TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

BÁO CÁO CUỐI KÌ

MÔN HỌC: NHẬP MÔN HỌC MÁY

Người hướng dẫn: **LÊ ANH CƯỜNG**

Người thực hiện: **HƯỜNG QUANG HUY - 52100893**

Lớp **: 21050301**

Khoá **: 25**

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

BÁO CÁO CUỐI KÌ

MÔN HỌC: NHẬP MÔN HỌC MÁY

Người hướng dẫn: **LÊ ANH CƯỜNG**

Người thực hiện: **HƯỜNG QUANG HUY - 52100893**

Lớp **: 21050301**

Khoá **: 25**

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2023

LỜI CẢM ƠN

Đầu tiên em xin chân thành cảm thầy Lê Anh Cường là người dạy trực tiếp bộ môn Nhập môn Học máy. Thầy đã tận tâm trong suốt quá trình dạy môn học này.

Chúng em cũng xin chân thành cảm ơn toàn thể giáo viên khoa Công nghệ thông tin của Trường Đại học Tôn Đức Thắng đã ủng hộ để hoàn thiện bài báo cáo này.

Cuối cùng lời cảm ơn này chúng em xin dành đến gia đình và những người bạn trong nhóm đã cùng nhau học tập, đồng hành, giúp đỡ lẫn nhau trong suốt thời gian hoàn thành bài báo cáo này.

Bài báo cáo này là minh chứng rõ nhất cho những cố gắng, nỗ lực học hỏi của nhóm em, tuy nhiên cũng không tránh khỏi thiếu sót. Mong rằng chúng em sẽ nhận được sự đóng góp, chỉ bảo từ các thầy cô để cho chúng em có thể sửa đổi, bổ sung và hoàn thành tốt hơn nữa.

Chúng em rất mong sẽ nhận được sự hài lòng, giúp đỡ của các thầy cô. Một lần nữa nhóm em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 19 tháng 03 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

BÁO CÁO ĐƯỢC HOÀN THÀNH

TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng của nhóm tôi 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 bài Báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Bài tập lớn của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 19 tháng 03 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____
_____
_____
_____
_____
_____
_____

Tp. Hồ Chí Minh, ngày tháng năm 2023
(Kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____
_____
_____
_____
_____
_____
_____

Tp. Hồ Chí Minh, ngày tháng năm 2023
(Kí và ghi họ tên)

TÓM TẮT

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.

MỤC LỤC

DANH MỤC CÁC CHỮ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
kNN	k-Nearest Neighbors (k láng giềng gần nhất)
ML	Machine Learning (Học máy)
RD	Random Forest (Rừng ngẫu nhiên)
DT	Decision Tree (Cây quyết định)

DANH SÁCH CÁC BẢNG BIỂU, HÌNH VẼ

DANH MỤC HÌNH

CHƯƠNG I: TRẢ LỜI CÂU 1

I. Tìm hiểu về phương pháp Optimizer

1. Khái niệm

Thuật toán tối ưu (**optimizers)** là một thuật ngữ chung để chỉ một phương pháp hoặc thuật toán được sử dụng để cải thiện hoặc "tối ưu hóa" một số khía cạnh của chương trình hoặc mô hình học máy , là cơ sở để xây dựng mô hình neural network với mục đích "học " được các features (hay pattern) của dữ liệu đầu vào, từ đó có thể tìm 1 cặp weights và bias phù hợp để tối ưu hóa model .

2. Ứng dụng chính của optimizers

*Có hai ứng dụng chính của optimizers:

1. **Trong Lập Trình và Hệ Thống Phần Mềm:**
    - **Tối ưu hóa hiệu suất:** Optimizers ở đây có thể giúp cải thiện tốc độ thực thi của chương trình, giảm bộ nhớ sử dụng, hoặc tối ưu hóa các tài nguyên hệ thống khác.
    - **Tối ưu hóa truy vấn:** Trong cơ sở dữ liệu, optimizer giúp xác định cách hiệu quả nhất để thực hiện một truy vấn.
2. **Trong Học Máy và Trí Tuệ Nhân Tạo:**
    - **Tối ưu hóa thuật toán:** Đây là quá trình điều chỉnh các tham số của mô hình để giảm thiểu một hàm mất mát (loss function). Hàm mất mát đo lường sự chênh lệch giữa dự đoán của mô hình và giá trị thực tế.
    - **Ví dụ của Optimizers trong Học Máy:** Gradient Descent, Stochastic Gradient Descent (SGD), Adam, và RMSprop. Chúng giúp tối ưu hóa mô hình học máy bằng cách cập nhật trọng số của mô hình để cải thiện độ chính xác trong dự đoán.

