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

**DỰ ÁN CUỐI KÌ**

**MÔN NHẬP MÔN HỌC MÁY**

# BÀI 1

# (Phần riêng)

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN NHÃ THẢO DUY – 52000325**

Lớp **: 20050401**

Khoá **: 24**

**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**

**DỰ ÁN CUỐI KÌ**
**MÔN NHẬP MÔN HỌC MÁY**

# BÀI 1

# (Phần riêng)

Người hướng dẫn: **TS LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN NHÃ THẢO DUY - 52000325**
Lớp **: 20050401**
Khoá **: 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

# LỜI CẢM ƠN

Em đã được học những kiến thức từ môn Nhập môn học máy qua sự giảng dạy của thầy Lê Anh Cường về lý thuyết và thực hành. Em xin gửi lời cảm ơn sâu sắc tới thầy. Thầy đã nhiệt tình truyền dạy những nội dung về môn học để em có thêm những kiến thức hữu ích giúp em thực hiện bài báo cáo này. Em đã vận dụng những bài học được thầy giảng dạy để hoàn thành bài báo cáo này. Tuy nhiên, vì sự hiểu biết còn hạn chế của em, bài báo cáo còn nhiều sai sót và chưa chỉnh chu như em mong muốn.

Em rất mong nhận được nhận xét và đánh giá của thầy để có thể rút kinh nghiệm cũng như sửa chữa lỗi sai của bản thân.

Em xin chân thành 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 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 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 23 tháng 12 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Nhã Thảo Duy*

# 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

**Đề bài:**

**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.

**Tóm tắt nội dung bài báo cáo:**

**1. Vấn đề nghiên cứu**

Bài báo cáo này trình bày Bài 1 (phần làm riêng) của dự án cuối kì môn Nhập môn học máy, gồm 2 phần chính:

- Câu 1: Giải quyết ý (1) của bài 1
- Câu 2: Giải quyết ý (2) của bài 1

**2. Hướng tiếp cận**

- Lý thuyết.

**3. Cách giải quyết vấn đề**

Xem lại những nội dung đã học qua slide bài giảng, các kiến thức được ghi chép lại trong quá trình học và nghiên cứu thêm các nguồn tài liệu trên mạng. Vận dụng chúng vào để giải quyết các nội dung trong đề bài.

# MỤC LỤC

# DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

## DANH MỤC HÌNH



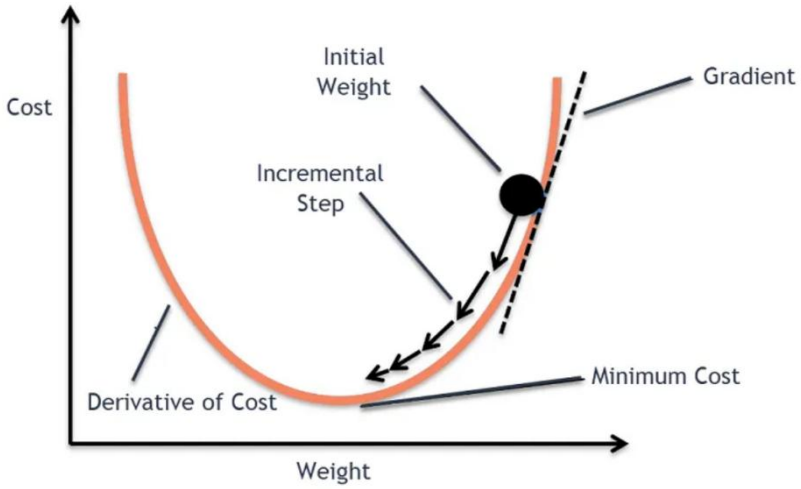
## DANH MỤC BẢNG

# CÂU 1 – Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

## 1.1 Tìm hiểu các phương pháp Optimizer trong huấn luyện mô hình học máy

### *1.1.1 Gradient Descent (GD)*

**Gradient Descent (GD)** là một thuật toán tối ưu hóa được sử dụng để tìm kiếm giá trị nhỏ nhất của một hàm số thông qua việc điều chỉnh các tham số của hàm đó. Thuật toán GD dựa trên việc tính toán gradient của hàm mất mát (hàm mục tiêu cần tối ưu) theo các tham số hiện tại và sử dụng thông tin này để điều chỉnh các tham số theo hướng giảm nhanh nhất của hàm mất mát.




