

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



**ĐỒ ÁN CUỐI KỲ**

**NHẬP MÔN HỌC MÁY**

*Người hướng dẫn:* **Giảng viên LÊ ANH CƯỜNG**

*Người thực hiện:* **ĐINH THỊ NGỌC PHƯỢNG – 52100923**

**Lớp: 21050301**

**Khóa: 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 HỌC MÁY**

*Người hướng dẫn:* **Giảng viên LÊ ANH CƯỜNG**

*Người thực hiện:* **ĐINH THỊ NGỌC PHƯỢNG – 52100923**

**Lớp: 21050301**

**Khóa: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

## LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành qua bài báo cáo cuối kỳ môn học "Nhập môn học máy", để bày tỏ lòng biết ơn sâu sắc đến Thầy Cô Khoa Công nghệ thông tin Đại học Tôn Đức Thắng. Đồng thời, Em muốn bày tỏ lòng biết ơn đến thầy Lê Anh Cường, người đã dành thời gian và công sức để tạo điều kiện cho Em có cơ hội tìm hiểu và mở rộng kiến thức về môn học này. Vì kiến thức của Em còn hạn chế, trong quá trình học tập và nghiên cứu chuyên đề này, không tránh khỏi các sai sót. Em mong rằng sẽ nhận được những kinh nghiệm quý báu từ Cô để em có thể tiếp thu và hoàn thiện hơn.

Em xin gửi lời chúc tốt đẹp đến Thầy Cô trong Khoa Công nghệ thông tin và Thầy Hiệu trưởng Trường Tôn Đức Thắng - Thầy Trần Trọng Đạo, hy vọng rằng Thầy và Cô luôn có sức khỏe dồi dào và niềm tin để tiếp tục đảm nhận sứ mệnh cao đẹp truyền đạt kiến thức cho các thế hệ sinh viên Trường Đại học Tôn Đức Thắng.

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

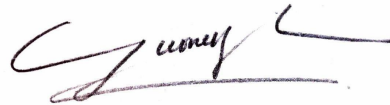
Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Giảng viên Hồ Thị Linh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng 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 chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 12 tháng 12 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*



*Đinh Thị Ngọc Phượng*

## PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

### Phần xác nhận của GV hướng dẫn

---

---

---

---

---

---

---

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

### Phần đánh giá của GV chấm bài

---

---

---

---

---

---

---

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

## TÓM TẮT

Báo cáo này là bài đồ án cuối kỳ môn Nhập môn học máy năm 2023, với các nội dung chính như sau:

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

### **Yêu cầu 1: (Tương ứng với Chương 1)**

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

### **Yêu cầu 2: (Tương ứng với 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.

## MỤC LỤC

LỜI CẢM ƠN.....	1
ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG...2	
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN.....	3
TÓM TẮT.....	4
MỤC LỤC.....	5
CHƯƠNG I: TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY.....	7
1. Thuật toán tối ưu optimizers.....	7
Khái niệm.....	7
2. Các thuật toán tối ưu.....	7
2.1 Gradient Descent (GD).....	7
2.1.1 Khái niệm.....	7
2.1.2 Cách hoạt động.....	7
2.1.3 Ưu điểm.....	10
2.1.4 Nhược điểm.....	10
2.2 Nesterov Momentum.....	11
2.2.1 Khái niệm.....	11
2.2.2 Cách hoạt động.....	12
2.2.3 Ưu điểm.....	13
2.2.4 Nhược điểm.....	14
2.3 Stochastic Gradient Descent.....	14
2.3.1 Khái niệm.....	14
2.3.2 Cách hoạt động.....	15
2.3.3 Ưu điểm.....	15
2.3.4 Nhược điểm.....	15
2.4 Mini-batch Gradient Descent.....	16
2.4.1 Khái niệm.....	16
2.4.2 Cách hoạt động.....	16
2.4.3 Ưu điểm.....	17
2.4.4 Nhược điểm.....	18

2.5 Root Mean Square Propagation (RMSProp).....	18
2.5.1 Khái niệm.....	18
2.5.2 Cách hoạt động.....	19
2.5.3 Ưu điểm.....	19
2.5.4 Nhược điểm.....	19
CHƯƠNG II: CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY.....	21
1. Continual learning.....	21
1.1 Khái niệm.....	21
1.2 Xây dựng model.....	21
1.3 Triển khai và kiểm tra.....	21
1.4 Vấn đề Catastrophic forgetting.....	22
1.4.1 Catastrophic forgetting là gì.....	22
1.4.2 Nguyên nhân.....	22
1.4.3 Giải pháp.....	22
1.4.3.1 Regularization.....	23
1.4.3.2 Rehearsal.....	23
1.4.3.3 Memory replay.....	24
1.5 Ứng dụng Continual learning.....	24
2. Test Production.....	24
2.1 Khái niệm Có bốn loại kiểm thử chính được sử dụng tại các giai đoạn khác nhau trong chu kỳ phát triển.....	25
2.2 Các phân loại kiểm thử.....	25
2.3 Cách thức thực hiện test.....	26
2.4 Những thách thức của Test Production.....	27
TÀI LIỆU THAM KHẢO.....	29



# CHƯƠNG I: TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

## 1. Thuật toán tối ưu optimizers

### Khái niệm

- Thuật toán tối ưu là học các đặc trưng từ dữ liệu đầu vào rồi từ đó sẽ tìm ra các tham số sao cho hàm mất mát đạt giá trị nhỏ nhất hoặc tối đa hóa hiệu suất của mô hình.
- Tuy nhiên không đơn giản là chọn một số lần hữu hạn để tìm (weights, bias) bằng cách chạy hết tất cả các tài nguyên và mong cho một giai đoạn nào đó sẽ tìm ra số thích hợp.
- Vì thế nên phương pháp Optimizer trong huấn luyện mô hình học máy ra đời để cải thiện các tham số của mô hình theo từng bước

## 2. Các thuật toán tối ưu

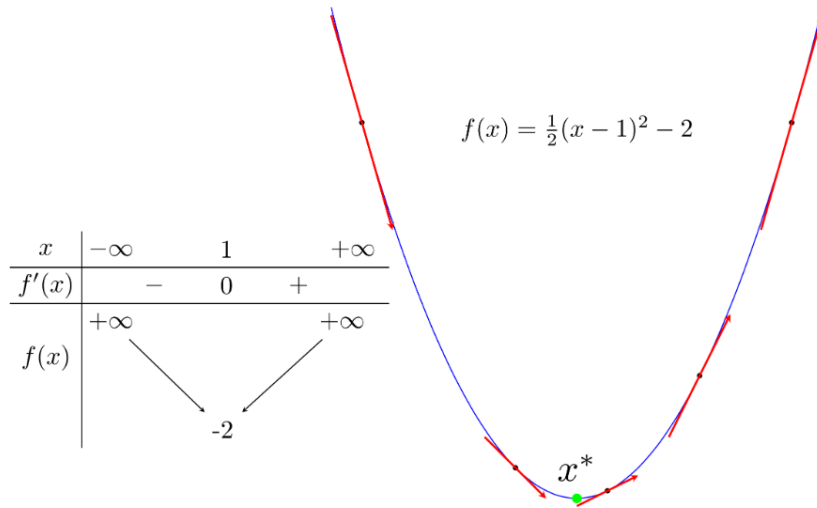
### 2.1 Gradient Descent (GD)