.

3. Lợi ích của việc dùng Optimizer trong học máy

- **Hiệu quả tài nguyên:** Trong lập trình, optimizer giúp chương trình chạy nhanh hơn và sử dụng ít tài nguyên hơn.
- **Tối ưu hóa hiệu suất mô hình:** Trong học máy, sử dụng optimizer giúp mô hình học nhanh hơn và chính xác hơn, đồng thời tránh vấn đề như overfitting hoặc underfitting.
- **Tìm kiếm cục bộ và toàn cục:** Optimizer giúp tìm kiếm cả giải pháp tối ưu cục bộ lẫn toàn cục trong không gian tham số.

Vì vậy, việc sử dụng optimizer là rất quan trọng trong cả việc phát triển phần mềm lẫn xây dựng và tinh chỉnh các mô hình học máy.

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

1. Tìm hiểu các phương pháp Optimizer trong huấn luyện mô hình học máy

1.1. Tìm hiểu về phương pháp Gradient Descent

Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó. Ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering. Nhìn chung, việc tìm global minimum của các hàm mất mát trong Machine Learning là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Phương pháp Gradient Descent (GD) là một trong những kỹ thuật tối ưu hóa cơ bản và phổ biến nhất trong lĩnh vực học máy và học sâu. Mục đích của Gradient Descent là tối thiểu hóa hàm mất mát (loss function) của một mô hình bằng cách tìm ra bộ tham số (weights) tối ưu

1.1.1. Nguyên lí hoạt động của Gradient Descent

- Gradient: Trong toán học, gradient của một hàm tại một điểm cụ thể cho biết hướng và tốc độ tăng nhanh nhất của hàm đó. Trong GD, gradient

được sử dụng để xác định hướng cần điều chỉnh tham số để giảm thiểu hàm mất mát.

- Cập nhật Tham số: GD cập nhật tham số của mô hình theo hướng ngược lại với gradient để giảm dần giá trị của hàm mất mát.

1.1.2. Công Thức Cập Nhật:

Tham số được cập nhật theo công thức:

$$\theta = \theta - n.\nabla_{\theta} J(\theta)$$

Trong đó : θ là tham số cần tối ưu.

n là tốc độ học (learning rate).

$\nabla_{\theta} J(\theta)$ là gradient của hàm mất mát J theo tham số θ

1.1.3. Tốc độ học

- Tốc độ học n quyết định mức độ lớn của bước đi trong không gian tham số. Nếu n quá lớn, thuật toán có thể vượt qua điểm tối ưu; nếu quá nhỏ, quá trình hội tụ sẽ chậm.
- Điều Chỉnh: Việc chọn tốc độ học thích hợp là rất quan trọng và thường cần thực hiện thử nghiệm để tìm ra giá trị tối ưu.

1.1.4. Phân lại Gradient Descent:

- Có 3 loại

- Batch Gradient Descent: Tính toán gradient trên toàn bộ tập dữ liệu. Đảm bảo hướng đi chính xác nhưng tốn nhiều tài nguyên và thời gian.

- Stochastic Gradient Descent (SGD): Tính toán gradient sau mỗi mẫu, giúp tăng tốc độ học nhưng kết quả có thể không ổn định.

- Mini-batch Gradient Descent: Kết hợp giữa hai phương pháp trên, tính toán gradient trên từng nhóm nhỏ các mẫu.

1.1.5. Ứng dụng

- Được sử dụng rộng rãi trong các mô hình học máy, từ hồi quy tuyến tính đơn giản đến các mạng nơ-ron phức tạp.

