TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

NGUYỄN THỊ MỸ HÂN – 52100196

BÁO CÁO CUỐI KỲ

NHẬP MÔN HỌC MÁY

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

NGUYỄN THỊ MỸ HÂN – 52100196

BÁO CÁO CUỐI KỲ

NHẬP MÔN HỌC MÁY

Người hướng dẫn

PGS.TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy Lê Anh Cường đã tận tình giảng dạy trong thời gian qua. Thầy đã truyền đạt những kiến thức bổ ích và mới lạ, góp phần giúp đỡ chúng em rất nhiều để hoàn thành tốt bài báo cáo cuối kỳ.

TP. Hồ Chí Minh, ngày 22 tháng 12 năm 2023

Tác giả

Nguyễn Thị Mỹ Hân

TÓM TẮT

Tìm hiểu về các phương pháp Optimizer và cách hoạt động của chúng, có kèm ví dụ minh họa.

So sánh ưu điểm và nhược điểm của các phương pháp đã liệt kê.

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

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

ABSTRACT

Learn about various Optimizer methods and their functioning, accompanied by illustrative examples. Compare the advantages and disadvantages of the listed methods.

Explore Continual Learning in constructing a machine learning solution to solve a particular problem.

Study Test Production when developing a machine learning solution to address a specific problem.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC CÁC CHỮ VIẾT TẮT

GD	Gradient Descent
SGD	Stochastic Gradient Descent
RMSProp	Root Mean Square Propagation
Adam	Adaptive Moment Estimation

1. CÁC PHƯƠNG PHÁP OPTIMIZER

1.1 Optimizer là gì?

- Optimizer (một thuật toán tối ưu hóa) là một thuật ngữ 1hem để chỉ các thuật toán, phương pháp hay công cụ được sử dụng để cập nhật trọng số của một mô hình học máy trong quá trình huấn luyện.
- Khi huấn luyện mô hình, chúng ta thường cần điều chỉnh các tham số của mô hình sao cho hàm mất mát (loss function) đạt giá trị nhỏ nhất hoặc tối đa hóa hiệu suất của mô hình. Optimizer chịu trách nhiệm cập nhật các tham số này dựa trên gradient (độ dốc – thường được đề cập đến là đạo hàm) của hàm mất mát.

1.2 Các phương pháp Optimizer phổ biến

1.2.1 Gradient Descent (GD)

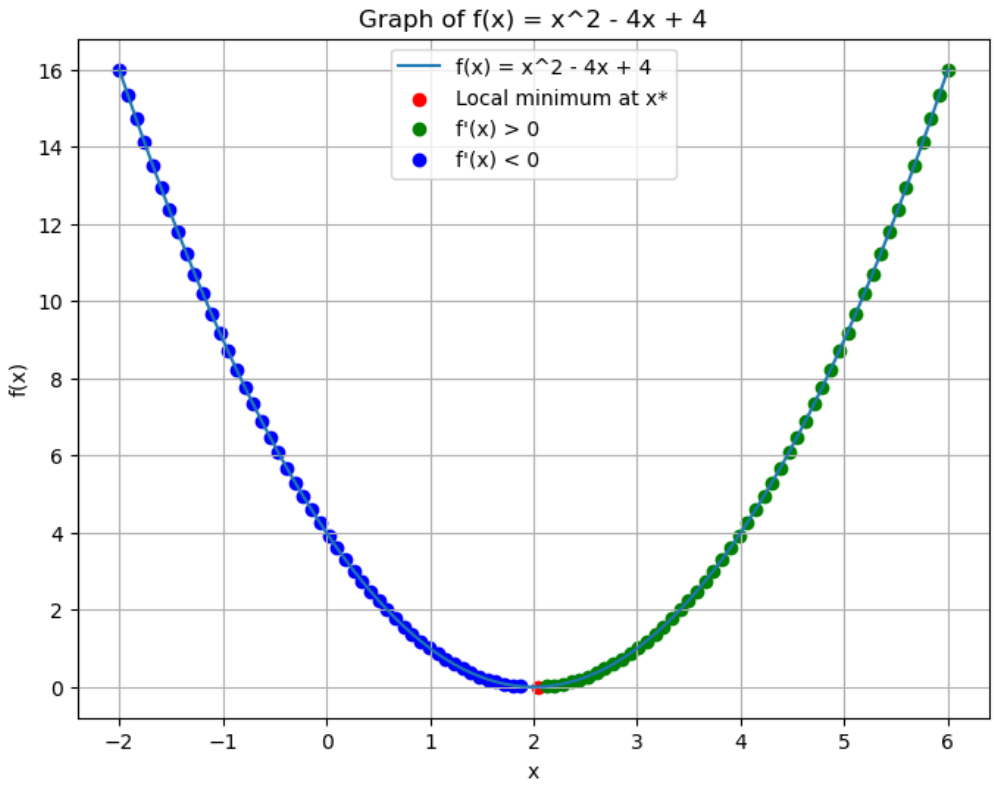
1.2.1.1 Giới thiệu

- Nhìn chung, việc tìm global minimum của các loss function trong Machine Learning là rất phức tạp. 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.
- Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta 1hem1à *gần* với nghiệm của bài toán, sau đó 1hem một phép toán lặp để *tiến dần* đến điểm cần tìm, tức đến khi đạo hàm gần với 0.
- Gradient Descent (GD) là một phương pháp được sử dụng khá phổ biến. Thuật toán tối ưu hóa này sử dụng phép tính vi phân để liên tục điều chỉnh các giá trị và đạt được điểm cực tiểu cục bộ.

1.2.1.2 Cách hoạt động của Gradient Descent

- Bắt đầu với một giá trị ban đầu cho các tham số: Gradient Descent bắt đầu với một bộ các tham số khởi tạo ngẫu nhiên hoặc được chọn một cách xác định trước.
- Tính toán gradient: Sau đó, nó sẽ tính gradient của hàm mất mát theo từng tham số. Gradient cho biết hướng và độ lớn mà hàm mất mát 2hem nhanh nhất khi thay đổi các tham số.
- Cập nhật các tham số: Gradient Descent sử dụng gradient tính được để di chuyển các tham số theo hướng giảm của hàm mất mát. Nó cập nhật các tham số bằng cách di chuyển một khoảng nhỏ (được gọi là learning rate) ngược với hướng của gradient.
- Lặp lại quá trình: lặp lại cho đến khi hàm mất mát đạt được giá trị cực tiểu cục bộ hoặc khi đạt đến số lần lặp cụ thể đã xác định trước.

1.2.1.3 Công thức cho hàm 1 biến



Hình 1: đồ thị hàm 1 biến f(x) = $x^2 - 4x + 4$

- Giả sử x_t là điểm ta tìm được sau vòng lặp thứ t.
- Nếu $f'(x_t) > 0$ (các điểm màu xanh lá) : x_t nằm bên phải x^* (điểm màu đỏ). Để điểm tiếp theo là x_{t+1} càng gần về x^*, cần di chuyển x_t về phía bên trái, tức là về phía âm, $f'(x_t)$ càng giảm đến gần về 0. Nói các khác, **chúng ta cần di chuyển ngược dấu với đạo hàm.**
- Nếu $f'(x_t) < 0$ (các điểm màu xanh dương) : x_t nằm bên trái x^* (điểm màu đỏ). Để điểm tiếp theo là x_{t+1} càng gần về x^*, cần di chuyển x_t về phía bên phải, tức là về phía dương, $f'(x_t)$ càng tăng đến khi gần về 0. Nói các khác, **chúng ta cần di chuyển ngược dấu với đạo hàm.**
- Từ 2 nhận xét trên, ta có cách cập nhật đơn giản là:

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

1.2.1.4 Công thức tổng quát cho hàm nhiều biến.

- Công thức trên được xây dựng để cập nhật lại nghiệm sau mỗi vòng lặp .

$$\theta_{j+1} := \theta_j - \alpha . \frac{\partial}{\partial \theta_j} J(\theta)$$

- θ_j : tham số cần cập nhật
- α : learning rate, xác định bước đi cập nhật.
- $\frac{\partial}{\partial \theta_j} . J(\theta)$: đạo hàm riêng của hàm số $J(\theta)$ theo tham số θ_j, được tính dựa trên công thức của hàm $J(\theta)$.