#### 2.1.1 Khái niệm

- Việc tìm global minimum của các hàm mất mát trong Machine Learning là phức tạp, do đó chúng ta thường tìm các điểm local minimum và coi chúng là nghiệm của bài toán. Ý tưởng của Gradient Descent có ý tưởng là khởi đầu từ một điểm gần với nghiệm và sử dụng phép toán lặp để tiến gần hơn đến điểm cần tìm, dựa trên đạo hàm của hàm số.

#### 2.1.2 Cách hoạt động

- a. Gradient Descent cho hàm 1 biến



- Cho  $x_t$  là điểm ta tìm được sau vòng lặp thứ. Ta cần tìm một thuật toán để đưa  $x_t$  về càng gần  $x^*$  càng tốt.

- Nhận xét:

- + Nếu  $f'(x_t) > 0$  :  $x_t$  nằm về bên phải so với  $x^*$  ( và ngược lại ). Để điểm tiếp theo  $x_{t+1}$  gần với  $x^*$  hơn, chúng ta cần di chuyển  $x_t$  về phía bên trái (phía âm). Nói cách khác, chúng ta sẽ di chuyển ngược dấu với dấu đạo hàm.

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

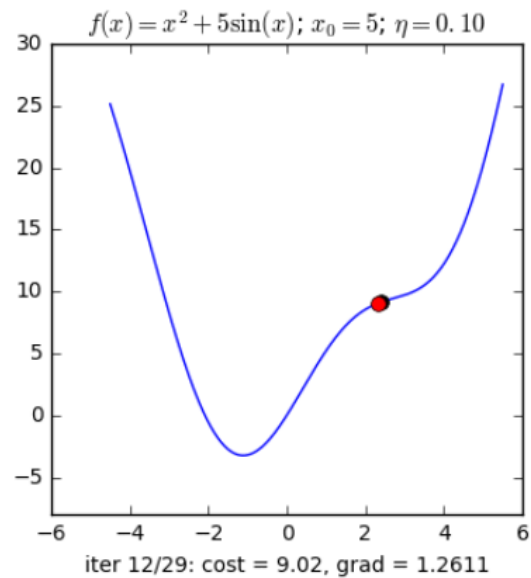
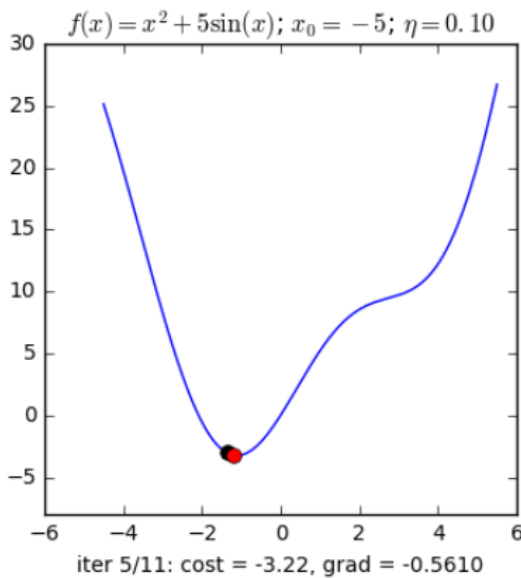
Trong đó:  $\Delta$  là một đại lượng ngược dấu với đạo hàm  $f'(x_t)$

- +  $x_t$  càng xa  $x^*$  về phía bên phải thì  $f'(x_t)$  càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển  $\Delta$  , một cách trực quan nhất, là tỉ lệ thuận với  $-f'(x_t)$

Hai nhận xét phía trên cho chúng ta một cách cập nhật đơn giản là:

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

- + Trong đó  $\eta$  (đọc là eta) là một số dương được gọi là **learning rate (tốc độ học)**. Dấu trừ thể hiện việc chúng ta phải đi ngược với đạo hàm (Đây cũng chính là lý do phương pháp này được gọi là Gradient Descent - descent nghĩa là đi ngược). Các quan sát đơn giản phía trên, mặc dù không phải đúng cho tất cả các bài toán, là nền tảng cho rất nhiều phương pháp tối ưu nói chung và thuật toán Machine Learning nói riêng.
- + Với điểm khởi tạo khác nhau như: Ngẫu Nhiên, Ở Trung Tâm Miền Giá Trị, Gần Điểm Cực Tiểu Dự Kiến, Giá Trị Cận Biên, v.v thì thuật toán của chúng ta tìm được nghiệm gần giống nhau, mặc dù với tốc độ hội tụ khác nhau



- + Learning rate khác nhau:

Learning rate đóng vai trò quan trọng trong quá trình tối ưu hóa bằng Gradient Descent. Đây là một tham số quyết định kích thước của bước cập nhật trọng số tại mỗi bước trong quá trình huấn luyện mô hình.

- Nếu learning rate lớn, mô hình có thể hội tụ nhanh hơn, nhưng cũng có nguy cơ vượt qua giá trị tối ưu.
- Nếu learning rate nhỏ, mô hình có thể hội tụ chậm, nhưng có khả năng vững chắc hơn.
- Phải thử nghiệm và điều chỉnh learning rate để đảm bảo mô hình học một cách hiệu quả và nhanh chóng.
- Kỹ thuật thường sử dụng là sử dụng giá trị learning rate nhỏ hơn và tăng dần nếu mô hình không hội tụ hoặc giảm nếu quá trình hội tụ quá chậm.

b. Gradient Descent cho hàm nhiều biến

Nếu cần tìm global minimum cho hàm  $f(\boldsymbol{\theta})$  trong đó  $\boldsymbol{\theta}$  là một vector, thường được dùng để ký hiệu tập hợp các tham số của một mô hình cần tối ưu (trong Linear Regression thì các tham số chính là hệ số  $W$ ).