Hình 1. 1 Gradient Descent

Ý tưởng của gradient descent là tính đạo hàm và di chuyển theo hướng giảm dần của gradient. Gradient descent được sử dụng để làm giảm thiểu một hàm cost function J(w) tham số ở đây là w. Đạo hàm sẽ cho chúng ta thấy được độ dốc và xiên của hàm cost function. Vì vậy mà để tối giản hóa cost function, chúng ta cần di chuyển ngược lại hướng của đạo hàm.

#### 1.1.1.1 Gradient Descent cho hàm 1 biến

Xét hàm một biến f(x). Thuật toán Gradient Descent tìm cực tiểu của hàm số bằng cách khởi tạo giá trị x tại một vị trí ngẫu nhiên, sau đó di chuyển x ngược hướng với đạo hàm của f(x). Thao tác này sẽ được lặp lại cho đến khi đạt đến một ngưỡng nào đó.

Công thức cập nhật x trong Gradient Descent:

$$x_{t+1} = x_t - \eta f'(x_t)$$

Trong đó:

- $x_t$ là giá trị x tại bước thứ t,
- $\eta$ là learning rate (tốc độ học),
- $f'(x_t)$ là đạo hàm của hàm f tại $x_t$.

#### 1.1.1.2 Gradient Descent cho hàm nhiều biến

Đối với hàm nhiều biến $f(\theta)$ trong đó $\theta$ là một vector. Đạo hàm của hàm số đó tại một điểm $\theta$ bất kỳ được ký hiệu là $\nabla_\theta f(\theta)$. Tương tự như hàm 1 biến, thuật toán GD cho hàm nhiều biến cũng bắt đầu bằng một điểm dự đoán $\theta_0$, sau đó, ở vòng lặp thứ t, quy tắc cập nhật là:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta_t)$$

Hoặc viết dưới dạng đơn giản hơn: $\theta = \theta - \eta \nabla_\theta f(\theta)$

Trong đó:

- $\theta_t$ là giá trị $\theta$ tại bước thứ t,
- $\eta$ là learning rate (tốc độ học),
- $\nabla_\theta f(\theta)$ là đạo hàm mất mát tại $\theta$

#### 1.1.1.3 Các thuật toán Gradient Descent

Có nhiều biến thể của thuật toán này tuỳ thuộc vào việc lựa chọn dữ liệu huấn luyện và thứ tự cập nhật:

- *Batch Gradient Descent*: Sử dụng toàn bộ dữ liệu huấn luyện để cập nhật θ trong mỗi bước lặp. Thuật toán này có độ chính xác cao nhưng mất nhiều thời gian do tính toán trên toàn bộ dữ liệu.
- *Stochastic Gradient Descent (SGD)*: Sử dụng chỉ một điểm dữ liệu huấn luyện ngẫu nhiên để cập nhật θ. Tốc độ hội tụ nhanh hơn, nhưng độ chính xác thấp hơn so với Batch Gradient Descent.
- *Mini-batch Gradient Descent*: Sử dụng một số lượng nhỏ điểm dữ liệu huấn luyện (mini-batch) để cập nhật θ. Kết hợp ưu điểm của cả Batch Gradient Descent và SGD.

#### 1.1.1.4 Biến thể

Một số cải tiến của Gradient Descent để tăng tốc độ hội tụ và giảm thiểu dao động trong quá trình hội tụ:

- *Momentum*: Giảm dao động qua lại của gradient và đi nhanh hơn dọc theo hướng tiến.
- *Nesterov Accelerated Gradient (NAG)*: Sử dụng momentum bằng cách tính gradient trước khi cập nhật vị trí của θ.
- *Adaptive Gradient (Adagrad)*: Đưa vào learning rate riêng cho mỗi parameter.
- *Adaptive Moment Estimation (Adam)*: Kết hợp momentum và adaptive learning rate.