1.2.1.5 Công thức Gradient Descent cho loss function của bài toán Linear Regression

- Hàm mất mát:

$$\text{L} = \frac{1}{2N} \sum_{i=1}^{N} (f(X_i) - y_i)^2)$$

$$= \frac{1}{2N} \sum_{i=1}^{N} ((w_0 + w_1 . x_{i1} + \cdots + w_k . x_{ik}) - y_i)^2$$

- Khởi tạo:

$$\text{W} = (w_0, w_1, \dots, w_k)$$

- Cập nhật các trọng số w_t theo công thức Gradient Descent, mục đích là tìm các tham số w sao cho giá trị loss là nhỏ nhất:

$$w_t := w_t - \alpha . \frac{\partial L}{\partial w_t}$$

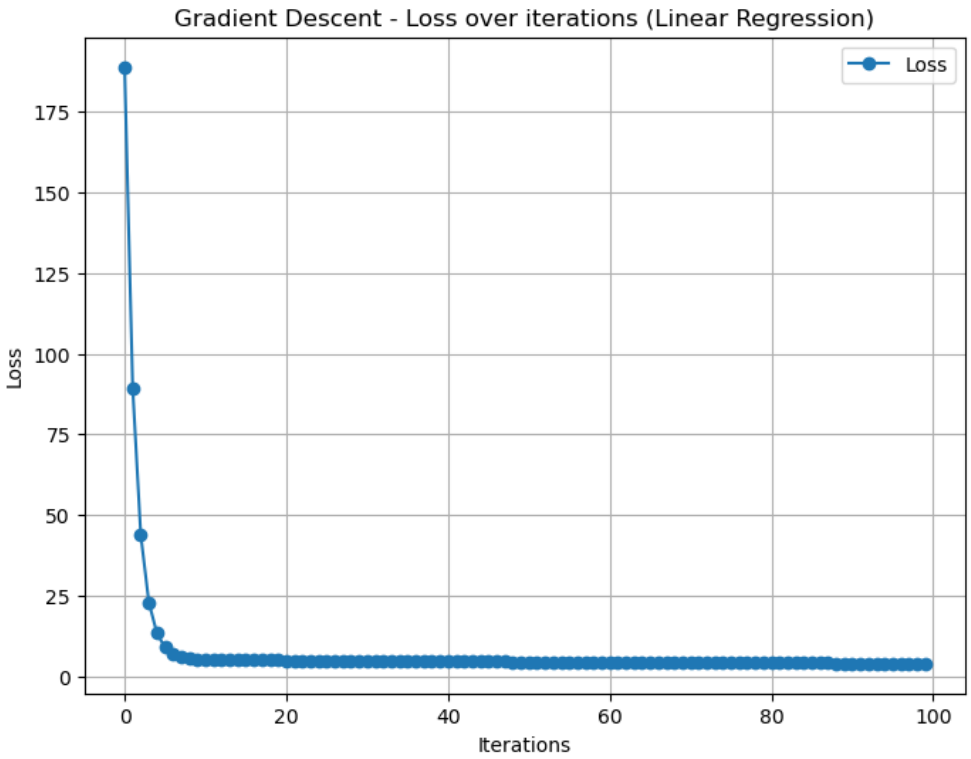
- Đạo hàm của hàm mất mát:

$$\frac{\partial L}{\partial w_t} = \frac{1}{N} \sum_{i=1}^{N} \left(f(X_i) - y_i \right) . x_{it})$$

  - Với X_i, y_i là tập training
  - $X_i = (x_{i1}, x_{i2}, \dots x_{ik})$
  - $y_i = f(X_i) = w_0 . x_{i0} + w_1 . x_{i1} + \dots + w_k . x_{ik}$

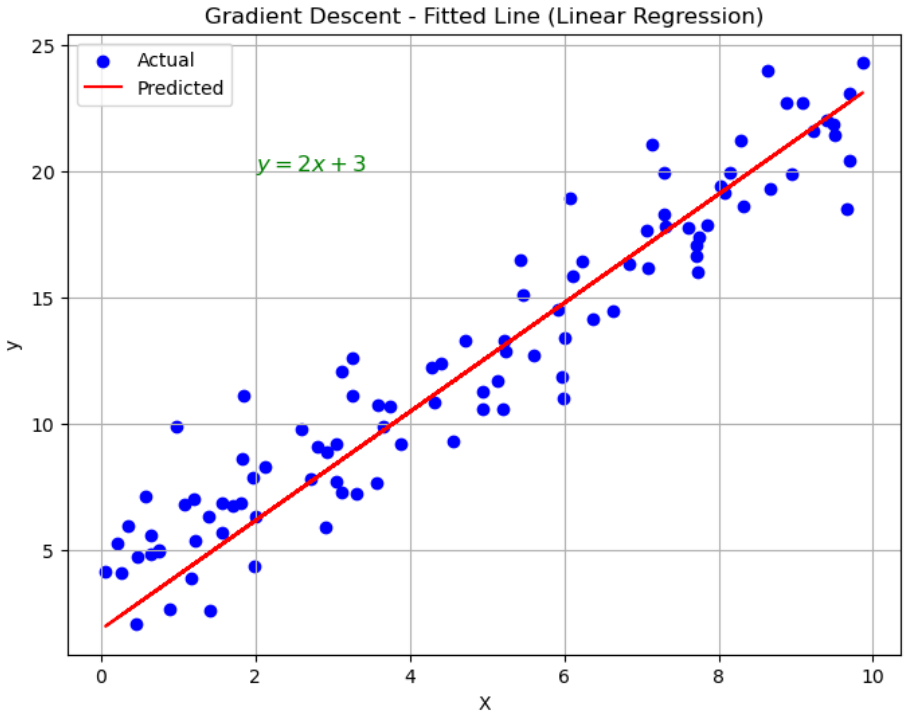
1.2.1.6 Ví dụ

- Áp dụng GD cho hàm tuyến tính y = 2x+3.ε



Hình 2: Giá trị loss sau mỗi vòng lặp

- Đồ thị này biểu diễn sự thay đổi của hàm mất mát (loss function) qua các vòng lặp của Gradient Descent trong quá trình huấn luyện mô hình Linear Regression. Nó sẽ hiển thị sự giảm dần của loss function khi thuật toán học được từ dữ liệu.



Hình 3: đừng dự đoán (predict line) từ Gradient Descent

- Đồ thị này thể hiện dữ liệu thực tế (điểm màu xanh) và đường dự đoán (đường màu đỏ) mà mô hình Linear Regression đã học được từ dữ liệu.

1.2.2 Stochstic Gradient Descent (SGD)

1.2.2.1 Giới thiệu

- Gradient Descent có hạn chế đối với dữ liệu lớn vì việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh. Và GD không phù hợp với online learning tức là khi dữ liệu được cập nhật liên tục (bài toán online learning) vì mỗi lần thêm dữ liệu ta phải tính lại đạo hàm trên toàn bộ dữ liệu => thời gian tính toán lâu.

- Để giải quyết vấn đề đó, chúng ta có phương pháp là Stochastic Gradient Descent, đây là một biến thể của phương pháp Gradient descent. Thuật ngữ "stochastic" có nghĩa là ngẫu nhiên. Trong SGD, thay vì lấy toàn bộ tập dữ liệu cho mỗi vòng lặp, chúng ta ngẫu nhiên chọn các batch dữ liệu.
- Batch là một phần của tập dữ liệu được chia nhỏ thành các nhóm nhỏ hơn để huấn luyện mô hình. Mỗi batch chứa một số lượng mẫu (samples) được sử dụng để cập nhật trọng số của mô hình trong mỗi vòng lặp huấn luyện.
- Điểm khác biệt so với phương pháp GD là SGD chỉ sử dụng một batch nhỏ mỗi lần cập nhật trọng số thay vì toàn bộ tập dữ liệu. Điều này giúp giảm thiểu thời gian tính toán và có thể tối ưu hóa tốt hơn đối với dữ liệu lớn, mặc dù nó có thể có độ chính xác thấp hơn trong mỗi bước cập nhật.

1.2.2.2 Cách hoat động:

- Khởi tạo trọng số: Bắt đầu với một bộ trọng số ngẫu nhiên hoặc được chọn trước đó.
- Chia dữ liệu thành các batch: Tập dữ liệu huấn luyện được chia thành các batch nhỏ. Mỗi batch chứa một số lượng mẫu dữ liệu (samples).
- Lặp qua các batch: SGD lặp qua từng batch trong tập dữ liệu. Mỗi lần lặp, nó chọn một batch ngẫu nhiên và sử dụng nó để cập nhật trọng số của mô hình.
- Tính toán gradient: Tại mỗi bước, gradient của hàm mất mát được tính dựa trên các mẫu trong batch hiện tại.
- Cập nhật trọng số: Sau khi tính toán gradient, trọng số của mô hình được cập nhật bằng cách di chuyển theo hướng ngược với gradient để giảm thiểu hàm mất mát.
- Lặp lại quá trình: Quá trình cập nhật trọng số thông qua các batch tiếp tục được lặp lại cho đến khi một điều kiện dừng được đáp ứng (ví dụ: số lần lặp, hội tụ đủ).
- Lưu ý:

- Tại mỗi bước cập nhật, gradient của hàm mất mát chỉ được tính dựa trên một điểm dữ liệu duy nhất x_i. Đây là cách tiếp cận giúp cập nhật tham số θ dựa trên từng điểm dữ liệu trong toàn bộ tập huấn luyện.
- Mỗi lần duyệt qua toàn bộ dữ liệu được gọi là một epoch. Trong SGD, mỗi epoch thực hiện N lần cập nhật θ với N là số điểm dữ liệu. Như vậy quá trình thực hiện mỗi epoch có thể chậm nhưng SGD thường chỉ gần số epoch nhỏ và giảm khi có dữ liệu mới.
- Xần xáo trộn thứ tự của dữ liệu sau mỗi epoch để đảm bảo tính ngẫu nhiên. Việc này ảnh hưởng đến hiệu suất của SGD.