Đạo hàm của hàm số đó tại một điểm  $\boldsymbol{\theta}$  bất kỳ được ký hiệu là  $\nabla f(\boldsymbol{\theta})$  (hình tam giác ngược đọc là nabla). Tương tự như hàm 1 biến, thuật toán GD cho hàm nhiều biến cũng bắt đầu bằng một điểm dự đoán  $\boldsymbol{\theta}_0$  sau đó, ở vòng lặp thứ  $t$  quy tắc cập nhật là:

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

Quy tắc cần nhớ: luôn luôn đi ngược hướng với đạo hàm

### 2.1.3 Ưu điểm

- Đơn Giản và Dễ Triển Khai:
- Hiệu Quả với Dữ Liệu Lớn:
- Phổ Biến và Linh Hoạt:

### 2.1.4 Nhược điểm

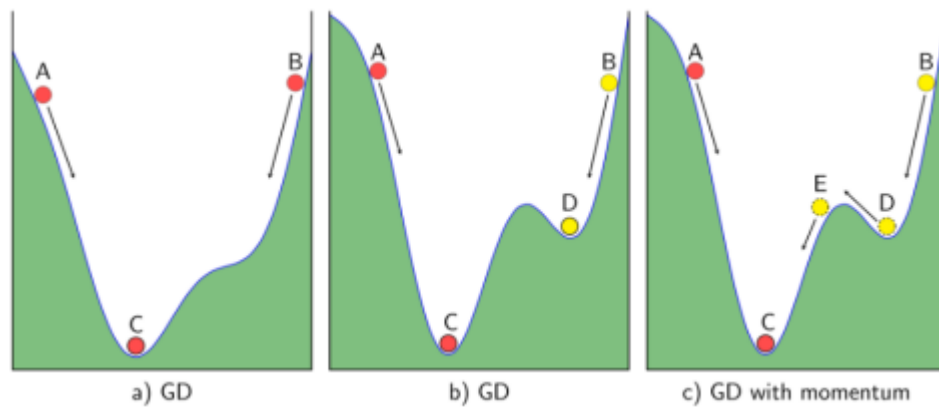
- Nhạy Cảm với Điểm Khởi Tạo:

- + Gradient Descent có thể rơi vào điểm tối thiểu cục bộ phụ thuộc vào điểm khởi tạo, và đôi khi không đạt được điểm tối ưu toàn cục.
- Yêu Cầu Chọn Tổ Lớn Learning Rate
- Chậm Hội Tụ Trong Một Số Trường Hợp:
  - + Có thể hội tụ chậm đối với một số bài toán phức tạp, đặc biệt là khi hàm mất mát có độ cong thay đổi lớn hoặc có nhiều biến.
- Nhạy Cảm với Tính Nhiễu
- Không Thích Hợp Cho Mô Hình Phi Tuyến:
  - + Có thể không thích hợp cho các mô hình phi tuyến, nơi có nhiều điểm tối thiểu cục bộ và biến thể như Momentum hoặc Adagrad thường được ưa chuộng.

## 2.2 Nesterov Momentum

### 2.2.1 Khái niệm

- Nhắc lại thuật toán Gradient Descent : Để giải bài toán tìm điểm global optimal của loss function  $J(\theta)$  với  $\theta$  là tập hợp các tham số của model.
- Dưới góc nhìn vật lý: Thuật toán GD thường được ví với tác dụng của trọng lực lên một hòn bi đặt trên một mặt có dạng như hình một thung lũng giống như hình 1a) dưới đây. Bất kể ta đặt hòn bi ở A hay B thì cuối cùng hòn bi cũng sẽ lăn xuống và kết thúc ở vị trí C.



- Tuy nhiên, nếu như bề mặt có hay đáy thung lũng như hình 1b thì tùy vào việc đặt bi ở A hay B, vị trí cuối cùng của bi sẽ ở C hoặc D. Điểm D là một điểm local minimum chúng ta không mong muốn.
- Nếu suy nghĩ một cách vật lý hơn, vẫn trong Hình 1b), nếu vận tốc ban đầu của bi khi ở điểm B đủ lớn, khi bi lăn đến điểm D, theo *đà*, bi có thể tiếp tục di chuyển lên dốc phía bên trái của D. Và nếu giả sử vận tốc ban đầu lớn hơn nữa, bi có thể vượt dốc tới điểm E rồi lăn xuống C như trong Hình 1c). Đây chính là điều chúng ta mong muốn. Bạn đọc có thể đặt câu hỏi rằng liệu bi lăn từ A tới C có theo *đà* lăn tới E rồi tới D không.

=> Kết luận: *Dựa trên hiện tượng này, một thuật toán được ra đời nhằm khắc phục việc nghiệm của GD rơi vào một điểm local minimum không mong muốn. Thuật toán đó có tên là Momentum*

### 2.2.2 Cách hoạt động

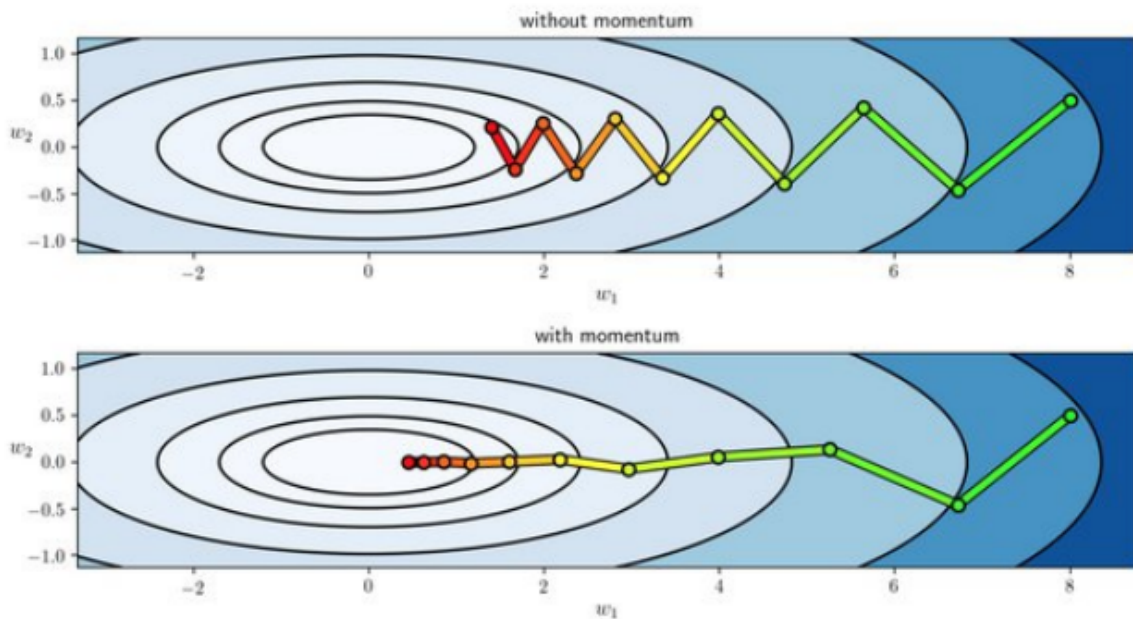
- **Gradient Descent với Momentum:** Trong GD, chúng ta cần tính lượng thay đổi ở thời điểm  $t$  để cập nhật vị trí mới cho nghiệm (tức hòn bi). Nếu chúng ta coi đại lượng này như vận tốc  $vt$  trong vật lý, vị trí mới của hòn bi sẽ là  $\theta_{t+1} = \theta_t - vt$ . Dấu trừ thể hiện việc phải di chuyển ngược với đạo hàm. Công việc của chúng ta bây giờ là tính đại lượng  $vt$  sao cho nó vừa mang thông tin của độ dốc (tức đạo hàm), vừa mang thông tin của *đà*, tức vận tốc trước đó  $vt-1$  (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:

$$vt = \gamma vt-1 + \eta \nabla \theta J(\theta)$$

- Trong đó  $\gamma$  thường được chọn là một giá trị khoảng 0.9,  $vt$  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:

$$\theta = \theta - \eta \nabla J(\theta)$$

- Thuật toán đơn giản này tỏ ra rất hiệu quả trong các bài toán thực tế (trong không gian nhiều chiều, cách tính toán cũng hoàn toàn tương tự). Dưới đây là một ví dụ trong không gian một chiều.



### 2.2.3 Ưu điểm

- Momentum giúp tăng tốc quá trình hội tụ của thuật toán tối ưu hóa. Thay vì chỉ di chuyển theo gradient hiện tại, Momentum tích lũy thông tin từ các bước trước đó để xác định hướng cập nhật. Điều này giúp vượt qua các điểm yên ngựa và tối ưu hóa nhanh hơn.
- Trong quá trình tối ưu hóa, gradient có thể dao động mạnh khi tiếp cận điểm tối ưu. Momentum giúp giảm thiểu dao động này bằng cách tích lũy thông tin từ các bước trước đó. Điều này giúp hội tụ ổn định hơn.

- Momentum giúp vượt qua các điểm yên ngựa (nơi gradient bằng không) bằng cách tích lũy động năng từ các bước trước đó. Điều này giúp tối ưu hóa tiếp tục di chuyển thay vì dừng lại tại các điểm yên ngựa.
- Momentum không yêu cầu nhiều tham số điều chỉnh và dễ dàng triển khai trong các thuật toán tối ưu hóa.

#### 2.2.4 Nhược điểm

- Mặc dù Momentum không yêu cầu nhiều tham số so với một số phương pháp khác, nhưng vẫn cần phải điều chỉnh hệ số momentum để đạt được hiệu suất tối ưu. Nếu không chọn giá trị phù hợp, có thể dẫn đến hội tụ không ổn định hoặc tăng thời gian tính toán.
- Trong một số trường hợp, Momentum có thể khiến thuật toán bị kẹt ở điểm cực tiểu cục bộ, không thể vượt qua để tìm điểm tối ưu toàn cục.
- Nếu hàm mục tiêu không mượt (không có tính chất khả vi hoặc siêu khả vi), Momentum có thể không hoạt động hiệu quả và dẫn đến các vấn đề về hội tụ.
- Trong các bài toán không đối xứng, Momentum có thể không hoạt động tốt vì nó không xem xét hướng gradient theo từng chiều riêng biệt.

### 2.3 Stochastic Gradient Descent.

#### 2.3.1 Khái niệm

- Trong thuật toán này, chúng ta tính đạo hàm của hàm mất mát dựa trên từng điểm dữ liệu riêng lẻ  $x_i$  và sau đó cập nhật  $\theta$  dựa trên đạo hàm này. Quá trình này được lặp lại cho tất cả các điểm dữ liệu trong tập huấn luyện. Mặc dù phương pháp này rất đơn giản, nhưng thực tế đã chứng minh rằng nó hoạt động 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 (chủ yếu là Deep Learning mà chúng ta sẽ thấy trong phần sau của blog) và các bài toán yêu cầu mô hình thay đổi liên tục, tức online learning.

### 2.3.2 Cách hoạt động

- 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.
  - + Một cách toán học, quy tắc cập nhật của SGD là

$$\theta = \theta - \eta \nabla_{\theta} J(\theta; \mathbf{x}_i; \mathbf{y}_i)$$

- + Trong đó:  $J(\theta; \mathbf{x}_i; \mathbf{y}_i)$  là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là  $(\mathbf{x}_i; \mathbf{y}_i)$

### 2.3.3 Ưu điểm

- SGD thích hợp và hiệu quả khi làm việc với các tập dữ liệu lớn vì chỉ cần tính toán gradient trên một mẫu dữ liệu thay vì toàn bộ tập dữ liệu.
- Với việc cập nhật trọng số sau mỗi mẫu dữ liệu, SGD có thể hội tụ nhanh chóng và thậm chí là dẫn đến giải pháp tốt nếu được cấu hình đúng.
- SGD thường phù hợp cho các mô hình phi tuyến, nơi các điểm cực tiểu cục bộ có thể được tránh.
- Đặc biệt phù hợp cho mô hình Deep Learning, nơi tập dữ liệu thường rất lớn và cần một quá trình huấn luyện hiệu quả.

### 2.3.4 Nhược điểm

- Do chỉ tính toán gradient trên một mẫu dữ liệu, SGD có thể bị ảnh hưởng bởi nhiễu và dao động nhiều hơn so với Batch Gradient Descent.

- SGD không đảm bảo hội tụ đúng điểm tối thiểu toàn cục. Thậm chí, nó có thể "nhảy" qua điểm tối ưu hoặc bị mắc kẹt ở điểm cực tiểu cục.
- Với learning rate lớn, SGD có thể vượt qua điểm cực tiểu và không hội tụ, dẫn đến việc mô hình không đạt được hiệu suất tốt.
- Do sự ngẫu nhiên trong quá trình chọn mẫu, SGD có thể mất thông tin quan trọng từ các mẫu dữ liệu và không đảm bảo mức độ biểu diễn tốt nhất của dữ liệu.
- SGD có khả năng bị mắc kẹt ở các điểm cực tiểu cục và không thể vượt qua chúng.

## 2.4 Mini-batch Gradient Descent

### 2.4.1 Khái niệm

- Với Mini-BGD, training dataset sẽ được chia thành nhiều gói (batch) khác nhau để xử lý. Sau khi tất cả dữ liệu trong batch, cost sẽ được tính toán và model sẽ được cập nhật tương ứng.
- Với cách tiếp cận này, tần suất cập nhật của model sẽ không quá nhiều như SGD và lượng dữ liệu cần phải xử lý trong một lần cũng nhỏ hơn SGD. Với khả năng cân bằng cùng việc cho deep learning. Tuy nhiên, chúng ta cần lưu ý rằng việc tùy chỉnh batch-size sẽ ảnh hưởng trực tiếp đến kết quả.