1.2. Tìm hiểu về phương pháp Stochastic Gradient Descent (SGD):

Gradient Descent, có một thuật ngữ gọi là “batch”, biểu thị tổng số mẫu từ tập dữ liệu được sử dụng để tính toán gradient cho mỗi lần lặp. Trong tối ưu hóa Gradient Descent điển hình, như Batch Gradient Descent, lô được coi là toàn bộ tập dữ liệu. Mặc dù, việc sử dụng toàn bộ tập dữ liệu thực sự hữu ích để đi đến cực tiểu theo cách ít ồn ào hơn và ít ngẫu nhiên hơn, nhưng vấn đề nảy sinh khi tập dữ liệu của chúng ta lớn hơn. Giả sử, bạn có một triệu mẫu trong tập dữ liệu của mình, vì vậy nếu bạn sử dụng kỹ thuật tối ưu hóa Gradient Descent điển hình, bạn sẽ phải sử dụng tất cả một triệu mẫu để hoàn thành một lần lặp trong khi thực hiện Gradient Descent, và nó phải được thực hiện cho mỗi lần lặp cho đến khi đạt đến cực tiểu. Do đó, nó trở nên rất tốn kém về mặt tính toán để thực hiện

- Vấn đề này được giải quyết bằng Stochastic Gradient Descent. Trong SGD, nó chỉ sử dụng một mẫu duy nhất, tức là kích thước lô của một mẫu, để thực hiện mỗi lần lặp. Mẫu được xáo trộn ngẫu nhiên và được chọn để thực hiện lặp lại.

1.2.1. Cách Hoạt Động của Stochastic Gradient Descent (SGD):

1. - **Chọn Ngẫu Nhiên:** Trong mỗi lần lặp, SGD chọn một mẫu dữ liệu (hoặc một mini-batch nhỏ) một cách ngẫu nhiên từ tập dữ liệu huấn luyện.
2. **Tính Gradient:** Tính gradient của hàm mất mát liên quan đến trọng số mô hình, nhưng chỉ dựa trên mẫu dữ liệu được chọn.
3. **Cập Nhật Trọng Số:** Cập nhật trọng số của mô hình theo hướng làm giảm gradient của hàm mất mát

.

1.2.2. Thuật toán Stochastic Gradient Descent (SGD) :

Ta có thuật toán Stochastic Gradient Descent (SGD)

$$for\ i\ in\ range\ (m):$$

$$\theta_j = \theta_j - \alpha\,(\hat{y}^i - y^i)\,x_j^i$$

-Trong SGD, chúng ta tìm ra gradient của chi phí hàm của một ví dụ duy nhất tại mỗi lần lặp thay vì tổng gradient của hàm chi phí của tất cả các ví dụ.

Trong SGD, vì chỉ có một mẫu từ tập dữ liệu được chọn ngẫu nhiên cho mỗi lần lặp, nên đường dẫn mà thuật toán thực hiện để đến cực tiểu thường ồn ào hơn so với thuật toán Gradient Descent điển hình của bạn. Nhưng điều đó không quan trọng lắm vì đường đi của thuật toán không quan trọng, miễn là chúng ta đạt đến cực tiểu và với thời gian đào tạo ngắn hơn đáng kể.

1.3. Tìm hiểu về phương pháp Momentum:

Phương pháp Momentum là một cải tiến của Gradient Descent trong lĩnh vực học máy và học sâu. Nó giúp tăng tốc độ hội tụ của thuật toán và giảm thiểu vấn đề dao động.

1.3.1. Nguyên lí cơ bản của Momentum

- **Ý Tưởng:** Momentum kết hợp ý tưởng vật lý về động lượng vào quá trình cập nhật tham số. Nó tính toán "đà" của các bước cập nhật, giúp "đẩy" cập nhật theo hướng đúng và qua các điểm rãnh hẹp.

- Mục Đích: Giảm thiểu dao động và tăng tốc độ hội tụ, đặc biệt hữu ích trong các bề mặt mất mát có dạng rãnh hẹp.

1.3.2. Công Thức Cập Nhật:

Tham số được cập nhật theo công thức:

$$v_t = \gamma\, v_{t-1} + n\, \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

Trong đó : -θ là tham số cần tối ưu.