### *1.1.2 Stochastic Gradient Descent (SGD)*

Trong thuật toán này, tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu $x_i$ rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực

hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả.

Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là một epoch. Với GD thông thường thì mỗi epoch ứng với 1 lần cập nhật $\theta$, với SGD thì mỗi epoch ứng với N lần cập nhật $\theta$ với N là số điểm dữ liệu. Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt). Vì vậy SGD phù hợp với các bài toán có lượng cơ sở dữ liệu lớn và các bài toán yêu cầu mô hình thay đổi liên tục (online learning).

**Thứ tự lựa chọn điểm dữ liệu:**

Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần shuffle (xáo trộn) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD.

**Quy tắc cập nhật của SGD là:**

$$\theta = \theta - \eta \nabla_\theta J(\theta; x_i; y_i)$$

Trong đó, $J(\theta; x_i; y_i)$ là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là $(x_i; y_i)$.

Chú ý: chúng ta hoàn toàn có thể áp dụng các thuật toán tăng tốc GD như Momentum, AdaGrad,… vào SGD.

### *1.1.3 Gradient Descent với Momentum*

**Gradient Descent với Momentum** là một phương pháp tối ưu hóa kết hợp giữa GD và Momentum để cải thiện tốc độ hội tụ và giảm độ dao động trong quá trình huấn luyện mô hình học máy. Nó sử dụng một hệ số momentum để tích lũy thông tin về gradient trước đó và giúp cho quá trình tối ưu hóa trở nên ổn định hơn.

Momentum là một phương pháp tối ưu hóa độ dốc giảm dần, thêm phần trăm vectơ cập nhật trước vào vectơ cập nhật hiện tại để tăng tốc quá trình học. Về cơ bản, động lượng là một phương pháp làm trơn tru việc cập nhật tham số mô hình và cho

phép trình tối ưu hóa tiếp tục tiến lên theo cùng hướng như trước đây, giảm thiểu dao động và tăng tốc độ hội tụ.

Trong GD, chúng ta cần tính lượng thay đổi ở thời điểm t để cập nhật vị trí mới cho nghiệm (tức hòn bi). Nếu coi đại lượng này như vận tốc $v_t$ trong vật lý, vị trí mới của hòn bi sẽ là $\theta_{t+1} = \theta_t - v_t$. Dấu trừ thể hiện việc phải di chuyển ngược với đạo hàm. Công việc của chúng ta bây giờ là tính đại lượng $v_t$ sao cho nó vừa mang thông tin của độ dốc (tức đạo hàm), vừa mang thông tin của đà, tức vận tốc trước đó $v_{t-1}$ (chúng ta coi như vận tốc ban đầu $v_0$=0). Một cách đơn giản nhất, ta có thể cộng (có trọng số) hai đại lượng này lại:

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

Trong đó:

- $\gamma$ là hệ số momentum, thường có giá trị trong khoảng (0, 1) (thường được chọn khoảng 0.9),
- $\eta$ là learning rate (tốc độ học),
- $v_t$ là vận tốc tại thời điểm trước đó,
- $\nabla_\theta J(\theta)$ chính là độ dốc của điểm trước đó. Sau đó vị trí mới của hòn bi được xác định như sau:

$$\theta = \theta - v_t$$

### *1.1.4 AdaGrad (Adaptive Gradient)*

**AdaGrad (Adaptive Gradient)** là một phương pháp tối ưu hóa trong huấn luyện mô hình học máy, tự động điều chỉnh tỷ lệ học (learning rate) cho từng tham số dựa trên lịch sử của gradient. AdaGrad là một biến thể của thuật toán giảm độ dốc giúp điều chỉnh tốc độ học cho từng tham số riêng lẻ. Ý tưởng cơ bản là tốc độ học sẽ thấp hơn đối với các tham số có độ dốc cao và đối với các tham số có độ dốc thấp, tốc độ học sẽ lớn hơn. Điều này cho phép thuật toán hội tụ nhanh hơn đối với dữ liệu thưa thớt.

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

Trong đó:

- $g_t$ là gradient của hàm mất mát tại bước t,
- $g_{t,i}$ là đạo hàm riêng của hàm mất mát theo $\theta_i$ tại bước t.