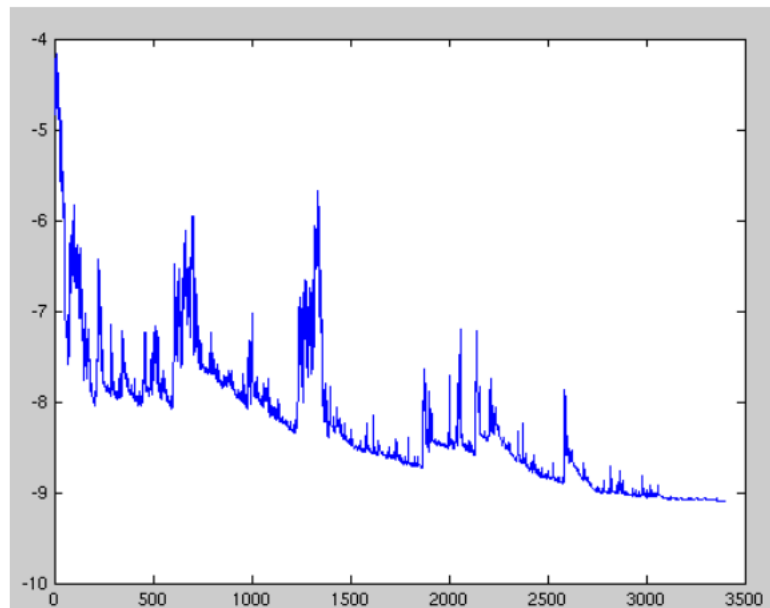
### 2.4.2 Cách hoạt động

- Thuật toán này bắt đầu mỗi epoch bằng cách xáo trộn ngẫu nhiên dữ liệu và chia toàn bộ dữ liệu thành các mini-batch, mỗi mini-batch chứa  $n$  điểm dữ liệu (trừ mini-batch cuối có thể có số điểm ít hơn  $n$  nếu  $N$  không chia hết cho  $n$ ). Mỗi lần cập nhật, thuật toán này lấy ra một mini-batch để tính toán đạo hàm và cập nhật các tham số. Công thức của thuật toán có thể được biểu diễn như sau:

$$\theta = \theta - \eta \nabla_{\theta} J(\theta; \mathbf{x}_{i:i+n}; \mathbf{y}_{i:i+n})$$

- Trong đó  $x_{i:i+n}$  được hiểu là dữ liệu thứ  $i$  tới thứ  $i + n + 1$
- Dữ liệu này sau mỗi epoch là khác nhau vì chúng cần được xáo trộn. Một lần nữa, các thuật toán khác cho GD như Momentum, ... cũng có thể được áp dụng vào đây Mini-batch GD được sử dụng trong hầu hết các thuật toán Machine Learning, đặc biệt là trong Deep Learning. Giá trị  $n$  thường được chọn là khoảng từ 50 đến 100.

Ví dụ về giá trị của hàm mất mát mỗi khi cập nhật tham số  $\theta$  của một bài toán khác phức tạp hơn.



Hàm mất mát *nhảy lên nhảy xuống* (fluctuate) sau mỗi lần cập nhật nhưng nhìn chung giảm dần và có xu hướng hội tụ về cuối. (Nguồn: [Wikipedia](#)).

### 2.4.3 Ưu điểm

- Mini-Batch Gradient Descent kết hợp ưu điểm của cả Batch Gradient Descent và Stochastic Gradient Descent, làm cho nó hiệu quả khi làm việc với dữ liệu lớn.

- Phù hợp cho các bài toán lớn mà không yêu cầu toàn bộ tập dữ liệu được tải vào bộ nhớ.
- Mini-Batch GD giúp giảm áp lực trên bộ nhớ so với Batch GD, vì chỉ cần lưu một lượng nhỏ dữ liệu.

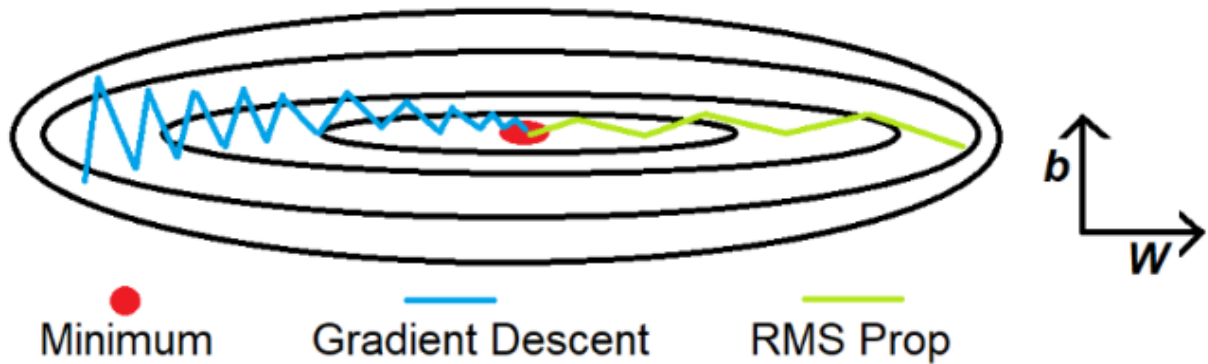
#### 2.4.4 Nhược điểm

- Vẫn có khả năng rơi vào điểm cực tiểu cục bộ tùy thuộc vào sự chọn lựa của kích thước mini-batch và learning rate.
- Có thể vượt qua điểm tối ưu nếu learning rate lớn và không được kiểm soát.
- Cần điều chỉnh thêm hyperparameters như learning rate và kích thước mini-batch để đạt được hiệu suất tốt nhất.
- Với kích thước mini-batch nhỏ, có thể mất đi thông tin quan trọng từ một số mẫu dữ liệu.
- Có thể bị ảnh hưởng bởi nhiễu nếu kích thước mini-batch quá nhỏ và dữ liệu nhiễu.

## 2.5 Root Mean Square Propagation (RMSProp)

### 2.5.1 Khái niệm

RMSprop tính toán một trung bình có trọng số của bình phương của các đạo hàm gradient. Nó sử dụng giá trị này để điều chỉnh kích thước của bước cập nhật cho mỗi trọng số. Mục tiêu là giảm độ lớn của gradient cho các trọng số có đạo hàm lớn và tăng độ lớn cho các trọng số có đạo hàm nhỏ. Điều này giúp giảm tốc độ học cho các trọng số có gradient lớn và tăng tốc độ học cho các trọng số có gradient nhỏ, giúp mô hình hội tụ nhanh chóng hơn.