- v_t là động lực tại thời điểm t
- γ là hệ số động lượng, thường lựa chọn trong khoảng 0.90.9 đến 0.990.99.
- n là tốc độ học (learning rate).
- $\nabla_\theta J(\theta)$ là gradient của hàm mất mát J theo tham số θ

1.3.3. Hệ số động lượng

- Vai Trò: Hệ số động lượng giúp xác định mức độ "động lượng" từ các bước cập nhật trước đó. Một giá trị cao của γ giúp tích lũy nhiều động lượng hơn, làm cho cập nhật trở nên mượt mà và ổn định hơn.
- Chọn Lựa: Cần thực hiện thử nghiệm để tìm ra giá trị tối ưu, tùy thuộc vào dữ liệu và mô hình.

1.3.4. Ứng dụng của Momentum

- Momentum thường được sử dụng trong huấn luyện các mạng nơ-ron sâu, nơi mà bề mặt mất mát thường rất phức tạp và có nhiều điểm cực tiểu địa phương.

1.3.5. Biến thể của Momentum

- Nesterov Accelerated Gradient (NAG): Một biến thể của Momentum, NAG trước hết "nhìn trước" để cập nhật gradient (dựa trên vị trí cập nhật sắp tới), sau đó mới áp dụng động lượng. Điều này giúp NAG phản ứng nhanh hơn và thường hiệu quả hơn so với Momentum truyền thống.

1.4. Tìm hiểu về phương pháp Mini-batch Gradient Descent

- Mini-batch Gradient Descent là một biến thể của Gradient Descent, thường được sử dụng trong huấn luyện mô hình học máy và học sâu. Nó kết hợp những ưu điểm của cả Batch Gradient Descent và Stochastic Gradient Descent (SGD) để cải thiện hiệu suất và tốc độ hội tụ

1.4.1. Nguyên lí hoạt động của Mini-batch Gradient Descent

- **Xử lý dữ liệu theo lô (mini-batch):** Thay vì cập nhật tham số sau mỗi mẫu (như SGD) hoặc toàn bộ tập dữ liệu (như Batch GD), Mini-batch GD tính toán gradient và cập nhật tham số dựa trên một lô (batch) nhỏ các mẫu.
- **Kích thước Mini-batch:** Kích thước của mỗi mini-batch thường chọn từ 32 đến 256 mẫu, tùy thuộc vào dung lượng bộ nhớ và tính chất của dữ liệu.

.

1.4.2. Công Thức Cập Nhật:

Tham số được cập nhật theo công thức:

$$\theta = \theta - n.\nabla_{\theta} J(\theta, X_{mini-batch})$$

Trong đó : θ là tham số cần tối ưu.

n là tốc độ học (learning rate).

$J(\theta, X_{mini-batch})$ là hàm mất mát tính trên mỗi mini-batch.

1.4.3. Lựa chọn kích thước mini-bath

- Kích thước mini-batch phải được lựa chọn cẩn thận dựa trên dung lượng bộ nhớ và tính toán của hệ thống, cũng như tính chất của dữ liệu và mô hình.
- Kích thước quá lớn có thể dẫn đến việc mất mát tính chất SGD, trong khi kích thước quá nhỏ có thể không hiệu quả về mặt tính toán.

1.4.4. Ứng dụng

- Mini-batch GD là phương pháp được sử dụng rộng rãi trong việc huấn luyện các mạng nơ-ron sâu, do nó mang lại sự cân bằng tốt giữa hiệu suất và tốc độ học.

1.5. Tìm hiểu về Adaptive Learning Rate Methods

Các phương pháp Adaptive Learning Rate trong học máy và học sâu được thiết kế để tự động điều chỉnh tốc độ học (η) trong quá trình huấn luyện. Điều này giúp cải thiện hiệu suất và tốc độ hội tụ của mô hình. Các phương pháp này thích hợp cho các vấn đề phức tạp, nơi mà việc chọn tốc độ học cố định là không hiệu quả . Ta sẽ tìm hiểu về một số phương pháp Adaptive Learning Rate Methods phổ biến

1.5.1. Tìm hiểu về Adagrad (Adaptive Gradient Algorithm)

- **- Cơ chế:** Adagrad điều chỉnh tốc độ học cho mỗi tham số dựa trên lịch sử của gradient. Nó phân biệt giữa các tham số, làm cho những tham số ít được cập nhật có tốc độ học lớn hơn.
- **Công thức:** $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \nabla_\theta J(\theta_t)$

Trong đó : *Gt* là tổng bình phương của gradient cho đến thời điểm *t*, và ϵ là một số nhỏ để tránh chia cho 0.