1.2.2.3 Công thức của Stochastic Gradient Descent

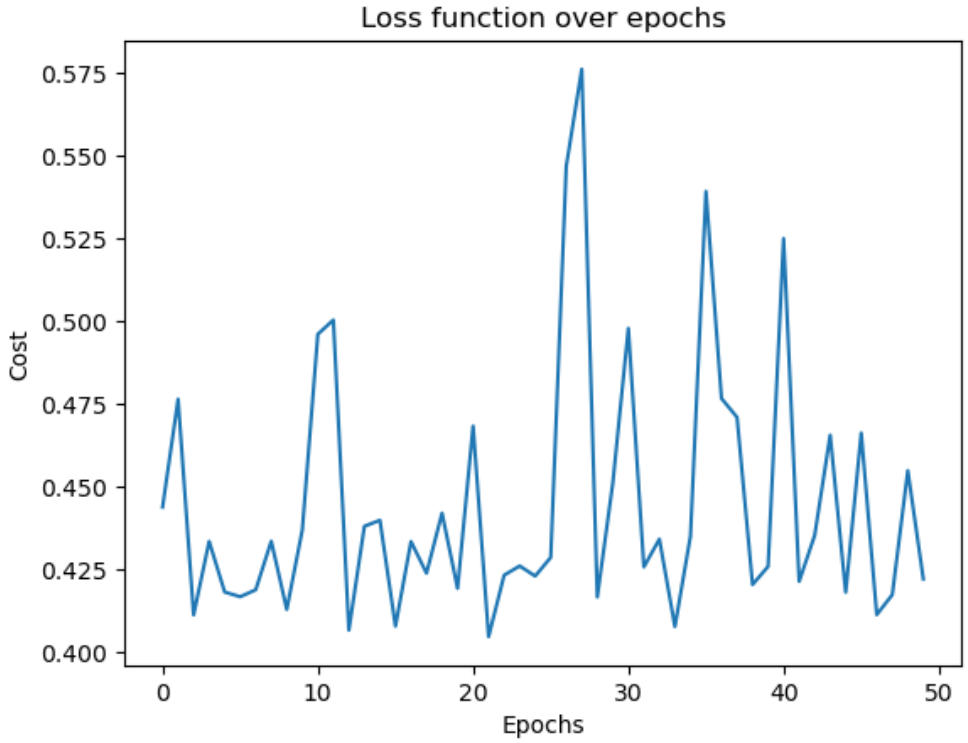
- Công thức này áp dụng cập nhật trọng số dựa trên gradient tính toán từ một batch nhỏ dữ liệu. Mỗi lần cập nhật, trọng số của mô hình được điều chỉnh theo hướng ngược với gradient để giảm thiểu hàm mất mát. Việc lặp lại quá trình này thông qua từng batch giúp mô hình học được cách điều chỉnh trọng số để tối ưu hóa hàm mất mát.

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

- θ: trọng số cần được cập nhật
- α : learning rate (tốc độ học), là một hằng số dương, quyết định kích thước của bước di chuyển trong không gian trọng số.
- $\nabla_\theta J(\theta; x_i; y_i)$: gradient của hàm mất mát tại trọng số θ tức là vector chứa đạo hàm riêng của hàm mất mát theo từng tham số của mô hình.
- $(x_i; y_i)$: dữ liệu từ các batch.

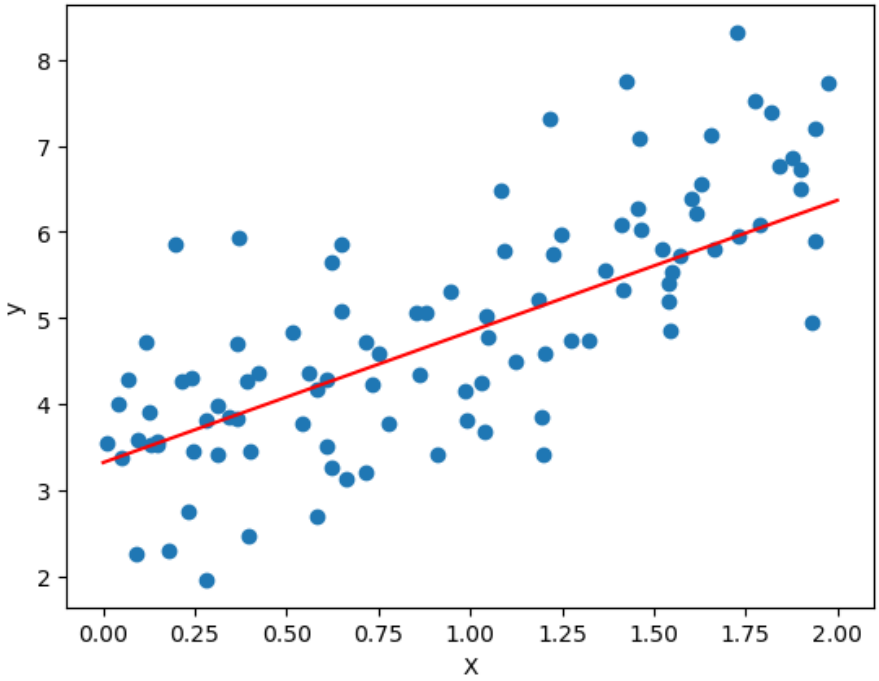
1.2.2.4 Ví dụ:

- Áp dụng SGD cho hàm tuyến tính y = 2x+3.ε



Hình 4: đồ thị hàm mất mát sau mỗi epoch

- Có thể thấy là đồ thị của SGD có đường đi khá là zig zắc, không mượt như GD. Vì việc chia nhỏ các dữ liệu ngẫu nhiên thay vì toàn bộ dữ liệu lớn.
- Khi mô hình được huấn luyện qua mỗi epoch, giá trị hàm mất mát giảm dần, cho thấy việc cải thiện chất lượng của mô hình theo thời gian. Đồ thị thường giảm đều dần nhưng có thể không ổn định do sự ngẫu nhiên trong việc chọn các điểm dữ liệu ngẫu nhiên để cập nhật tham số mô hình.



Hình 5: đường dự đoán của SGD

- Các điểm trên đồ thị scatter biểu diễn dữ liệu thực tế, trong khi đường thẳng đỏ biểu diễn đường dự đoán của mô hình. Đường thẳng này cố gắng 'fit' dữ liệu một cách tốt nhất có thể dựa trên các tham số đã học được sau quá trình huấn luyện.

1.2.3 Mini-batch Gradient Descent

1.2.3.1 Giới thiệu

- Mini-batch Gradient Descent là một biến thể của Gradient Descent (GD) được sử dụng rộng rãi trong Machine Learning, đặc biệt là trong Deep Learning.
- Khác với Stochastic Gradient Descent – sử dụng từng điểm dữ liệu một thì Mini-batch Gradient Descent sử dụng một lượng dữ liệu trung bình, được chia thành các mini-batch có kích thước n. Do đó, Mini-batch GD thường được ưa chuộng bởi sự linh hoạt của nó giữa sự chính xác (tính toán đạo hàm từ một lượng dữ liệu lớn hơn so với SGD).

1.2.3.2 Cách hoạt động của Mini-batch Gradient Descent

- Chia dữ liệu thành các mini-batch: Trước khi bắt đầu quá trình huấn luyện, dữ liệu được chia thành các mini-batch có kích thước cố định (ví dụ: 50, 100 điểm dữ liệu). Mini-batch cuối cùng có thể có số lượng ít hơn nếu kích thước của dữ liệu không chia hết cho kích thước mini-batch.
- Khởi tạo trọng số: trọng số của mô hình được khởi tạo ngẫu nhiên hoặc theo một phương pháp cụ thể.
- Xáo trộn dữ liệu và huấn luyện: Mỗi epoch (một vòng lặp qua toàn bộ dữ liệu), dữ liệu được xáo trộn ngẫu nhiên để đảm bảo tính ngẫu nhiên và không chồng lắp giữa các mini-batch.
- Cập nhật trọng số: thuật toán tính toán đạo hàm của hàm mất mát (loss function) dựa trên các điểm dữ liệu trong mini-batch.
- Lặp lại quá trình: Quá trình này được lặp lại qua số lượng epoch được xác định trước hoặc đến khi đạt điều kiện dừng cụ thể.

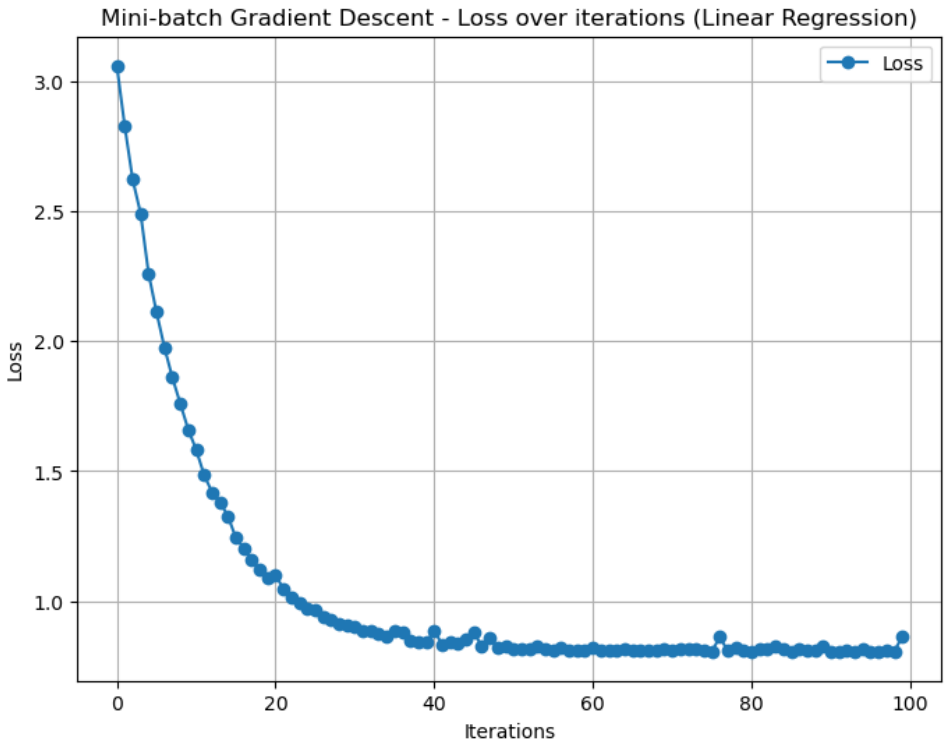
1.2.3.3 Công thức của Mini-batch Gradient Descent

$$\theta = \theta - \alpha . \nabla_\theta J(\theta;\ x_i : i+n\ ;\ y_i : i+n)$$

- θ : trọng số cần cập nhật.
- α : learning rate.
- $J(\theta;\ x_i : i+n\ ;\ y_i : i+n)$: hàm mất mát được tính trên mini-batch.
- $x_i : i+n\ ;\ y_i : i+n$: các điểm dữ liệu tương ứng với mini-batch.
- $i+n$: chỉ ra khoảng các điểm dữ liệu được chọn để tạo thành một mini-batch cụ thể trong quá trình huấn luyện.