### 2.5.2 Cách hoạt động

- Tính gradient  $g_t$  của hàm mất mát tại vị trí hiện tại
- Tính toán squared gradient  $v_t = \beta v_{t-1} + (1 - \beta)g_t^2$ . Trong đó,  $\beta$  là hệ số trung bình động ( $0 < \beta < 1$ )
- Cập nhật trọng số  $\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot g_t$ . Trong đó,  $\alpha$  là tỉ lệ học và  $\epsilon$  là giá trị nhỏ để tránh chia cho 0

### 2.5.3 Ưu điểm

- RMSProp thường đạt hiệu suất tốt và ổn định trên nhiều bài toán, đặc biệt là khi có sự không đồng đều về độ chính xác của các tham số.
- Phù hợp cho nhiều loại mô hình và dữ liệu khác nhau, làm giảm thiểu vấn đề về tỷ lệ học không đồng nhất.
- RMSProp có khả năng tự động điều chỉnh tỷ lệ học dựa trên biến trung bình động của squared gradient, giúp tối ưu hóa hiệu suất.

### 2.5.4 Nhược điểm

- Như nhiều thuật toán tối ưu hóa khác, RMSProp cần sự điều chỉnh cẩn thận của các hyperparameters như tỷ lệ học và hệ số trung bình động.
- Có thể mất thông tin quan trọng nếu tỷ lệ học được điều chỉnh quá mạnh.

- Có thể bị mắc kẹt ở các điểm cực tiểu cục và không thể vượt qua chúng trong một số trường hợp.

## CHƯƠNG II: CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY

### 1. Continual learning

#### 1.1 Khái niệm

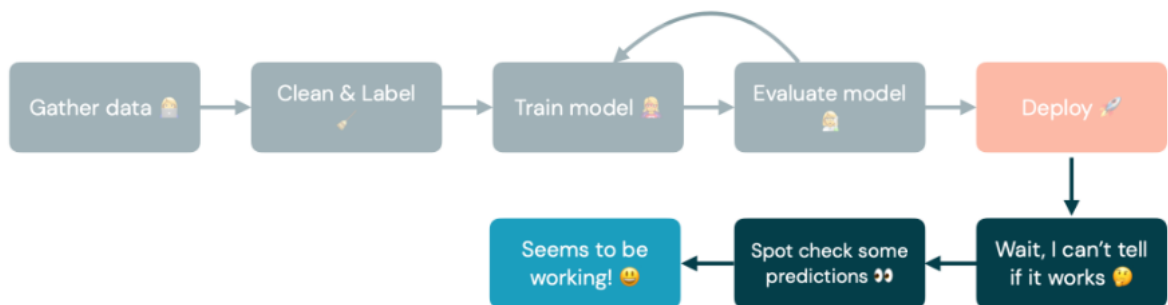
- Đúng với tên gọi của mô hình, học liên tục có nghĩa là mô hình sẽ tự phát triển dữ liệu mới để học và không quên kiến thức dữ liệu cũ, cũng có thể nói kiến thức cũ cũng là nền tảng để mô hình học và phát triển.

#### 1.2 Xây dựng model



Ban đầu, ta sẽ thu thập, làm sạch, và gán nhãn một số dữ liệu. Chúng ta huấn luyện một mô hình trên dữ liệu đó. Sau đó, chúng ta đánh giá mô hình và quay lại huấn luyện mô hình để cải thiện nó dựa trên đánh giá của chúng ta. Cuối cùng, chúng ta có được một mô hình tối thiểu khả dụng và triển khai nó.

#### 1.3 Triển khai và kiểm tra



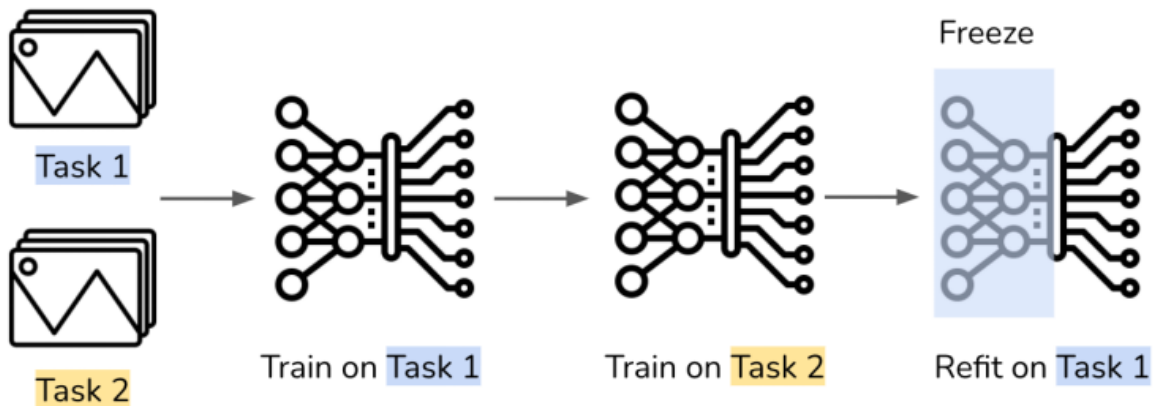
Vấn đề bắt đầu sau khi triển khai mô hình, thường không có một cách tốt để đo lường cách mô hình của chúng ta thực sự hoạt động trong sản xuất mà chỉ kiểm tra một số dự

đoán để xem liệu chúng có hoạt động theo cách mà chúng ta mong muốn không. Nếu nó có vẻ hoạt động, thì tốt.

#### 1.4 Vấn đề Catastrophic forgetting

##### 1.4.1 *Catastrophic forgetting là gì*

Đây là thách thức lớn nhất trong Continual Learning. Catastrophic forgetting xảy ra khi một hệ thống học máy quên đi kiến thức đã học trước đó khi được đào tạo trên các nhiệm vụ mới.



##### 1.4.2 Nguyên nhân

- Interference là khi dữ liệu mới xảy ra "xung đột" với dữ liệu cũ, khiến hệ thống khó phân biệt các mô hình.
- Mạng có thể không có đủ "chỗ" để lưu trữ cả kiến thức cũ và mới, dẫn đến việc dữ liệu cũ bị ghi đè.
- Các thuật toán tối ưu thường thiên về học tập các nhiệm vụ mới, bỏ qua các nhiệm vụ cũ.