- **Ưu điểm:** Hiệu quả với dữ liệu thưa và các vấn đề không đồng nhất.
- **Nhược điểm:** Tốc độ học giảm quá nhanh, có thể làm cho Adagrad ngừng học sớm trong quá trình huấn luyện.

.

1.5.2. Tìm hiểu về RMSprop (Root Mean Square Propagation)

- **Cơ chế:** RMSprop cải thiện Adagrad bằng cách sử dụng trung bình di động của bình phương gradient, giúp tránh vấn đề giảm tốc độ học quá nhanh.
- **Công thức:**

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

Trong đó : $E[g^2]_t$ là trung bình di động của bình phương gradient.

Ưu điểm: Giảm thiểu vấn đề tốc độ học giảm quá nhanh của Adagrad.

1.5.3. Tìm hiểu về Adam (Adaptive Moment Estimation)

Cơ chế: Kết hợp ý tưởng của Momentum và RMSprop. Adam không chỉ tính toán trung bình di động của bình phương gradient (như RMSprop) mà còn giữ lại trung bình di động của gradient (tương tự như Momentum).

Công thức :

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- **Ưu điểm:** Hiệu quả trong nhiều tình huống, ít cần điều chỉnh tham số.
- **Nhược điểm:** Có thể không hiệu quả trong một số trường hợp đặc biệt, như các mô hình

2. So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

Optimizer	Cơ chế	Ưu điểm	Nhược điểm

Gradient Descent	Cập nhật tham số dựa trên đạo hàm của hàm mất mát	Đơn giản, dễ hiểu, hiệu quả với dữ liệu ít và đơn giản	Tốc độ học cố định, có thể mất nhiều thời gian để hội tụ
Stochastic Gradient Descent	Cập nhật tham số sau mỗi mẫu hoặc mini-batch nhỏ	Nhanh hơn GD với dữ liệu lớn, có khả năng thoát khỏi cực tiểu địa phương	Có thể không ổn định, dao động lớn trong quá trình học
Momentum	Sử dụng động lượng để 'đẩy' cập nhật về hướng đúng hơn	Giảm dao động, hội tụ nhanh hơn, hiệu quả trong bề mất mát phức tạp	Cần chọn tham số động lượng, có thể vượt qua điểm tối ưu
Adagrad	Điều chỉnh tốc độ học cho mỗi tham số, giảm dần qua thời gian	Hiệu quả với dữ liệu thưa và các vấn đề không đồng nhất	Tốc độ học có thể giảm quá nhanh và mô hình dừng học sớm
RM.Sprop	Cải thiện Adagrad bằng cách điều chỉnh tốc độ học dựa trên trung bình di động của bình phương gradient	Tránh vấn đề học quá chậm của Adagrad, hiệu quả trong nhiều trường hợp	Cần điều chỉnh tham số, có thể không ổn định trong một số trường hợp
Adam	Kết hợp ý tưởng của Momentum và RMSprop, tính toán trung bình di động của gradient và	Hiệu quả trong nhiều tình huống, ít cần điều chỉnh tham số	Có thể không hiệu quả trong một số trường hợp đặc biệ

	bình phương gradient		

III. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

1. Tìm hiểu về Continual Learning khi xây dựng một giải pháp học máy để giải quyết

a) Khái niệm

Ban đầu, Lifelong learning trong deep learning thường được gọi là continual learning

• Continual learning quan tâm đến vấn đề quên nghiêm trọng (catastrophic forgetting)

• Khi huấn luyện trên tasks mới, neural network có xu hướng quên thông tin đã học được từ những tasks trước.