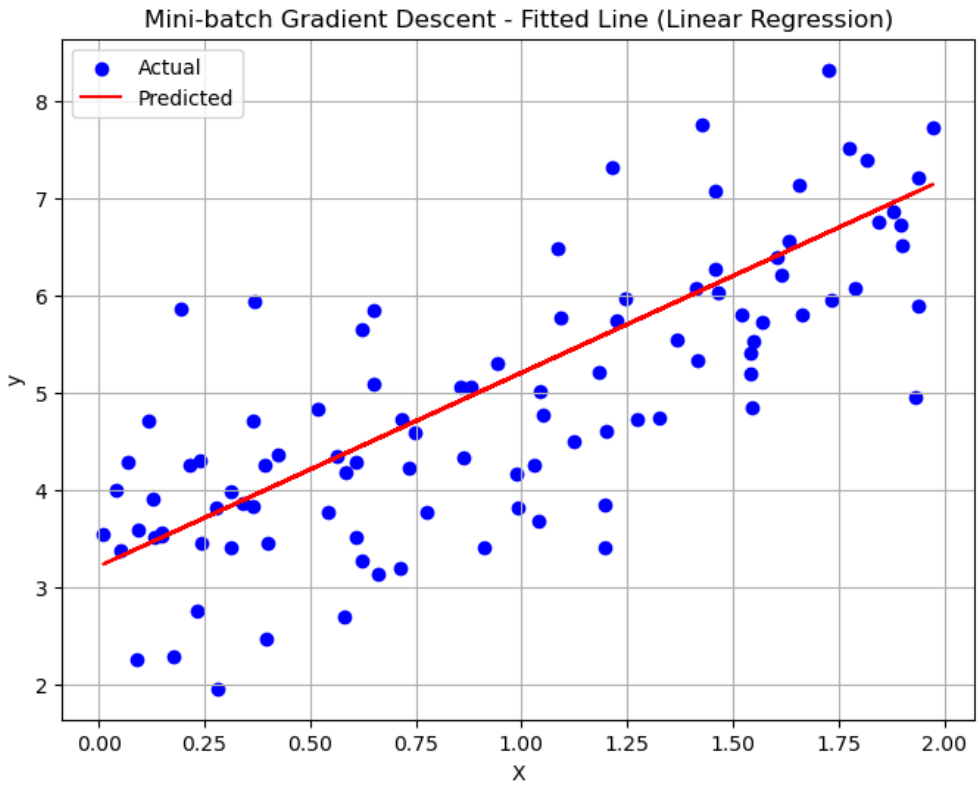
1.2.3.4 Ví dụ:

- Áp dụng Mini-batch Gradient Descent cho hàm tuyến tính y = 2x+3.ε



Hình 6: đồ thị hàm mất mát qua mỗi iteration

- Đồ thị biểu diễn sự thay đổi của loss function qua các iterations khi sử dụng Mini-batch Gradient Descent trong Linear Regression. Đồ thị này cho thấy sự giảm dần của loss function qua mỗi vòng lặp, và có thể thấy rõ sự dao động của loss function khi so sánh với số lần iterations. Sự dao động này thường phản ánh việc cập nhật weights và bias dựa trên từng mini-batch dữ liệu, không sử dụng toàn bộ tập dữ liệu mỗi lần cập nhật.



Hình 7: đường dự đoán của Mini-batch Gradient Descent

- Đồ thị biểu diễn dữ liệu thực tế (các điểm màu xanh) và đường tối ưu hóa được học từ Mini-batch Gradient Descent (đường màu đỏ). Đường đỏ biểu thị cho đường tối ưu hóa mô hình Linear Regression dựa trên dữ liệu đã được xử lý bằng Mini-batch Gradient Descent. Sự khác biệt giữa dữ liệu thực và dự đoán có thể được thấy qua sự phân tán của các điểm xanh xung quanh đường dự đoán màu đỏ. Điều này có thể là kết quả của việc áp dụng một mô hình học máy đơn giản để fitting dữ liệu thực tế, và cũng có thể bắt nguồn từ sự biến động của loss function trong quá trình huấn luyện.

1.2.4 Root Mean Square Propagation (RMSProp)

1.2.4.1 Giới thiệu

- Root Mean Square Propagation, là một trong những thuật toán tối ưu hóa phổ biến. Nó phát triển từ Rprop và giải quyết vấn đề của gradient thay đổi trong quá trình huấn luyện.
- Gradient thường có độ lớn khác nhau, từ nhỏ đến lớn, làm cho việc sử dụng một learning rate thống nhất trở nên không hiệu quả.
- Rprop ban đầu đã giải quyết vấn đề này bằng cách xem xét dấu của các gradient, điều chỉnh kích thước bước riêng biệt cho mỗi trọng số. Nếu các gradient có cùng dấu, thuật toán 14hem kích thước bước một cách từ từ, chỉ ra hướng đi đúng. Ngược lại, nếu chúng có dấu ngược nhau, nó giảm kích thước bước, đảm bảo sự cẩn trọng trong việc cập nhật.
- RMSprop mở rộng từ ý tưởng của Rprop để giữ lại những ưu điểm này và cải thiện thêm bằng cách duy trì một trung bình chạy của bình phương độ dốc trước đó. Thuật toán này giúp điều chỉnh tốc độ học tập linh hoạt hơn, giúp mô hình hội tụ tốt hơn trong quá trình huấn luyện.

1.2.4.2 Cách hoạt động của RMSProp

- Khởi tạo các tham số: trọng số của mô hình được khởi tạo ngẫu nhiên hoặc theo một phương pháp cụ thể và tạo một tốc độ học (learning rate).
- Tính toán gradient: Đây là đạo hàm của hàm mất mát theo từng trọng số.
- Duy trì trung bình bình phương của gradient (độ dốc) trước đó: RMSprop sử dụng một trung bình chạy của bình phương độ dốc để điều chỉnh tốc độ học. Đây là bước quan trọng, thuật toán tính trung bình có trọng số giữa bình phương độ dốc hiện tại và trung bình bình phương độ dốc trước đó.
- Cập nhật trọng số: RMSprop sử dụng trung bình bình phương độ dốc để điều chỉnh tốc độ học cho mỗi trọng số.
- Lặp lại quá trình: Quá trình trên sẽ được lặp lại qua nhiều epoch cho đến khi đạt được điều kiện dừng hoặc hội tụ tới mức tối ưu hóa mong muốn.

1.2.4.3 Công thức tính trung bình bình phương

$$E[g^2]_t = \beta . E[g^2]_{t-1} + (1-\beta) . {g_t}^2$$

- $E[g^2]_t$: giá trị trung bình bình phương của gradient tại thời điểm t.
- β : hệ số trọng số giữa giá trị trước đó $E[g^2]_{t-1}$ và bình phương gradient hiện tại ${g_t}^2$
- g_t : gradient tại thời điểm t, là đạo hàm của hàm mất mát theo các tham số của mô hình.
- Bằng cách làm này, sẽ tạo ra một trung bình di chuyển giữa gradient ở thời điểm trước và gradient hiện tại, giúp giảm thiểu các đợt dao động lớn trong việc cập nhật trọng số. Thông qua việc sử dụng các trung bình chạy, nó làm cho việc cập nhật trọng số ổn định hơn và giúp thuật toán hội tụ một cách trơn tru hơn.