**Quy tắc cập nhật của AdaGrad:**

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

Trong quy tắc cập nhật của mình:

- Adagrad sửa đổi tốc độ học tập chung $\eta$ tại mỗi bước thời gian t cho mọi tham số $\theta_i$ dựa trên độ dốc trong quá khứ cho $\theta_i$.
- Mẫu số là chuẩn L2 (L2 norm) của ma trận đường chéo $G_t$, trong đó phần tử i,i là tổng bình phương của các gradient tương ứng với $\theta_i$ tính đến bước t .
- $\varepsilon$ là một số dương khá nhỏ nhằm tránh trường hợp mẫu số bằng 0.

**Quy tắc cập nhật trên có thể viết dưới dạng tổng quát hơn như sau:**

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \odot g_t$$

Trong đó, $\odot$ là phép nhân ma trận-vectơ giữa $G_t$ và $g_t$. Có thể nhận thấy rằng trong thuật toán Adagrad tốc độ học được tự động điều chỉnh. Adagrad thường khá hiệu quả đối với bài toán có dữ liệu phân mảnh. Tuy nhiên, hạn chế của Adagrad là các tổng bình phương ở mẫu số ngày càng lớn khiến tốc độ học ngày càng giảm và có thể tiệm cận đến giá trị 0 khiến cho quá trình huấn luyện gần như đóng băng. Bên cạnh đó, giá trị tốc độ học $\eta$ cũng phải được xác định một cách thủ công.

### *1.1.5 RMSProp (Root Mean Square Propagation)*

**RMSprop** là một thuật toán tối ưu hóa sử dụng trung bình bình phương của gradient để chuẩn hóa gradient. Có tác dụng cân bằng kích thước bước - giảm bước cho độ dốc lớn để tránh hiện tượng phát nổ độ dốc (Exploding Gradient), và tăng bước cho độ dốc nhỏ để tránh biến mất độ dốc (Vanishing Gradient). RMSProp tự động điều

chỉnh tốc độ học tập, và chọn một tỉ lệ học tập khác nhau cho mỗi tham số. Phương pháp cập nhật các trọng số được thực hiện như mô tả:

$$s_t = \rho s_{t-1} + (1-\rho) * g_t^2$$

$$\Delta x_t = -\frac{\eta}{\sqrt{s_t + \varepsilon}} * g_t$$

$$x_{t+1} = x_t + \Delta x_t$$

Trong đó:

- $s_t$ là tích luỹ phương sai của các gradient trong quá khứ,
- $\rho$ là tham số suy giảm,
- $\Delta x_t$ là sự thay đổi các tham số trong mô hình,
- $g_t$ là gradient của các tham số tại vòng lặp t,
- $\varepsilon$ là một hằng số nhỏ được thêm vào để tránh chia cho 0.

### *1.1.6 Adam (Adaptive Moment Estimation)*

**Adam** được xem như là sự kết hợp của RMSprop và Stochastic Gradient Descent với động lượng. Adam (Ước tính thời điểm thích ứng) là một thuật toán tối ưu hóa sử dụng đường trung bình động của gradient và gradient bình phương để chia tỷ lệ học tập. Adam tính toán tốc độ học cho từng tham số bằng cách sử dụng khoảnh khắc thứ nhất và thứ hai của độ dốc và điều chỉnh tốc độ học dựa trên sự phân rã theo cấp số nhân của những khoảnh khắc này. Adam đã được chứng minh là hoạt động tốt trong nhiều ứng dụng và được coi là một trong những thuật toán tối ưu hóa tốt nhất. Tuy nhiên, qua nghiên cứu thực nghiệm, trong một số trường hợp, Adam vẫn còn gặp phải nhiều thiếu sót so với thuật toán SGD. Thuật toán Adam được mô tả:

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

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

Trong đó:

- $v_t$ là trung bình động của bình phương,
- $m_t$ là trung bình động của gradient,

- $g_t$ là gradient tại thời điểm t,
- $\beta_1$ và $\beta_2$ là tốc độ di chuyển.

