toán học real time. Giả sử như ta có một bộ dữ liệu với các điểm dữ liệu được thêm vào một cách riêng lẻ (cơ sở dữ liệu thông tin mua sắm, tài khoản người dùng…). Với những bộ dữ liệu này, nếu qua mỗi lần cập nhật điểm dữ liệu, lại phải cập nhật lại đạo hàm của toàn bộ cơ sở dữ liệu sẽ ảnh hưởng nhiều đến tốc độ cũng như lãng phí tài nguyên tính toán.
    - SGD ra đời nhằm để khắc phục tình trạng này. Bằng cách tính lại đạo hàm trên mỗi điểm dữ liệu, sẽ tiết kiệm được rất nhiều thời gian, chi phí khi áp dụng vào những bộ dữ liệu lớn.
- Ưu điểm: Thuật toán giải quyết được với cơ sở dữ liệu lớn mà Gradient Descent không làm được.
- Nhược điểm:
    - Thuật toán vẫn chưa giải quyết được 2 nhược điểm lớn của gradient descent (learning rate, điểm dữ liệu ban đầu) và xử lý triệt để các update điểm dữ liệu nhiễu. Vì vậy ta phải kết hợp SGD với 1 số thuật toán khác như: Momentum, AdaGrad và RMSprop.

3. Momentum: Để khắc phục những hạn chế cơ bản của Gradient Descent, chúng ta thường dùng Gradient Descent với biến số Momentum (động lượng).
    - Trong Gradient Descent thông thường, sau mỗi lần lặp, thông số được cập nhật phụ thuộc hoàn toàn vào đạo hàm (gradient) hiện tại của hàm mất mát. Ngược lại, Gradient Descent với Momentum bao gồm một giá trị trung bình di động của gradient quá khứ để xác định cập nhật hiện tại. Quy tắc cặp nhật θ ở lần lặp t được cho bởi công thức sau:
    - $$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot \nabla J(\theta_t)$$
      $$\theta_{t+1} = \theta_t - \alpha \cdot v_t$$
    - Trong đó:
        - ∇***J(θt)*** là độ dốc của hàm mất mát theo các tham số θt.
        - ***α*** là tỷ lệ học, quyết định kích thước bước của cập nhật.
        - ***β*** là biến động lượng, thường được đặt giá trị trong khoảng từ 0 đến 1.
        - Biến ***vt*** là biến động lượng, là trung bình động của các thông số gradient trong quá khứ. Nó sẽ tích lũy các gradient theo hướng chuyển động hiện tại và thực hiện cập nhật theo thông số này. Mục tiêu là nhằm đẩy thuật toán qua khỏi các biến nhiễu, đẩy thuật toán di động trơn tru hơn đến cực tiểu của hàm mất mát.
    - Ưu điểm:
        - **Hội tụ Nhanh Chóng:** Thuật ngữ động lượng giúp thuật toán tối ưu hóa "giữ lại" một số vận tốc từ các lần lặp trước đó, cho phép nó đi qua các khu vực phẳng và các điểm tối thiểu cục bộ nhỏ nhanh chóng hơn.
        - **Giảm Dao Động:** Thuật ngữ động lượng giúp giảm độ dao động và cho phép hội tụ mượt mà

hơn, giảm nguy cơ bị kẹt trong hành vi dao động.
        - **Thuật toán tối ưu giải quyết được vấn đề:** Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.
    - Nhược điểm:
        - Tuy momentum giúp thuật toán vượt dốc tiến tới điểm đích, tuy nhiên khi tới gần đích, nó vẫn mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn, điều này được giải thích thuật toán tốn thời gian để "mất đà" (biến momentum cần thời gian để về 0)
    - Lựa chọn của thuật ngữ động lượng ***(β)*** là quan trọng. Một giá trị phổ biến là khoảng 0.9, nhưng giá trị tối ưu có thể phụ thuộc vào vấn đề và bộ dữ liệu cụ thể.
    - Trong thực tế, gradient descent với động lượng được sử dụng rộng rãi và là một thành phần của các thuật toán tối ưu hóa nâng cao hơn như trình tối ưu Adam, kết hợp động lượng với tỷ lệ học thích ứng.
4. Adagrad (Adaptive Gradient Descent) là một giải thuật tối ưu hóa được áp dụng vào machine learning và deep learning. Adagrad điều chỉnh learning rate của model dựa vào lịch sử của các gradient (khác với các thuật toán trước, learning rate là hằng số). Mục tiêu chính là cung cấp learning rate nhỏ cho các tham số đã nhận những cập nhật lớn trong quá khứ và tỷ lệ học lớn cho các tham số có cập nhật nhỏ.
    - Quy tắc cập nhật Adagrad cho một tham số ***θ*** ở lần lặp ***t*** được cho bởi:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla J(\theta_t)$$