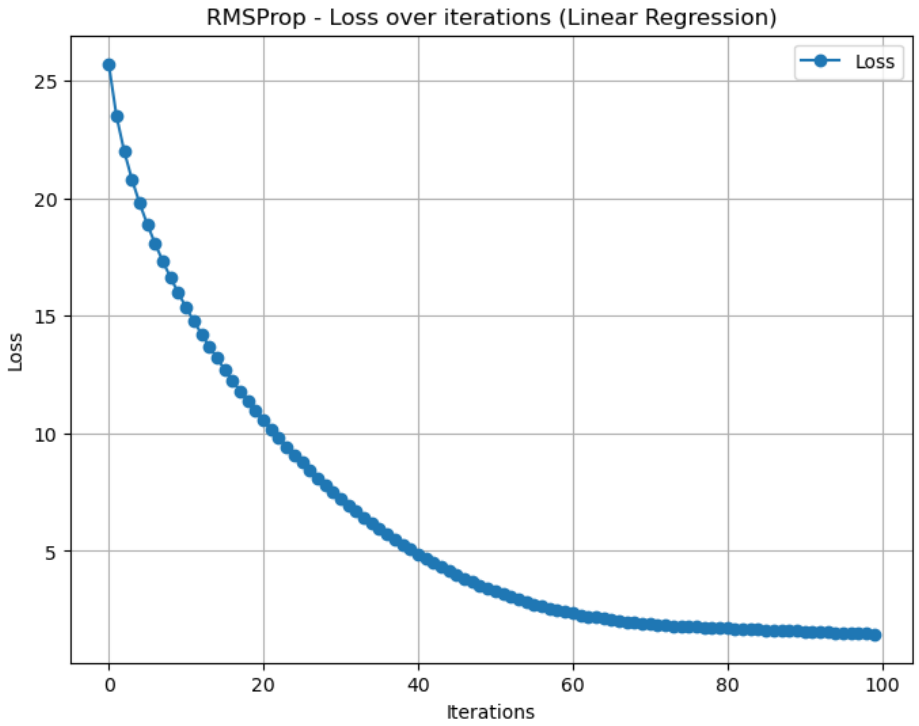
1.2.4.4 Công thức cập nhật trong RMSProp

$$\theta = \theta - \alpha . \frac{1}{\sqrt{E[g^2]_t + \varepsilon}} . g$$

- θ là các tham số mà chúng ta cần cập nhật
- α là tỉ lệ học (learning rate), quyết định tốc độ cập nhật các tham số.
- g là gradient của hàm mất mát theo các tham số θ tại thời điểm hiện tại.
- $E[g^2]_t$ là trung bình có trọng số của bình phương gradient (đạo hàm) theo thời gian.
- ε : là một giá trị nhỏ được thêm vào để tránh việc chia cho 0.

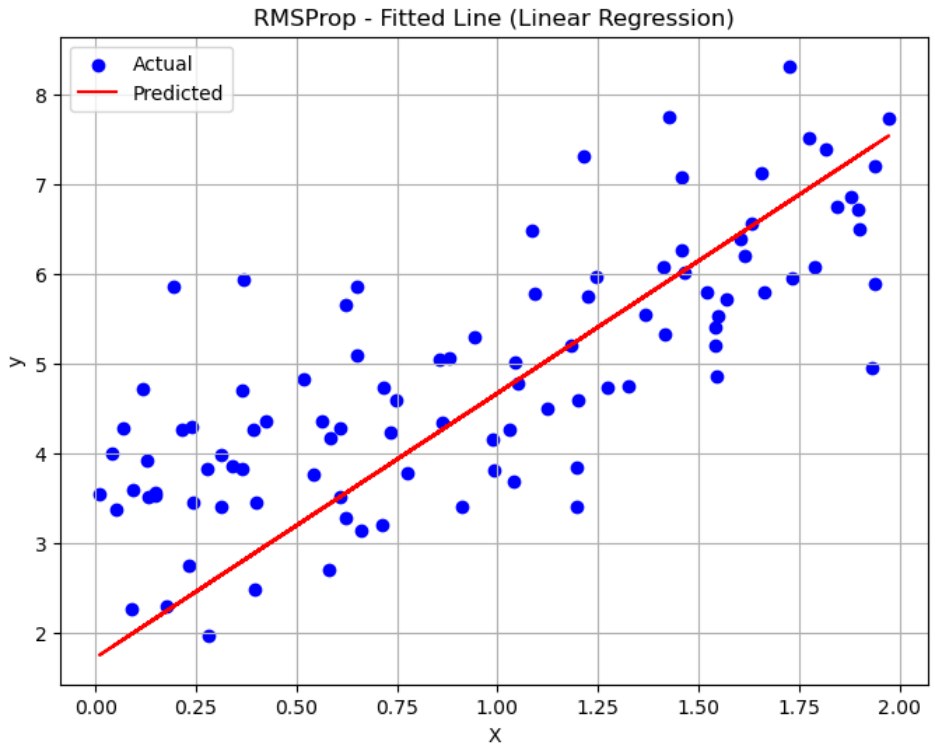
1.2.4.5 Ví dụ

- Áp dụng Root Mean Square Propagation cho hàm tuyến tính y = 2x+3.ε



Hình 8: đồ thị hàm mất mát qua mỗi iteration

- Ban đầu, giá trị của hàm mất mát giảm đáng kể khi mô hình được huấn luyện và cập nhật các tham số. Tuy nhiên, sau một số vòng lặp, giảm dần tốc độ hội tụ của loss function. Quá trình hội tụ dần trở nên chậm lại khi mô hình gần đến giá trị tối ưu, và thậm chí ổn định ở một giá trị cố định.



Hình 9: đường dự đoán của RMSProp

- Dữ liệu thực (màu xanh dương) biểu diễn các điểm dữ liệu thực tế trên trục X và giá trị tương ứng của chúng trên trục y. Đường màu đỏ, thể hiện đường dự đoán được học từ thuật toán RMSProp sau khi huấn luyện. Nó biểu diễn một mô hình tốt nhất có thể dự đoán các điểm dữ liệu mới từ các điểm dữ liệu huấn luyện.

1.2.5 Momentum

1.2.5.1 Giới thiệu

- Hay còn gọi là Gradient Descent với Momentum, là một cải tiến của thuật toán Gradient Descent và việc cập nhật trọng số không chỉ dựa trên thông tin từ độ dốc mà còn tính đến vận tốc trước đó để di chuyển.
- Đối với Gradient Descent, việc cập nhật trọng số được xác định dựa trên độ dốc của hàm mất mát. Tuy nhiên, nếu hàm mất mát có nhiều cực tiểu cục bộ (local), phương pháp này có thể bị kẹt ở các cực tiểu đó và không thể tiến về cực tiểu toàn cục (global).

- Momentum có thể giải quyết vấn đề này bằng cách giữ lại thông tin từ các bước trước đó để định hướng di chuyển. Ý tưởng chính là thêm vào công thức cập nhật trọng số một thành phần momentum, tượng trưng cho vận tốc của quá trình tối ưu.

1.2.5.2 Cách hoạt động của Momentum

- Tính toán gradient: Tại mỗi bước trong quá trình huấn luyện, thuật toán Gradient Descent tính độ dốc của hàm mất mát tại điểm hiện tại.
- Tích lũy gradient trước đó: Momentum lưu trữ thông tin về độ dốc trước đó và tích lũy thông tin này theo một hệ số gamma (γ).
- Tính toán tốc độ di chuyển: Sử dụng gradient hiện tại cùng với thông tin gradient tích lũy, thuật toán tính toán vận tốc di chuyển tiếp theo. Vận tốc di chuyển bao gồm độ dốc hiện tại và hướng của vận tốc trước đó.
- Cập nhật vị trí mới: Thuật toán cập nhật vị trí mới của điểm tối ưu (vị trí mới của nghiệm) bằng cách sử dụng vận tốc di chuyển tính được.