⇒ Làm giảm hiệu suất mô hình trên những tasks cũ.

• Hiện tượng này liên quan chặt chẽ đến hiện tượng stability-plasticity dilemma .

• Nếu model quá ổn định thì khó khăn học thông tin mới.

• Nếu model quá mềm dẻo thì gặp vấn đề quên đi thông tin đã học được.

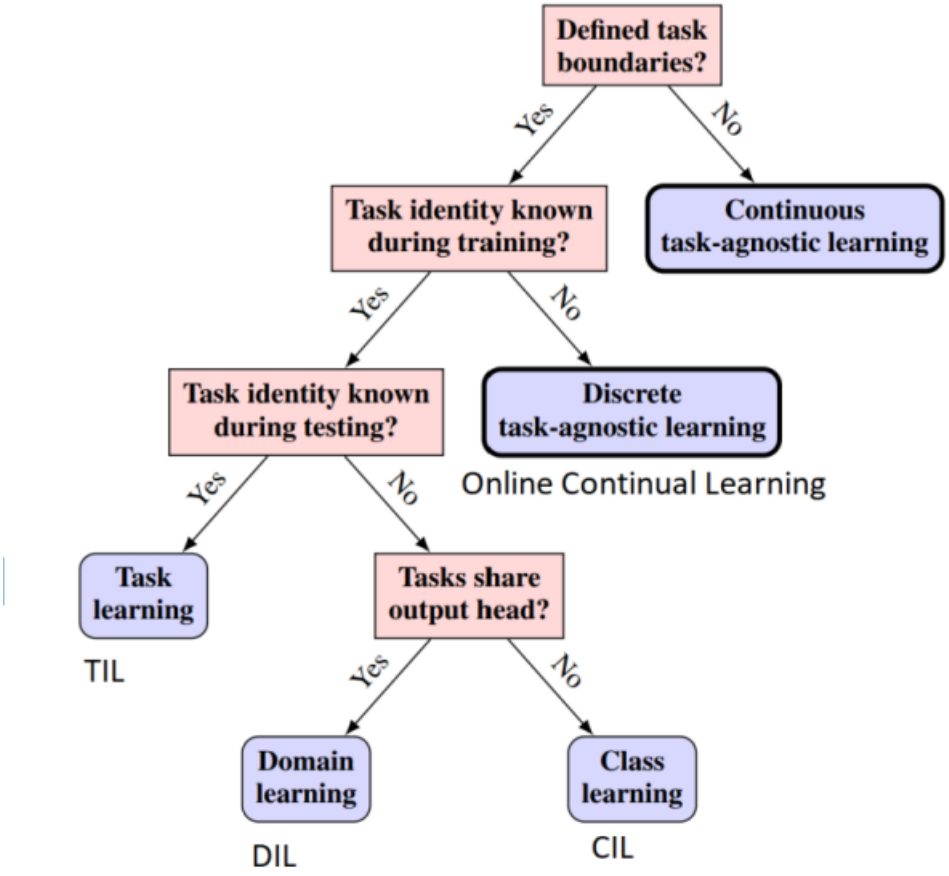
• Continual learning được xem xét trong các ML models: SVM , Topic models , decision tree

b) Các kịch bản phổ biến

-Có kịch bản phổ biến

+ Task-Incremental Learning (TIL).
+ Domain-Incremental Learning (DIL).
+ Class-Incremental Learning (CIL)

Online continual learning: Nghiên cứu continual learning trong online setup (dữ liệu đến liên tục, phân phối của dữ liệu thường xuyên thay đổi).



Một số kịch bản chính trong CL [10]

c) Số liệu đánh giá

• Sử dụng tập hold-out test cho mỗi task trong T tasks • Huấn luyện lần lượt các task và đánh giá trên tập hold-out test của các tasks: Xây dựng matrix R_{TxT} trong đó $R_{I,j}$là accuracy của mô hình trên task I khi học xong task J.