        - Trong đó:
            - ∇ ***J(θt)*** là gradient của hàm mất mát theo các tham số ***θt***.
            - ***η*** là tỷ lệ học.

- ***Gt*** là một ma trận đường chéo trong đó mỗi phần tử đường chéo GiiGii là tổng của bình phương các gradient trong quá khứ cho tham số ***θi***.
- ***ϵ*** là một hằng số nhỏ (thường được thêm để đảm bảo tính ổn định số học để tránh chia cho số không).

- Nhược điểm của Adagrad là learning rate của mỗi tham số có thể sẽ trở nên rất nhỏ theo thời gian, làm mô hình không thể tiếp tục học.
- Trong thực tế, Adagrad thường đóng vai trò như một hàm điều chỉnh trong các thuật toán tối ưu hóa cao hơn. Ví dụ như việc sử dụng tổng bình phương gradient đã tích lũy trong mẫu số có thể thấy trong các trình tối ưu hóa khác như RMSprop và Adam. Các trình tối ưu hóa này giải quyết một số hạn chế của Adagrad và cung cấp hiệu suất tốt hơn trong một số tình huống cụ thể.

5. RMSprop (Root Mean Square Propagation): là một biến thể của thuật toán gradient descent được thiết kế để làm giảm ảnh hưởng của gradient lớn hoặc nhỏ lên quá trình cập nhật tham số.
    - Công thức cập nhật cho RMSprop của một tham số ***θ*** tại lần lặp ***t*** là:

$$E[g^2]_t = \beta \cdot E[g^2]_{t-1} + (1 - \beta) \cdot (\nabla J(\theta_t))^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot \nabla J(\theta_t)$$

    - Trong đó:
        - ∇***J(θt)*** là gradient của hàm mất mát theo các tham số ***θt***.
        - ***η*** là tỷ lệ học (learning rate).

- ***β*** là một hệ số giảm dần gradient với giá trị thường được đặt gần 1, chẳng hạn như 0.9.
- ***E[g^2]t*** là một trung bình động bình phương của gradient, nơi ***g*** là gradient và ***ϵ*** là một hằng số nhỏ để tránh chia cho số không.

- Ý tưởng chính của RMSprop là thay vì sử dụng toàn bộ lịch sử gradient như trong Adagrad, nó chỉ sử dụng trung bình động bình phương của gradient. Điều này giúp giảm hiệu ứng của các gradient lớn, giúp mô hình hội tụ mạnh mẽ hơn và giảm nguy cơ quá trình học bị chậm lại.
- RMSprop thường được sử dụng trong các mô hình học máy và mạng nơ-ron sâu như một lựa chọn tối ưu hóa hiệu quả. Nó cũng là một thành phần của nhiều thuật toán tối ưu hóa phức tạp hơn như Adam.

6. Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa thường kết hợp cả ý tưởng của gradient descent with momentum và RMSprop để tạo ra một phương pháp tối ưu hóa hiệu quả.
    - Công thức cập nhật cho Adam cho một tham số ***θ*** tại lần lặp ***t*** là:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla J(\theta_t) \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla J(\theta_t))^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t
\end{aligned}
$$

Dưới đây là các thành phần chính của thuật toán Adam:

1. **Moment Estimates:**

- ***mt***: First moment estimate (momentum term) - dùng để giữ lại thông tin về độ dốc của hàm mất mát.
- ***vt***: Second moment estimate (squared gradient term) - dùng để giữ lại thông tin về biến động của độ dốc.