## 1.2 So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

So sánh các yếu tố quan trọng của các phương pháp tối ưu hóa GD, SGD, Momentum, AdaGrad, RMSProp và Adam trong mô hình học máy:

Bảng 1. 1 So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

| | Tính ổn định | Tốc độ hội tụ | Điều chỉnh tỷ lệ học |
|---|---|---|---|
| **GD** | Có tính ổn định cao do cập nhật tham số dựa trên toàn bộ tập dữ liệu. | Có thể hội tụ đến điểm cực tiểu cục bộ hoặc toàn cục đối với hàm lồi, nhưng có thể bị kẹt trong "thung lũng". | Yêu cầu điều chỉnh tỷ lệ học thủ công. |
| **SGD** | Không ổn định, có thể dao động quanh điểm cực tiểu do cập nhật tham số dựa trên một mẫu dữ liệu ngẫu nhiên. | Có thể hội tụ tốt với đủ lượng dữ liệu, nhưng yêu cầu tốt về thiết kế tỷ lệ học. | Yêu cầu điều chỉnh tỷ lệ học thủ công. |
| **Momentum** | Có tính ổn định cao hơn SGD nhờ sử dụng đà trong quá trình cập nhật tham số. | Giúp tăng tốc quá trình tối ưu hóa và vượt qua điểm cực tiểu cục bộ, giúp hội tụ nhanh hơn SGD. | Cần điều chỉnh hệ số ma trận động (momentum) và tỷ lệ học. |

| | | | |
|---|---|---|---|
| **AdaGrad** | Có tính ổn định cao trong việc điều chỉnh tỷ lệ học cho từng tham số dựa trên lịch sử gradient. | Có thể hội tụ nhanh đối với các bài toán có các tham số thưa thớt hoặc độ lớn khác nhau. | Tự động điều chỉnh tỷ lệ học cho từng tham số dựa trên lịch sử gradient. |
| **RMSProp** | Cải thiện tính ổn định hơn AdaGrad bằng cách điều chỉnh tỷ lệ học theo từng tham số và giá trị tham số giảm dần. | Cải thiện tốc độ hội tụ và ổn định của quá trình tối ưu hóa so với GD và SGD. | Cần điều chỉnh giá trị tham số giảm dần (decay rate). |
| **Adam** | Kết hợp lợi thế của Momentum và RMSProp, có tính ổn định cao và hiệu quả trong nhiều bài toán. | Kết hợp lợi thế của Momentum và RMSProp, có tốc độ hội tụ tốt và hiệu quả trong nhiều bài toán. | Tự động điều chỉnh tỷ lệ học, hệ số ma trận động và giá trị tham số giảm dần. |

# 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 Tìm hiểu về Continual Learning

**Continual Learning** là một lĩnh vực học máy tập trung vào việc phát triển các mô hình có thể học và thích nghi với dữ liệu mới theo thời gian. Trong thế giới thực, dữ liệu và yêu cầu của các ứng dụng học máy luôn thay đổi. Do đó, việc xây dựng các mô hình có thể học liên tục là rất quan trọng để đảm bảo hiệu suất của các mô hình này trong thời gian dài.

### *2.1.1 Phân loại*

Có hai loại chính của Continual Learning:

- Incremental Learning: Mô hình được đào tạo trên dữ liệu mới mà không cần xóa dữ liệu cũ.
- Lifelong Learning: Mô hình được đào tạo trên dữ liệu mới trong suốt vòng đời của nó.

### *2.1.2 Ưu điểm*

Ưu điểm của Continual Learning:

- Giúp mô hình học và thích nghi với dữ liệu mới theo thời gian.
- Giảm chi phí và thời gian đào tạo lại mô hình.
- Tăng cường khả năng bảo mật và quyền riêng tư của dữ liệu.

### *2.1.3 Nhược điểm*

Nhược điểm của Continual Learning:

- Overfitting: Mô hình có thể học quá mức dữ liệu mới và dẫn đến kết quả kém trên dữ liệu cũ.
- Data Drift: Dữ liệu mới có thể khác biệt đáng kể so với dữ liệu cũ, gây khó khăn cho việc học liên tục.
- Computational Cost: Quá trình học liên tục có thể tốn kém về mặt tính toán.