1.2.5.3 Công thức cập nhật

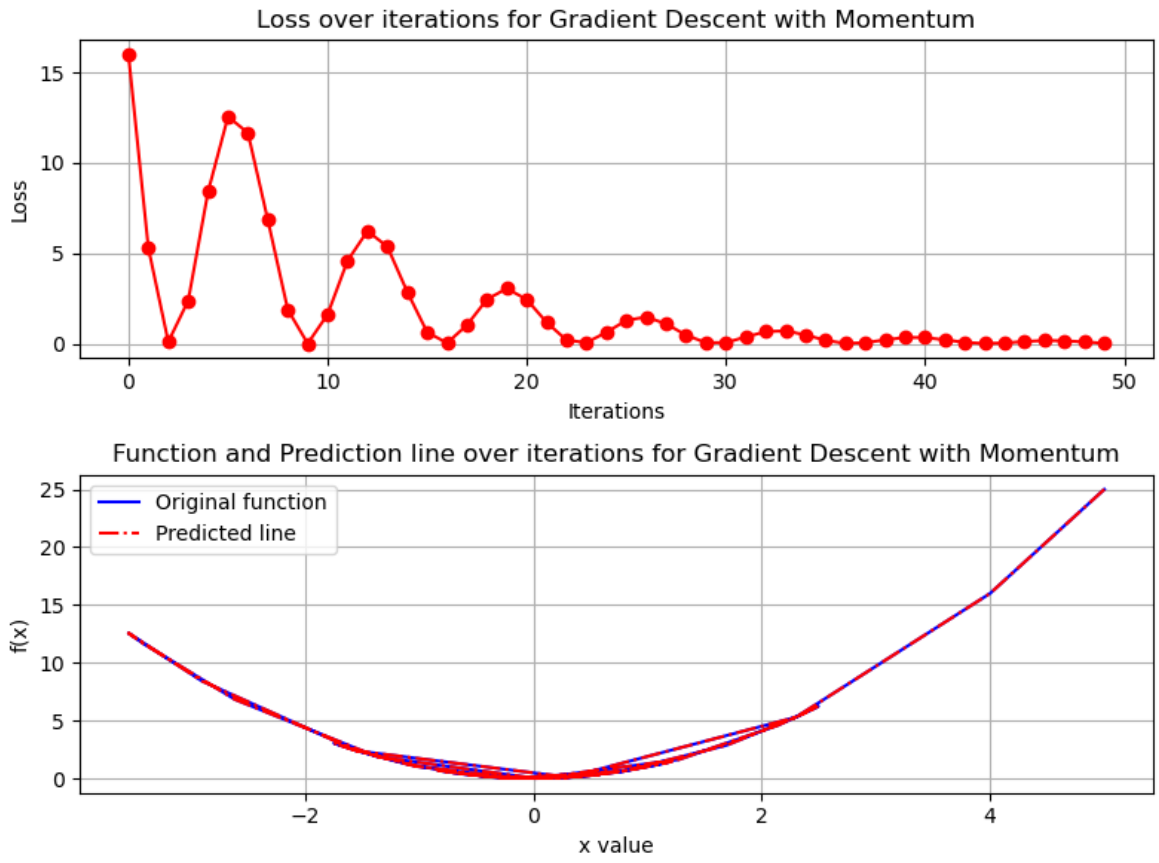
$$\theta = \theta - v_t$$

$$\text{Với } v_t = \gamma . v_{t-1} + \alpha . \nabla J(\theta)$$

- v_t : vận tốc tại bước thời gian t.
- γ : hệ số momentum, thường được chọn trong khoảng từ 0.8 đến 0.99.
- α : learning rate
- $\nabla J(\theta)$: gradient của hàm mất mát tại điểm hiện tại.
- θ : tham số cần cập nhật.

1.2.5.4 Ví dụ:

- Áp dụng Gradient Descent với Momentum cho f(x) = x^2



Hình 10: đồ thị hàm mất mát và đường dự đoán của Momentum

- Đồ thị đầu tiên là biểu đồ biểu diễn sự thay đổi của độ lỗi qua từng vòng lặp trong quá trình huấn luyện. Nó thể hiện giảm dần của độ lỗi khi thuật toán gradient descent được áp dụng. Ban đầu, độ lỗi giảm mạnh sau mỗi vòng lặp, nhưng sau một số lần lặp, việc giảm này trở nên chậm dần và cuối cùng độ lỗi hội tụ ở một giá trị gần với 0.
- Đồ thị thứ hai biểu diễn sự biến thiên của hàm số f(x) = x^2 (đường màu xanh dương) cùng với đường dự đoán (đường màu đỏ) khi áp dụng thuật toán Gradient Descent với Momentum. Kết quả này cho thấy đường dự đoán (màu đỏ) hội tụ về điểm cực tiểu của hàm số f(x) sau mỗi vòng lặp, và sau một số lần lặp, nó xấp xỉ với đường của hàm số f(x) (màu xanh dương) tại điểm cực tiểu này. Điều này chỉ ra rằng thuật toán Gradient Descent với Momentum hội tụ về điểm cực tiểu của hàm số f(x) và dự đoán đúng hình dạng của hàm số này.

1.2.6 Adaptive Moment Estimation (Adam)

1.2.6.1 Giới thiệu

- Adam là một thuật toán tối ưu hóa lặp để giảm thiểu hàm mất mát trong quá trình huấn luyện mạng nơ-ron. Adam có thể được coi như sự kết hợp giữa RMSprop và Momentum.
- Adam sử dụng gradient bình phương để điều chỉnh tốc độ học tập giống như RMSprop, và nó tận dụng Momentum bằng cách sử dụng trung bình di chuyển của gradient như Momentum.

1.2.6.2 Cách hoạt động của Adam

- Tương tự như RMSprop, Adam sử dụng squared gradients để điều chỉnh tỷ lệ học tập cho từng tham số của mô hình. Điều này giúp làm giảm tỷ lệ học tập đối với các tham số có gradient lớn, và tăng tỷ lệ học tập đối với các tham số có gradient nhỏ.
- Adam sử dụng Momentum bằng cách duy trì trung bình di chuyển của gradient thay vì sử dụng gradient chính xác. Điều này giúp cải thiện việc di chuyển của các trọng số, giúp mô hình tránh được việc mắc kẹt ở các điểm cực tiểu địa phương.
- Adam cũng điều chỉnh learning rate cho từng tham số riêng biệt dựa trên squared gradients và Momentum của chúng. Quá trình này giúp tối ưu hóa tốc độ học tập cho từng tham số cụ thể, tạo điều kiện thuận lợi cho việc hội tụ nhanh chóng và hiệu quả hơn.

1.2.6.3 Công thức cập nhật

- Công thức tính gradient:

$$g_t = \frac{\partial L}{\partial w_t}$$

- Công thức cập nhật trung bình gradient:

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

- m_t : trung bình động của gradient tại thời điểm t.
- β_1 : hệ số momentum cho gradient (thường được đặt là khoảng 0.9).

- Công thức cập nhật trung bình bình phương gradient

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

- v_t : trung bình bình phương gradient tại thời điểm t.
- β_2 : hệ số momentum cho squared gradient (thường được đặt là khoảng 0.999)

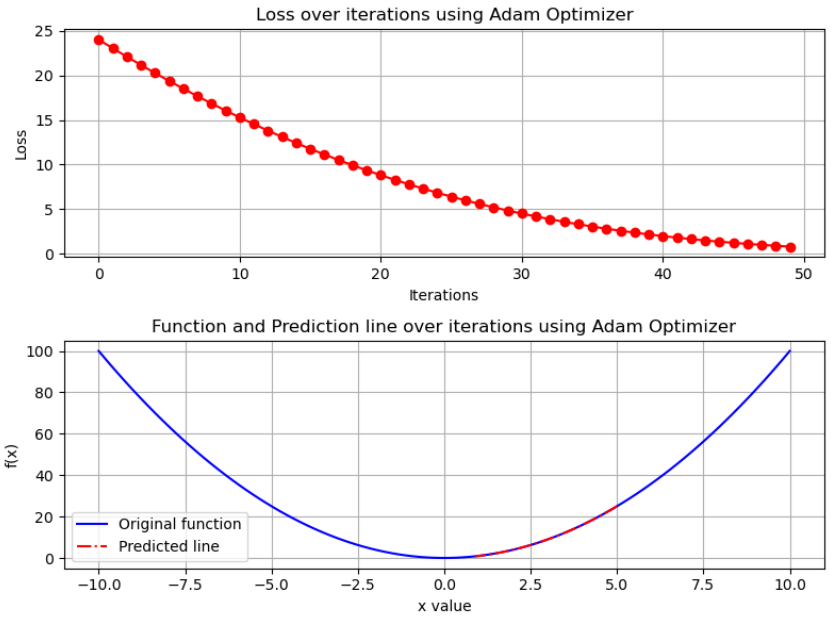
- Công thức cập nhật trọng số:

$$\theta = \theta - \alpha . \frac{1}{\sqrt{v_t} + \varepsilon} . m_t$$

- θ : trọng số cần cập nhật
- α : learning rate
- ε : một số rất nhỏ để tranh trường hợp chia cho 0.

1.2.6.4 Ví dụ

- Áp dụng Adam cho f(x) = x^2



Hình 11: đồ thị hàm mất mát và đường dự đoán của Adam

- Đồ thị đầu tiên biểu diễn sự thay đổi của hàm mất mát theo số lần lặp khi sử dụng thuật toán tối ưu Adam. Thuật toán được thực hiện để tối thiểu hóa hàm số f(x) = x^2 với x là biến số. Đồ thị này cho thấy giá trị mất mát giảm dần theo thời gian, cho thấy thuật toán Adam đang hội tụ đến điểm cực tiểu của hàm số.
- Đồ thị thứ hai hiển thị đường cong của hàm số f(x) = x^2 (màu xanh) cùng với đường dự đoán (màu đỏ) được tối ưu hóa bằng thuật toán Adam. Đường màu đỏ biểu thị sự di chuyển của dự đoán qua các vòng lặp theo thuật toán Adam. Như có thể thấy, đường dự đoán tiến dần đến điểm cực tiểu của hàm số f(x), một biểu hiện rõ ràng về việc thuật toán Adam hiệu quả trong tối ưu hóa hàm mất mát.