2. **Exponential Moving Averages:**

$$
\begin{aligned}
&\bullet\ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla J(\theta_t) \\
&\bullet\ v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla J(\theta_t))^2 \\
&\bullet\ \hat{m}_t = \frac{m_t}{1-\beta_1^t} \\
&\bullet\ \hat{v}_t = \frac{v_t}{1-\beta_2^t}
\end{aligned}
$$

Trong đó:

- ***β1*** và ***β2*** là các hệ số giảm dần gradient và gradient bình phương, thường được đặt gần 1.
- ∇***J(θt)*** là gradient của hàm mất mát theo các tham số ***θt***.
- ***t*** là số lần lặp.

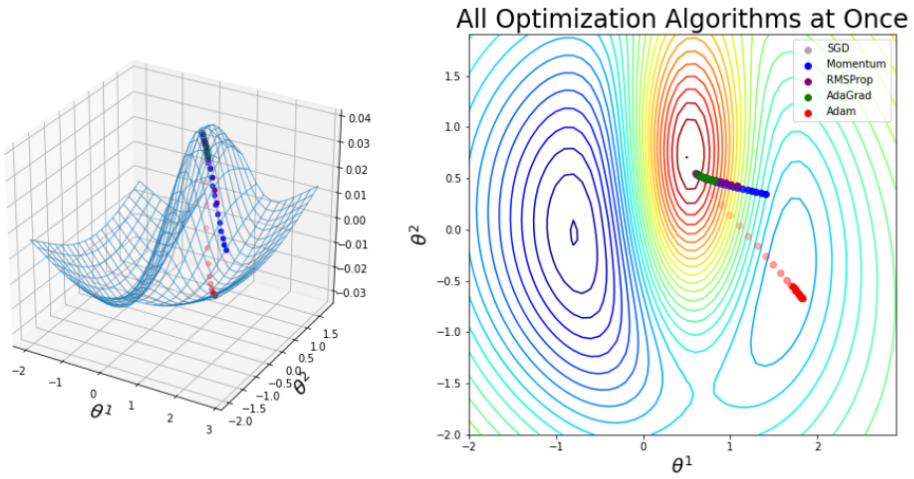
3. **Update Rule:**

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

Trong đó:

- ***η*** là tỷ lệ học (learning rate).
- ***ϵ*** là một hằng số nhỏ để tránh chia cho số không.
    - Adam có những ưu điểm như khả năng tự điều chỉnh tỷ lệ học và xử lý hiệu quả các vấn đề như gradient thưa (sparse gradients). Nó thường được ưa chuộng trong cộng đồng machine learning vì khả năng làm giảm thiểu vấn đề của các thuật toán tối ưu hóa khác và có khả năng hội tụ tốt trên nhiều loại dữ liệu và mô hình.

- Tổng quan: Như đã trình bày ở trên, có rất nhiều loại optimizer algorithm cho các model machine learning, tùy thuộc vào nhu cầu và kiểu dữ liệu mà người lập trình viên phải lựa chọn và tinh chỉnh sao cho phù hợp với model của mình. Trong số các optimizer kể trên thì Adam là phổ biến nhất trong cộng đồng machine learning. Tuy hơi phức tạp về mặt lý thuyết, Adam đơn giản về mặt thực hành. Không cần tốn nhiều thời gian tinh chỉnh, tính toán một cách hiệu quả - ít tốn tài nguyên và sử dụng bộ nhớ một cách tương đối tối ưu.



Ta thấy Adam vượt mặt các optimizer khác để đến được cực tiểu một cách nhanh chóng hơn, hiệu quả hơn. Vậy nên, nếu người lập trình viên muốn giải quyết một bài toán nhanh chóng để benchmark dataset, hay còn mới mẻ với machine learning, Adam optimizer nên là lựa chọn đầu tiên.

II. Continual Learning và Test Production

1. Continual Learning (học liên tục) hay còn được gọi là Incremental Learning (học cộng dồn) hoặc Lifelong Learning (học vòng đời) là một phương pháp trong machine learning mà khi đó, mô hình học từ một bộ dữ liệu liên tục, được cập nhật theo thời gian mà không cần phải giữ lại dữ liệu cũ (record) hay huấn luyện lại từ đầu. Tuy nhiên, mô hình không bị ràng buộc chỉ học duy nhất một bộ dữ liệu cụ thể mà còn có thể tích hợp và thích ứng với dữ liệu mới qua thời gian.
    - Một số vấn đề của Continual Learning: Vì mô hình được huấn luyện trên dữ liệu cũ trước, khi đưa dữ liệu mới vào, có khả năng mô hình sẽ quên đi dữ liệu cũ. Từ đó ta có một số vấn đề:

        - **Cô lập cũ và mới (Catastrophic Learning):** Khi mô hình học dữ liệu mới, có khả năng quên đi thông tin liên quan đến dữ liệu cũ.
        - **Mã hóa kiến thức cũ:** Làm cách nào để mô hình có thể "nhớ" được kiến thức từ dữ liệu cũ khi chuyển sang học dữ liệu mới.
        - **Quản lý tài nguyên:** Khi dữ liệu ngày càng lớn hơn và đa dạng hơn, nhu cầu sử dụng tài nguyên (bộ nhớ, năng lượng xử lý) càng nặng nề hơn, làm cách nào để giải quyết và quản lý tài nguyên một cách tối ưu nhất.
    - Một số giải pháp được áp dụng để giải quyết những vấn đề nêu trên bao gồm:
        - **Regularization Techniques:** Điều chỉnh các phương pháp học để giảm thiểu hiện tượng quên các thông tin cũ.
        - **Memory-Augmented Networks**: Sử dụng bộ nhớ bổ sung để lưu trữ thông tin quan trọng từ dữ liệu cũ.
        - **Relay Mechanism:** Sử dụng lại dữ liệu cũ trong quá trình huấn luyện dữ liệu mới để mô hình không quên.
        - **Dynamic Architecture:** Tự điều chỉnh cấu trúc mô hình trong quá trình huấn luyện dữ liệu mới để ứng với sự đa dạng của dữ liệu.
    - Continual learning đặt ra nhiều thách thức và vẫn là một lĩnh vực nghiên cứu tích cực để phát triển các phương pháp và mô hình hiệu quả. Nó có ứng dụng quan trọng trong các hệ thống AI và robot tự động hoá nơi mà mô hình cần liên tục học từ môi trường và tương tác với dữ liệu mới.
2. Test Production: Trong lĩnh vực machine learning, "test production" thường được hiểu là quá trình kiểm thử mô hình machine learning khi nó đã được triển khai và đang chạy trong môi trường sản xuất thực tế. Điều này nhấn mạnh việc đảm bảo rằng mô hình hoạt động hiệu quả, đáp ứng đúng đắn với dữ liệu thực tế và không gặp vấn đề không mong muốn trong quá trình triển khai.
    - Dưới đây là một số khía cạnh quan trọng khi thực hiện "test production" trong machine learning:
        1. **Độ Chính Xác (Accuracy):** Kiểm tra xem mô hình có độ chính xác cao và hiệu suất tốt trên dữ liệu thực tế hay không.

Các yếu tố như sự thay đổi trong phân phối dữ liệu có thể ảnh hưởng đến độ chính xác.

2. **Hiệu suất và Thời gian đáp ứng:** Đảm bảo rằng mô hình có thể đáp ứng kịp thời và không gây ra độ trễ không mong muốn trong môi trường sản xuất.
3. **Quản lý Dữ liệu Mới:** Kiểm tra khả năng của mô hình khi phải xử lý và dự đoán trên dữ liệu mới, đặc biệt là khi dữ liệu thay đổi theo thời gian.
4. **Ổn định và Điều Chỉnh (Robustness and Adaptability):** Xác minh xem mô hình có ổn định và có khả năng thích ứng với sự biến động của môi trường không.
5. **Giảm Tác Động (Mitigating Impact):** Đối mặt với vấn đề không mong muốn hoặc tác động của mô hình đối với hệ thống sản xuất và tìm cách giảm thiểu chúng.
6. **Bảo mật và Quản lý Rủi ro:** Kiểm tra bảo mật của mô hình và quá trình triển khai để đảm bảo không có lỗ hổng nào có thể được tận dụng.
7. **Giữ Nguyên Trạng Thái Đối Tượng (Maintaining Statefulness):** Đối với các mô hình yêu cầu giữ nguyên trạng thái đối tượng, đảm bảo rằng thông tin trạng thái được duy trì chính xác và đáng tin cậy.

Test production trong machine learning là một bước quan trọng để đảm bảo rằng mô hình không chỉ hoạt động tốt trong các điều kiện kiểm thử mà còn đáp ứng đúng đắn và hiệu quả trong môi trường sản xuất thực tế.

.