-Có 3 độ đo phổ biến: Average Accuracy (ACC), Backward Transfer (BWT), Forward Transfer (FWT)

$$ACC = \frac{1}{T} \sum_{i=1}^{T} R_{T,i}$$

$$BWT = \frac{1}{T-1} \sum_{i=1}^{T} (R_{T,i} - R_{i,i})$$

$$FWT = \frac{1}{T-1} \sum_{i=1}^{T} R_{i-1,i}$$

d) Một số cách tiếp cận

Ba cách tiếp cận phổ biến:

• Prior/Regularization-based approach

• Memory-based approach

• Architecture-based approach

***Cách tiếp cận Prior/Regularization-based approach** :

-Bổ sung đại lượng regularization hoặc distillation loss vào hàm loss hiện tại để giữ tham số mô hình hiện tại "gần" với tham số mô hình ở task trước

$$\mathcal{L}_k(\theta) = L_k(\theta) + \lambda(\theta - \theta^{t-1})^T \Omega^{t-1}(\theta - \theta^{t-1})$$

Với θ là tham số mô hình (weights), L_k là hàm loss cho task , Ω^{t-1} mã hóa độ quan trọng của weights

• Nhiều cách khác nhau mã hóa độ quan trọng

• Elastic Weight Consolidation (EWC) sử dụng Fisher Information

• Synaptic Intelligence (SI) dựa trên giảm của hàm loss

• Memory Aware Synapses (MAS) dựa trên biến đổi của đầu ra (output)

• Variational Continual Learning (VCL) sử dụng cách tiếp cận Bayes:

$$P(\theta|D_1, D_2, \dots, D_t) \propto P(D_t|\theta)P(\theta|D_1, D_2, \dots, D_{t-1})$$

Với $\mathbf{D_t}$ là dữ liệu huấn luyện của task t

-Tri thức học được ở những task trước đóng vai trò là tiên nghiệm (prior) cho task hiện tại:

$$P(\theta|D_1, D_2, \dots, D_{t-1}) = q_{t-1}(\theta)$$

• Sử dụng online variational inference

$$E_{q(\theta)} \log P(D_t|\theta) - KL(q(\theta)||q_{t-1}(\theta))$$

***Cách tiếp cận Memory-based approach:**

Sử dụng bộ nhớ nhỏ (memory buffer) [15] để lưu lại data từ các tasks trước và huấn luyện lại cùng với task hiện tại (rehearsal)

• Nhiều chiến lược lựa chọn data đặc trưng được đề xuất

• Chọn data đại diện dựa theo lớp (herding algorithm)

• Chọn data "khó": Nằm gần biên phân loại (decision boundary) hoặc gây ra loss lớn

• Sử dụng các chiến lược bi-level optimization

• Xây dựng các mô hình generative models để sinh lại dữ liệu cũ hoặc sinh lại biểu diễn của dữ liệu cũ

• Gradient Episodic Memory (GEM) and A-GEM lưu dữ liệu để ngăn cập nhật weight theo hướng gradient làm giảm loss trên dữ liệu cũ.

***Cách tiếp cận Architecture-based approach:**

Học một phần kiến trúc (sub-network) cho mỗi task và lưu sub-network

• Đóng băng phần kiến trúc này và mở rộng mạng để học task mới

• Lặp lại 2 bước

Network expansion

Sparse learning

• Với TIL, tiếp cận dựa trên kiến trúc không bị quên task cũ

• Với DIL và CIL, cần bước xác định task boundary khi inference

2. Test Production

Test Production trong học máy là quá trình đánh giá mô hình trong một môi trường sản xuất thực tế, nơi mà mô hình được áp dụng để giải quyết các vấn đề thực tế.

Mục đích: Đảm bảo mô hình không chỉ hoạt động tốt trên dữ liệu thử nghiệm mà còn hiệu quả và ổn định trong môi trường thực.

Thách thức: Xử lý các vấn đề như dữ liệu không đồng nhất, thay đổi mô hình dựa trên feedback thực tế, và quản lý hiệu suất mô hình.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]

https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_c%C3%B3_gi%C3%A1m_s%C3%A1t

[2]

https://machinelearningcoban.com/general/2017/02/06/featureengineering/

Tiếng Anh

[3]

https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/

[4]

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[5]

https://www.kaggle.com/datasets/yasserh/housing-prices-dataset?resource=download