1.3 So sánh các phương pháp

	GD	SGD	Mini-batch GD
Dạng bài toán áp dụng	Tất cả các loại bài toán tối ưu hóa mà có thể tính gradient của hàm mất mát.	Phù hợp với bài toán khi dữ liệu lớn và có thể áp dụng gradient tính toán ngẫu nhiên.	Sử dụng trong bài toán lớn, có thể chia thành các mini-batch.
Ưu điểm	- Dễ hiểu và triển khai. - Có thể hội tụ đến điểm cực tiểu toàn cục nếu hàm mất mát là lồi.	- Hiệu quả khi làm việc với dữ liệu lớn. - Dễ dàng cập nhật trọng số với mỗi mẫu dữ liệu.	- Hiệu quả hơn SGD và GD khi áp dụng trên dữ liệu lớn. - Linh hoạt với việc chọn kích thước mini-batch.
Nhược điểm	-Chậm khi áp dụng trên dữ liệu lớn. - Dễ mắc kẹt ở cực tiểu cục bộ nếu hàm mất mát không lồi.	- Không ổn định và dao động quanh cực tiểu. - Cần đặt kích thước bước học (learning rate) thấp để tránh dao động lớn.	- Đòi hỏi điều chỉnh thêm hyperparameters như kích thước batch.
Tốc độ thực thi	Thực thi chậm, đặc biệt với dữ liệu lớn vì cần phải tính toán trên toàn bộ dataset.	Nhanh hơn GD do cập nhật trọng số sau mỗi mẫu dữ liệu, nhưng có thể dao động.	Hiệu quả hơn SGD và GD với dữ liệu lớn do cập nhật trọng số sau mỗi batch, giúp giảm thời gian tính toán.
Mức đánh giá	3/5	3.5/5	4/5

	RMSProp	Momentum	Adam
Dạng bài toán áp dụng	Phù hợp với các bài toán mà hàm mất mát không thể tìm được một cực tiểu toàn cục thông qua một cực tiểu cục bất kỳ.	Phù hợp với hầu hết các loại bài toán tối ưu hóa.	Phù hợp với nhiều loại bài toán tối ưu hóa, đặc biệt trong Deep Learning.
Ưu điểm	- Giúp tránh được mắc kẹt ở cực tiểu cục bộ. - Cải thiện tốc độ hội tụ.	- Giúp tăng tốc độ hội tụ, đặc biệt trong các bài toán với địa hình phức tạp. - Giảm độ dao động.	- Hiệu quả trong việc cập nhật trọng số. - Tự điều chỉnh learning rate tương ứng với từng tham số.
Nhược điểm	- Đòi hỏi sự điều chỉnh kỹ lưỡng của hyperparameters.	-Momentum có thể tăng lên quá nhanh, dẫn đến hiện tượng vị trí hiện tại của cực tiểu dự đoán bị đi quá xa so với vị trí cực tiểu thực sự.	- Cần điều chỉnh các hyperparameters.
Tốc độ thực thi	Nhanh hơn GD với khả năng tránh được mắc kẹt ở cực tiểu cục bộ, cải thiện tốc độ hội tụ.	Tăng tốc độ hội tụ nhưng không có tác động lớn đến thời gian thực thi.	Hiệu quả và linh hoạt trong việc tự điều chỉnh learningrate, thường đạt hiệu suất cao nhanh chóng.
Mức đánh giá	4/5	3.5/5	4.5/5

2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual Learning

2.1.1 Giới thiệu

- Một vấn đề phổ biến khi học tuần tự nhiều nhiệm vụ chính là hiện tượng quên tức là quá trình học kiến thức mới nhanh chóng làm xáo trộn thông tin đã được học được trước đó. Đa phần các mô hình học máy đều gặp vấn đề này.
- Continual Learning (hay còn gọi là Lifelong Learning) là một phương pháp trong Machine Learning, tập trung vào việc mô hình học máy không chỉ học từ dữ liệu ban đầu mà còn có khả năng học từ dữ liệu mới liên tục và không ngừng nghỉ.
- Đây là một khía cạnh quan trọng trong xây dựng các giải pháp học máy có khả năng tự cập nhật và điều chỉnh khi có dữ liệu mới, giúp mô hình không chỉ học từ dữ liệu ban đầu mà còn có khả năng học từ dữ liệu mới mà nó gặp phải sau này.
- Continual Learning là một lĩnh vực nghiên cứu quan trọng vì nó giải quyết tình huống thực tế trong đó dữ liệu và nhiệm vụ liên tục thay đổi, và một mô hình phải thích nghi với những thay đổi này mà không quên đi kiến thức trước đó.

2.1.2 Mục tiêu của Continual Learning

- Mô hình không thể có không gian lưu trữ vô hạn cho toàn bộ kinh nghiệm từ các nhiệm vụ trước. Do đó cần phải quản lý thông tin quan trọng từ quá khứ mà không làm quá tải hoặc làm suy giảm hiệu suất của mô hình.
- Mô hình cần có khả năng hạn chế truy cập vào thông tin từ các nhiệm vụ đã hoàn thành trước đó. Việc này giúp mô hình tập trung chủ yếu vào nhiệm vụ hiện tại mà không bị ảnh hưởng bởi thông tin không cần thiết.

- Trong quá trình học liên tục, mức độ tăng cường về khả năng và yêu cầu tính toán của mô hình cần được kiểm soát để tránh tăng quá mức ảnh hưởng đến hiệu suất hoặc khả năng thích ứng của mô hình.
- Mô hình Continual Learning cần có khả năng mở rộng mà không cần phải tạo ra một mô hình mới cho mỗi nhiệm vụ. Sự linh hoạt này giúp mô hình thích ứng với nhiệm vụ mới mà không tăng đáng kể độ phức tạp của nó.
- Mô hình cần có khả năng thích ứng nhanh chóng với các nhiệm vụ mới hoặc khi có sự thay đổi miền dữ liệu. Đồng thời, sau khi thích ứng, mô hình cũng cần có khả năng phục hồi nhanh chóng, duy trì hiệu suất cao khi đối mặt với các thay đổi này.

2.1.3 Những thách thức của Continual Learning

- Catastrophic forgetting (sự quên): Đây là vấn đề khi một mô hình học từ nhiều nhiệm vụ và sau đó quên thông tin từ các nhiệm vụ trước khi học nhiệm vụ mới. Điều này có thể làm giảm hiệu suất trên các nhiệm vụ đã học trước đó.
- Đặc điểm của dữ liệu và môi trường: Sự thay đổi liên tục của dữ liệu và môi trường làm phức tạp việc học liên tục, đặc biệt khi có sự thay đổi đột ngột hoặc không dự đoán được.
- Mâu thuẫn giữa tính ổn định và tính linh hoạt: Sự cân bằng giữa việc thích ứng nhanh chóng với dữ liệu mới và duy trì một mức độ ổn định trong kiến thức đã học là một thách thức, gọi là mâu thuẫn ổn định-linh hoạt.
- Khả năng thích ứng: Mô hình Continual Learning cần thích ứng nhanh chóng với dữ liệu mới mà không quên đi kiến thức quan trọng từ các nhiệm vụ trước đó. Điều này đặt ra thách thức trong việc duy trì cân bằng giữa việc học từ dữ liệu mới và duy trì kiến thức đã học.
- Khả năng tính toán: Mô hình cần giữ cân bằng giữa việc không tăng quá mức sức chứa và yêu cầu tính toán, trong khi vẫn cần thích ứng với dữ liệu mới và duy trì kiến thức đã học.

2.1.4 Cách hoạt động của Continual Learning

- Nhận nhiệm vụ tuần tự: Hệ thống nhận dữ liệu từ nhiều nguồn khác nhau đại diện cho các nhiệm vụ khác nhau một cách tuần tự.
- Học kiến thức mới: Mỗi khi học một nhiệm vụ mới, hệ thống cập nhật kiến thức của mình dựa trên dữ liệu mới này. Điều này có thể liên quan đến việc học các tính năng mới hoặc điều chỉnh các tham số trong mô hình.
- Lưu trữ và tái sử kiến thức cũ: Thay vì quên hoàn toàn thông tin từ các nhiệm vụ trước, hệ thống Continual Learning cố gắng giữ lại kiến thức cần thiết từ các nhiệm vụ trước đó. Điều này có thể thông qua việc sử dụng các kỹ thuật như bổ sung bộ nhớ hoặc tái sử dụng dữ liệu từ các nhiệm vụ trước.
- Thích ứng và cải thiện: Mỗi khi hệ thống đối mặt với nhiệm vụ mới, nó cố gắng thích ứng và cải thiện hiệu suất của mình dựa trên kinh nghiệm học được từ các nhiệm vụ trước.
- Đánh giá và điều chỉnh: Hệ thống liên tục được đánh giá để đảm bảo rằng kiến thức từ các nhiệm vụ trước đang được bảo tồn và hiệu suất trên các nhiệm vụ mới không bị suy giảm quá nhiều. Nếu cần, nó có thể điều chỉnh mô hình hoặc cơ chế học tập của mình để cải thiện khả năng học tiếp theo.
- Lặp lại quá trình: Quá trình này lặp đi lặp lại khi hệ thống tiếp tục nhận dữ liệu từ các nhiệm vụ mới và cố gắng học và thích ứng với chúng mà không quên kiến thức từ quá khứ.
- Theo dõi và nâng cao: Liên tục theo dõi hiệu suất của mô hình và đảm bảo rằng nó vẫn hiệu quả khi phải xử lý dữ liệu mới và cũ. Áp dụng các kỹ thuật và phương pháp mới để cải thiện khả năng học liên tục và khả năng chuyển giao kiến thức.