##### 1.4.3 Giải pháp

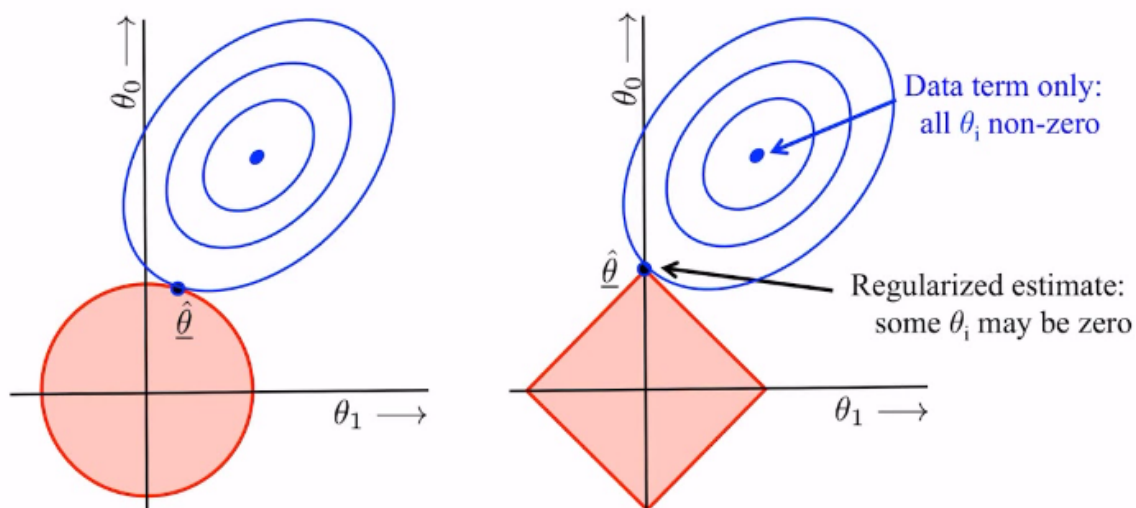
Để giải quyết các vấn đề này, các phương pháp như "Regularization" "Rehearsal" "Memory replay" đã được phát triển để giảm thiểu hiện tượng Catastrophic forgetting trong quá trình học máy.



### 1.4.3.1 Regularization

Regularization là một kỹ thuật phổ biến được sử dụng trong học máy để giảm thiểu overfitting. Trong trường hợp của CL, regularization có thể được sử dụng để ngăn chặn các trọng số của mạng thay đổi quá nhiều khi học các nhiệm vụ mới.

- L2 regularization: Thêm một hàm phạt vào hàm mục tiêu để giảm thiểu tổng bình phương của các trọng số.
- L1 regularization: Thêm một hàm phạt vào hàm mục tiêu để giảm thiểu tổng tuyệt đối của các trọng số.
- Dropout: Ngẫu nhiên loại bỏ một số kết nối giữa các neuron trong quá trình đào tạo.



### 1.4.3.2 Rehearsal

- Rehearsal là một kỹ thuật đơn giản nhưng hiệu quả để giảm thiểu CF. Kỹ thuật này bao gồm việc giữ lại một phần dữ liệu cũ và định kỳ "ôn lại" dữ liệu này khi đào tạo trên các nhiệm vụ mới.
- Khi ôn lại dữ liệu cũ, hệ thống có thể cập nhật các trọng số của mình để phù hợp với dữ liệu cũ. Điều này giúp hệ thống duy trì kiến thức đã học được trước đó.

#### 1.4.3.3 Memory replay

- Là một kỹ thuật tương tự như rehearsal, nhưng thay vì giữ lại dữ liệu cũ, hệ thống sẽ lưu trữ các trọng số của mình liên quan đến các nhiệm vụ cũ.
- Khi đào tạo trên các nhiệm vụ mới, hệ thống sẽ sử dụng các trọng số cũ để khởi tạo quá trình học tập. Điều này giúp hệ thống bắt đầu từ một điểm tốt hơn và giảm thiểu CF.

#### 1.5 Ứng dụng Continual learning

- Continual learning (CL) là một lĩnh vực học máy mới nổi, với tiềm năng cách mạng hóa cách chúng ta đào tạo và triển khai các hệ thống học máy. CL cho phép các hệ thống học máy liên tục học hỏi từ luồng dữ liệu theo thời gian, thích nghi và cải thiện kiến thức của chúng mà không quên đi những gì đã học được trước đó.
- CL có nhiều ứng dụng tiềm năng trong nhiều lĩnh vực như các ví dụ sau:
  - Một robot có thể được đào tạo để thực hiện một nhiệm vụ, chẳng hạn như bắt bóng. Khi robot thực hiện nhiệm vụ, nó có thể học hỏi từ những sai lầm của mình và cải thiện hiệu suất. Robot cũng có thể học các kỹ năng mới, chẳng hạn như bắt bóng trong môi trường khác nhau.
  - Một mô hình ngôn ngữ có thể được đào tạo để dịch từ tiếng Anh sang tiếng Pháp. Khi mô hình được tiếp xúc với nhiều dữ liệu dịch, nó có thể học thêm ngôn ngữ mới, chẳng hạn như tiếng Tây Ban Nha. Mô hình cũng có thể học các lĩnh vực mới, chẳng hạn như dịch y tế.
  - Một hệ thống đề xuất có thể được đào tạo để đề xuất sản phẩm cho người dùng. Khi hệ thống nhận được phản hồi từ người dùng, nó có thể học hỏi từ sở thích của người dùng và cải thiện các đề xuất. Hệ thống cũng có thể học các xu hướng mới, chẳng hạn như sản phẩm mới hoặc xu hướng thời trang.

## 2. Test Production

2.1 Khái niệm Có bốn loại kiểm thử chính được sử dụng tại các giai đoạn khác nhau trong chu kỳ phát triển

- Test Production là quá trình sản xuất các tập kiểm thử để đảm bảo rằng mô hình học máy đang được triển khai hoạt động đúng đắn và hiệu quả trên dữ liệu thực tế. Điều này là quan trọng để đảm bảo rằng mô hình không chỉ hoạt động tốt trên tập dữ liệu đào tạo mà còn đảm bảo tính tổng quát và hiệu suất trên dữ liệu mới.

2.2 Các phân loại kiểm thử

Có 5 loại kiểm thử chính được sử dụng tại các giai đoạn khác nhau trong chu kỳ phát triển

1. Kiểm Thử Đơn Vị - Unit Tests:

Kiểm thử trên mỗi thành phần, mỗi thành phần có một nhiệm vụ duy nhất (ví dụ: hàm lọc danh sách).

2. Kiểm Thử Tích Hợp - Integration Tests

Kiểm thử trên chức năng kết hợp của các thành phần riêng lẻ (ví dụ: xử lý dữ liệu).

3. Kiểm Thử Hệ Thống - System Tests

Kiểm thử trên thiết kế của hệ thống để đảm bảo kết quả đầu ra dự kiến cho đầu vào được cung cấp (ví dụ: huấn luyện, suy luận, v.v.).