### *2.1.4 Các phương pháp Continual Learning*

Có nhiều phương pháp Continual Learning khác nhau, mỗi phương pháp có ưu và nhược điểm riêng. Một số phương pháp phổ biến bao gồm:

- *Entropy Regularization*: Phương pháp này thêm một thuật ngữ vào hàm mất mát để giảm thiểu entropi của mô hình. Điều này giúp ngăn chặn mô hình học quá mức dữ liệu mới.
- *Elastic Weight Consolidation*: Phương pháp này giữ lại một số trọng số của mô hình cũ và sử dụng chúng để điều chỉnh trọng số mới. Điều này giúp mô hình duy trì khả năng dự đoán của nó trên dữ liệu cũ.
- *Generative Replay*: Phương pháp này tạo ra dữ liệu mới từ dữ liệu cũ. Dữ liệu mới này được sử dụng để đào tạo mô hình mới.

### *2.1.5 Giải pháp*

Khi xây dựng một giải pháp học máy để giải quyết một bài toán cụ thể, Continual Learning có thể được áp dụng như sau:

- *Lựa chọn mô hình phù hợp*: Không phải tất cả các mô hình học máy đều phù hợp với Continual Learning. Một số mô hình, chẳng hạn như các mạng nơ-ron sâu, có thể học liên tục hiệu quả hơn các mô hình khác.
- *Áp dụng các kỹ thuật Continual Learning*: Có nhiều kỹ thuật Continual Learning khác nhau có thể được sử dụng để cải thiện khả năng học liên tục của mô hình.

Một số lưu ý khi áp dụng Continual Learning:

- *Cần lựa chọn mô hình phù hợp*: Không phải tất cả các mô hình học máy đều phù hợp với Continual Learning. Một số mô hình, chẳng hạn như các mạng nơ-ron sâu, có thể học liên tục hiệu quả hơn các mô hình khác.
- *Cần áp dụng các kỹ thuật Continual Learning phù hợp*: Có nhiều kỹ thuật Continual Learning khác nhau có thể được sử dụng. Cần lựa chọn các kỹ thuật phù hợp với mô hình và dữ liệu cụ thể.

- *Cần có kế hoạch cập nhật dữ liệu*: Để Continual Learning hoạt động hiệu quả, cần có kế hoạch cập nhật dữ liệu thường xuyên.

## 2.2 Tìm hiểu về Test Production

**Test Production** là một phương pháp kiểm thử phần mềm trong đó các ứng dụng được kiểm thử trong môi trường sản xuất thực tế. Phương pháp này có thể được sử dụng để kiểm tra các ứng dụng học máy, đặc biệt là các ứng dụng có thể ảnh hưởng đến người dùng cuối.

### *2.2.1 Ưu điểm*

Ưu điểm của Test Production:

- Kiểm tra ứng dụng trong môi trường thực tế giúp đảm bảo rằng ứng dụng hoạt động chính xác và đáp ứng các yêu cầu của người dùng.
- Kiểm tra ứng dụng trong môi trường sản xuất giúp phát hiện các lỗi hoặc vấn đề có thể không được phát hiện trong môi trường thử nghiệm.
- Kiểm tra ứng dụng trong môi trường sản xuất giúp cải thiện độ tin cậy và khả năng sẵn sàng của ứng dụng.

### *2.2.2 Nhược điểm*

Nhược điểm của Test Production:

- Kiểm thử ứng dụng trong môi trường sản xuất có thể gây nguy hiểm cho người dùng cuối.
- Kiểm thử ứng dụng trong môi trường sản xuất có thể tốn kém về mặt thời gian và tài nguyên.

### *2.2.3 Các phương pháp Test Production*

Có nhiều phương pháp Test Production khác nhau, mỗi phương pháp có ưu và nhược điểm riêng. Một số phương pháp phổ biến bao gồm:

- *A/B Testing*: Phương pháp này sử dụng hai phiên bản của ứng dụng, một phiên bản là phiên bản hiện tại và phiên bản còn lại là phiên bản mới. Người dùng được chia ngẫu nhiên thành hai nhóm, mỗi nhóm sử dụng một phiên

bản của ứng dụng. Kết quả của việc sử dụng hai phiên bản ứng dụng được so sánh để đánh giá hiệu suất của phiên bản mới.