2.1.5 Cách Continual Learning vượt qua Catastrophic forgetting

Hệ thống Continual Learning vượt qua việc quên kiến thức đã học thông qua một số kỹ thuật và phương pháp:

- Regularization (Chuẩn hóa): thêm các ràng buộc hoặc hạn chế vào quá trình học để ngăn mô hình học thuộc lòng dữ liệu và duy trì khả năng tổng quát hóa. Trong Continual Learning, các phương pháp như Dropout hoặc Weight Decay được sử dụng. Ví dụ, Dropout loại bỏ ngẫu nhiên một số đơn vị nơ-ron trong quá trình huấn luyện, ngăn chặn mô hình phụ thuộc quá mức vào một số tính năng cụ thể.
- Rehearsal (tập luyện): liên quan đến việc lưu trữ và sử dụng lại một số mẫu từ các nhiệm vụ trước khi học nhiệm vụ mới. Các mẫu này được đưa vào quá trình huấn luyện của nhiệm vụ mới, giúp mô hình duy trì và tái sử dụng kiến thức từ quá khứ. Điều này có thể là một tập hợp các dữ liệu mẫu hoặc một số thông tin đặc biệt mà mô hình cần nhớ.
- Transfer learning (học chuyển giao): cho phép sử dụng kiến thức từ các nhiệm vụ trước đó để hỗ trợ học hiệu quả trên nhiệm vụ mới. Thay vì bắt đầu từ đầu, mô hình sử dụng những gì đã học được từ các nhiệm vụ trước để nhanh chóng học và cải thiện hiệu suất trên nhiệm vụ mới. Điều này giúp mô hình chỉ cần ít dữ liệu hơn để làm chủ nhiệm vụ mới.
- Backward transfer (chuyển giao ngược) : là khả năng cho phép kiến thức mới từ các nhiệm vụ sau có thể áp dụng ngược trở lại vào các nhiệm vụ trước. Điều này có thể cải thiện hiệu suất của mô hình trên các nhiệm vụ cũ mà không cần thêm dữ liệu mới từ chúng. Backward transfer cho phép các nhiệm vụ sau cung cấp thông tin hữu ích để cải thiện hiệu suất trên các nhiệm vụ đã học trước đó.

2.1.6 So sánh với những phương pháp học máy truyền thống

	Machine Learning Basic	Continual Learning
Dữ liệu và mô hình học tập	mô hình thường được huấn luyện trên một tập dữ liệu cố định và sau đó được sử dụng để thực hiện một nhiệm vụ cụ thể.	yêu cầu mô hình không chỉ học từ dữ liệu ban đầu mà còn có khả năng học từ dữ liệu mới khi nó đến. Mô hình cần thích nghi với dữ liệu liên tục và không ngừng thay đổi mà không quên đi kiến thức đã học.
Khả năng thích ứng	mô hình thường không có khả năng thích ứng tự nhiên với dữ liệu mới mà cần phải được huấn luyện lại từ đầu khi có dữ liệu mới.	Mục tiêu chính của Continual Learning là giúp mô hình thích ứng và cải thiện từ dữ liệu mới mà không quên đi kiến thức đã học từ dữ liệu cũ. Khả năng học liên tục và duy trì kiến thức trước đó là trọng tâm.
Thách thức của dữ liệu động	thường đối mặt với khó khăn khi dữ liệu và nhiệm vụ thay đổi, không thể sử dụng lại mô hình hiện tại một cách linh hoạt.	Có thể giải quyết vấn đề của dữ liệu động bằng cách xây dựng các mô hình có khả năng học từ dữ liệu liên tục và không ngừng cập nhật kiến thức để thích ứng với môi trường thay đổi.
Quản lý kiến thức đã học	Mô hình thường không có cơ chế tự nhiên để duy trì kiến thức từ các nhiệm vụ hoặc dữ liệu trước.	Mục tiêu là duy trì và sử dụng lại kiến thức đã học từ các nhiệm vụ trước đó, không gây ra việc quên tàn khốc và giúp mô hình nâng cao khả năng tổng quát hóa.

2.1.7 Xây dựng mô hình Continual Learning

Mô hình Continual Learning được xây dựng để linh hoạt và có khả năng thích ứng với dữ liệu mới thông qua một số cách tiếp cận:

- Kiến trúc module: Mô hình được thiết kế thành các phần nhỏ độc lập có khả năng học và cập nhật riêng biệt. Điều này giúp tăng tính linh hoạt và cho phép mô hình học từng phần một mà không ảnh hưởng đến toàn bộ cấu trúc.
- Memory-Augmented Networks: Sử dụng cơ chế bổ sung bộ nhớ để lưu trữ thông tin quan trọng từ các nhiệm vụ trước. Mô hình có khả năng truy xuất thông tin này khi học các nhiệm vụ mới.
- Tính linh hoạt của kiến trúc: Các kiến trúc mô hình được thiết kế để dễ dàng thích ứng với dữ liệu mới và thậm chí có thể điều chỉnh cấu trúc của chính nó để phù hợp với các nhiệm vụ mới.

2.1.8 Hướng phát triển trong tương lai

- Cải thiện hiệu suất: Nâng cao khả năng học của mô hình và giảm thiểu việc quên thông tin là mục tiêu quan trọng.
- Đa nhiệm: Tăng cường khả năng của mô hình để thực hiện nhiều nhiệm vụ cùng một lúc mà không ảnh hưởng đến hiệu suất.
- Học không giám sát: Mở rộng Continual Learning sang việc học từ dữ liệu không được gắn nhãn, giúp mô hình tự học hỏi từ nguồn dữ liệu rộng lớn và đa dạng.

2.2 Test Production

2.2.1 Giới thiệu

- Kiểm thực học máy là kiểm tra các logic đã học được trong quá trình huấn luyện.
- Test Production trong Machine Learning là quá trình đảm bảo rằng mô hình học máy hoạt động đúng đắn và hiệu quả khi được triển khai vào môi trường thực tế.
- Quá trình này bao gồm chuẩn bị, kiểm tra, và đánh giá mô hình trong môi trường thực tế để đảm bảo tính ổn định, hiệu suất và đáp ứng được yêu cầu của ứng dụng.

2.2.2 Vai trò của test production

- Đảm bảo chất lượng mô hình:
    - Đánh giá xem mô hình hoạt động như mong đợi hay không, đảm bảo rằng nó hoạt động đúng đắn trên dữ liệu thực tế.
    - Đo lường hiệu suất của mô hình trong môi trường sản xuất, sử dụng các thước đo và tiêu chí đã xác định trước.
- Đảm bảo tính ổn dịnh và độ tin cậy:
    - Xác định tính ổn định và đáng tin cậy của mô hình trong môi trường thực tế, đảm bảo rằng mô hình không gây ra lỗi hoặc sự cố không mong muốn.
    - Thiết lập hệ thống giám sát để theo dõi hoạt động của mô hình sau khi triển khai.
- Tối ưu hóa hiệu quả triển khai:
    - Xác định yêu cầu và chuẩn bị môi trường triển khai mô hình để đảm bảo việc triển khai mô hình được thực hiện một cách hiệu quả.
    - Áp dụng các biện pháp bảo trì và cập nhật định kỳ để duy trì hiệu suất của mô hình trong môi trường sản xuất.
- Đối phó với thay đổi và phát triển:

- Đảm bảo rằng mô hình có thể đáp ứng với sự thay đổi trong dữ liệu và môi trường mà không làm giảm hiệu suất.
- Dựa trên kết quả đánh giá và giám sát, điều chỉnh và cải tiến mô hình để nâng cao hiệu suất và sẵn sàng cho thách thức mới.

2.2.3 Quy trình Test Production

- Chuẩn bị dữ liệu:
  - Đánh giá tính chính xác và đại diện của dữ liệu thực tế sẽ được sử dụng trong môi trường sản xuất. Điều này bao gồm kiểm tra độ toàn vẹn, sự chuẩn xác, và tính thống nhất của dữ liệu.
  - Chuẩn bị dữ liệu cho việc triển khai, bao gồm xử lý, tiền xử lý và chuẩn hóa để đảm bảo phù hợp với mô hình.
- Chuẩn bị môi trường triển khai:
  - Xác định môi trường mà mô hình sẽ được triển khai, bao gồm hạ tầng, quy mô, yêu cầu về tài nguyên, và khả năng tích hợp với hệ thống hiện tại.
  - Dựa trên phân tích trước đó, chuẩn bị môi trường triển khai bằng cách cấu hình hệ thống, chuẩn bị tài nguyên và xác định các yếu tố khác cần thiết để triển khai mô hình.
- Triển khai mô hình:
  - Xây dựng môi trường triển khai cho mô hình, bao gồm cấu hình hệ thống, cài đặt phần mềm và tài nguyên cần thiết.
  - Đưa mô hình đã huấn luyện vào môi trường sản xuất.
- Kiểm tra và đánh giá:
  - Tiến hành kiểm tra kỹ thuật để đảm bảo rằng mô hình hoạt động chính xác trong môi trường thực tế và không gây ra lỗi.
  - Đánh giá hiệu suất của mô hình trong môi trường sản xuất, sử dụng các tiêu chí và thước đo đã được xác định trước (ví dụ: độ chính xác, recall, precision).
- Theo dõi và bảo trì:

- Thiết lập hệ thống giám sát để liên tục theo dõi hoạt động của mô hình trong môi trường sản xuất.
- Áp dụng các biện pháp bảo trì định kỳ, kiểm tra và cập nhật mô hình nếu cần thiết để duy trì hiệu suất và đáp ứng yêu cầu thực tế.

TÀI LIỆU THAM KHẢO

Tiếng Anh

[1] Parisi et al., "Continual Lifelong Learning with Neural Networks: a review," Proceedings of the IEEE Conference on X, Volume, Page 123-135.

[2] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," arXiv preprint arXiv:1609.04747, (2016).

[3] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations (ICLR), (2015).

[4] N. Ashia, "RMSprop: Divide the Gradient by a Running Average of its Recent Magnitude," Courant Institute of Mathematical Sciences, New York University, (2012).

[5] M. Käpylä, P. Kärkkäinen, "Testing and Monitoring Machine Learning Models in Real-time Production Systems," Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA), page 123-135.

[6] L. Xu, "Understanding and Improving Optimizers for Machine Learning," Proceedings of the AAAI Conference on Artificial Intelligence, Volume (Năm), Trang 123-135.