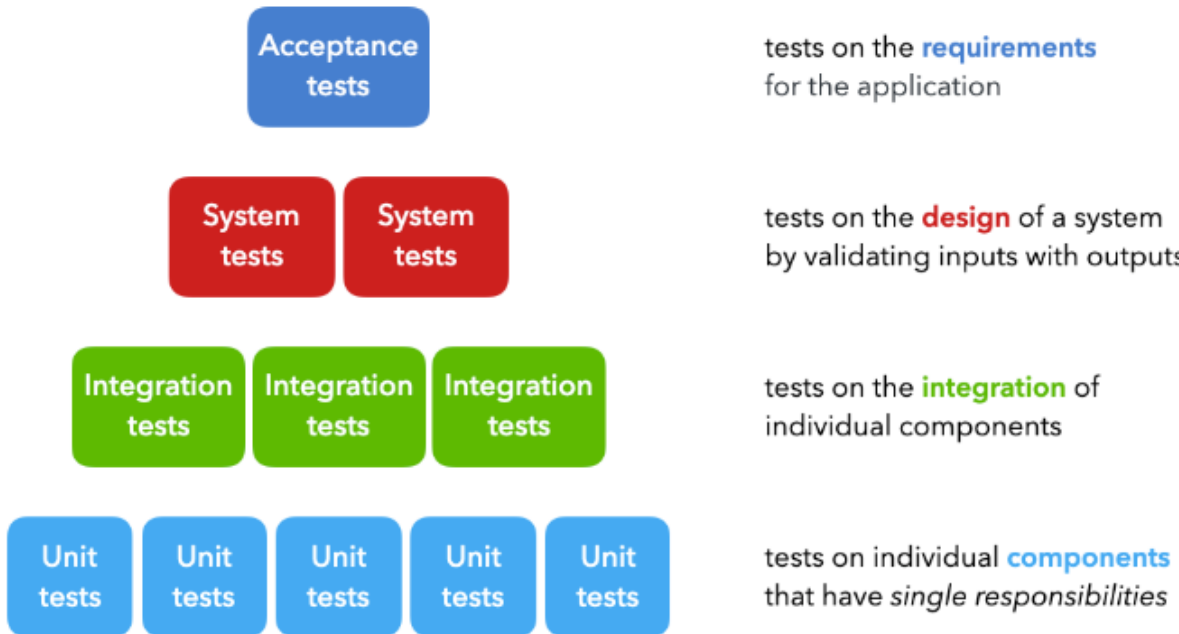
4. Kiểm Thử Chấp Nhận - Acceptance Tests

Kiểm thử để xác nhận rằng các yêu cầu đã được đáp ứng, thường được gọi là Kiểm Thử Chấp Nhận Người Dùng - UAT - User Acceptance Testing

5. Kiểm Thử Hồi Quy - Regression Tests

Kiểm thử dựa trên các lỗi chúng ta đã thấy trước đó để đảm bảo rằng các thay đổi mới không đưa chúng trở lại.

Trong khi hệ thống Học Máy có tính chất xác suất, chúng được tạo thành từ nhiều thành phần xác định có thể được kiểm thử theo cách tương tự như các hệ thống phần mềm truyền thống. Sự phân biệt giữa việc kiểm thử hệ thống Học Máy bắt đầu khi chúng ta chuyển từ việc kiểm thử mã nguồn sang việc kiểm thử dữ liệu và mô hình.



### 2.3 Cách thức thực hiện test

#### 1. Chuẩn Bị Dữ Liệu Kiểm Thử:

Tập trung vào việc lựa chọn và chuẩn bị dữ liệu kiểm thử đại diện cho môi trường triển khai thực tế.

- inputs: data types, format, length, edge cases (min/max, small/large, etc.)
- outputs: data types, formats, exceptions, intermediary and final outputs

#### 2. Kiểm Thử Mô Hình:

Áp dụng mô hình lên tập kiểm thử và thu thập thông tin về hiệu suất, độ chính xác, và các thước đo hiệu suất khác.

#### 3. Phân Tích Kết Quả:

Phân tích kết quả kiểm thử để đảm bảo rằng mô hình hoạt động đúng đắn và đáp ứng yêu cầu của dự án.

#### 4. Tối Ưu Hóa và Điều Chỉnh:

Dựa vào kết quả kiểm thử, có thể cần thực hiện tối ưu hóa mô hình và điều chỉnh hyperparameters để cải thiện hiệu suất.

#### 5. Tự Động Hóa Quy Trình Kiểm Thử:

Tự động hóa quy trình kiểm thử để đảm bảo rằng quy trình này có thể được thực hiện một cách liên tục và hiệu quả.

### 2.4 Những thách thức của Test Production

- **Tính Tương Thích:** Sản phẩm của bạn có thể phải chạy trên nhiều nền tảng, trình duyệt, thiết bị và hệ điều hành khác nhau. Đảm bảo tính tương thích có thể là một thách thức, đặc biệt là khi có các phiên bản mới của hệ điều hành hay trình duyệt xuất hiện.
- **Kiểm Thử Hiệu Năng:** Đôi khi, sản phẩm phải đối mặt với nhiều người sử dụng cùng một lúc. Kiểm thử hiệu năng giúp đảm bảo rằng sản phẩm của bạn có thể xử lý tải trọng cao mà không giảm hiệu suất.
- **Bảo Mật:** Bảo mật là một vấn đề lớn khi đưa sản phẩm ra thị trường. Phải đảm bảo rằng không có lỗ hổng bảo mật nào có thể bị tận dụng để tấn công hệ thống hay lấy thông tin người dùng.
- **Kiểm Thử Tự Động:** Việc triển khai kiểm thử tự động có thể là một thách thức. Tuy nhiên, nếu triển khai đúng cách, nó có thể giúp tiết kiệm thời gian và giảm lỗi.
- **Quản lý Dự Án:** Quản lý tiến độ và tương tác giữa các nhóm là quan trọng để đảm bảo rằng quá trình test production diễn ra mượt mà. Điều này bao gồm cả việc quản lý tài nguyên và lịch trình.

- Kiểm Thử Tích Hợp: Khi có nhiều phần mềm hay thành phần phải hoạt động cùng nhau, việc kiểm thử tích hợp để đảm bảo sự tương tác và tương thích là quan trọng.
- Phản Hồi Người Dùng: Đôi khi, việc thu thập và xử lý phản hồi từ người dùng có thể là một thách thức. Điều này đòi hỏi sự linh hoạt để nhanh chóng đáp ứng và cải thiện sản phẩm.

## TÀI LIỆU THAM KHẢO

### 1. Tiếng Việt

- <https://csdlkhoahoc.hueuni.edu.vn/data/2021/5/BaiDangHoiThao.pdf>
- [https://mmlab.uit.edu.vn/tutorials/ml/deep-learning/fine\\_tune\\_model\\_part2](https://mmlab.uit.edu.vn/tutorials/ml/deep-learning/fine_tune_model_part2)

### 2. Tiếng Anh

- <https://madewithml.com/courses/mlops/testing/>