- *Canary Testing*: Phương pháp này bắt đầu triển khai phiên bản mới của ứng dụng cho một số lượng nhỏ người dùng trước khi triển khai cho tất cả người dùng. Điều này giúp xác định bất kỳ vấn đề nào với phiên bản mới trước khi triển khai cho tất cả người dùng.
- *Blue-Green Deployment*: Phương pháp này triển khai phiên bản mới của ứng dụng song song với phiên bản hiện tại. Khi phiên bản mới đã sẵn sàng, phiên bản hiện tại sẽ bị tắt và phiên bản mới sẽ được kích hoạt.

### *2.2.4 Giải pháp*

Khi xây dựng một giải pháp học máy để giải quyết một bài toán cụ thể, Test Production có thể được áp dụng như sau:

- *Kiểm tra khả năng hoạt động của mô hình trong môi trường sản xuất*: Test Production giúp đảm bảo rằng mô hình có thể hoạt động chính xác và đáp ứng các yêu cầu của người dùng cuối trong môi trường sản xuất thực tế.
- *Phát hiện các lỗi hoặc vấn đề có thể không được phát hiện trong môi trường thử nghiệm*: Test Production giúp phát hiện các lỗi hoặc vấn đề có thể không được phát hiện trong môi trường thử nghiệm, chẳng hạn như các lỗi do tương tác giữa mô hình và các thành phần khác của hệ thống.
- *Cải thiện độ tin cậy và khả năng sẵn sàng của hệ thống*: Test Production giúp cải thiện độ tin cậy và khả năng sẵn sàng của hệ thống bằng cách phát hiện và giải quyết các lỗi hoặc vấn đề trước khi chúng ảnh hưởng đến người dùng cuối.

Test Production là một kỹ thuật quan trọng có thể được sử dụng để cải thiện chất lượng và độ tin cậy của các giải pháp học máy. Khi xây dựng một giải pháp học máy để giải quyết một bài toán cụ thể, cần xem xét xem Test Production có thể được áp dụng hay không.

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. Trí tuệ nhân tạo (2023), "Bài 5: Gradient Descent", URL: https://trituenhantao.io/machine-learning-co-ban/bai-5-gradient-descent/
2. Tiep Vu Huu (2017), "Bài 8: Gradient Descent (phần 2/2)", URL: https://machinelearningcoban.com/2017/01/16/gradientdescent2/
3. Khoa Tin học – Trường ĐHSP Huế (2021), "ĐÁNH GIÁ HIỆU NĂNG CỦA CÁC THUẬT TOÁN TỐI ƯU TRONG MÔ HÌNH HỌC SÂU ĐỐI VỚI BÀI TOÁN PHÂN LỚP HÌNH ẢNH", URL: https://csdlkhoahoc.hueuni.edu.vn/data/2021/5/BaiDangHoiThao.pdf
4. Khoa Điện-Điện tử và Công nghệ vật liệu Trường Đại học Khoa học, Đại học Huế (2021), "ĐÁNH GIÁ CÁC THUẬT TOÁN TỐI ƯU ĐỐI VỚI MÔ HÌNH MẠNG NƠ-RON TÍCH CHẬP TRONG TÁC VỤ NHẬN DIỆN HÌNH ẢNH", URL: https://jos.husc.edu.vn/backup/upload/vol_18/no_1/668_fulltext_4.%C4%90TVT%20-%20Phuoc%20-%20Vuong%20Quang%20Phuoc.pdf
5. anon73939363 (2020), "Phương pháp Gradient Descent", URL: https://stories.magestore.com/t/ph-ng-phap-gradient-descent/1301.

**Tiếng Anh**

6. Datacamp (2023), "What is Continuous Learning? Revolutionizing Machine Learning & Adaptability", URL: https://www.datacamp.com/blog/what-is-continuous-learning