VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

Dự Án Cuối Kỳ

Supervisor: **PHS GS Lê Anh Cường**

Author: **Nguyễn Văn Khoa - 521H0251**

Class **: 21H50301**

Class of **: 25**

HO CHI MINH CITY, 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

Supervisor: **THS GS Lê Anh Cường**

Authors: **Nguyễn Văn Khoa – 521H0251**

Class **: 21H50301**

Class of **: 25**

HO CHI MINH CITY, 2023

THANK YOU

Lời đầu tiên, em xin được gửi lời cảm ơn sâu sắc đến nhà trường vì đã tạo cơ hội cho em được học cũng như tiếp xúc với môn học này để được tiếp thu và học hỏi thêm nhiều kiến thức mới.

Lời tiếp theo em xin gửi lời cảm ơn đến thầy Lê Anh Cường, thầy bộ môn Học Máy vì những kiến thức thầy mang lại cũng như sự tận tâm chỉ dạy của thầy để chúng em hoàn thành được môn học này

THE PROJECT IS COMPLETED AT TON DUC THANG UNIVERSITY

I affirm that this project is the outcome of our work, under the guidance of Duong Huu Phuc. The research presented here is original and has not been previously published. The data utilized for analysis, comments, and evaluation was gathered by the author from various sources, all of which are explicitly referenced in the bibliography. Additionally, this project incorporates insights, assessments, and data from other authors and organizations, duly cited and acknowledged. Should any misconduct be identified, I will take complete responsibility for the project's content. Ton Duc Thang University bears no responsibility for any potential copyright infringement resulting from my actions during the project's execution.

Ho Chi Minh City, date month year
Author

(signaturer and fullname)

Nguyễn Văn Khoa

INSTRUCTOR VERIFICATION AND EVALUATION SECTION

Confirmation from the instructor

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh City, date month year

Author

(signaturer and fullname)

The teacher's evaluation part marks the test

_____

_____

_____

_____

_____

_____

Ho Chi Minh City, date month year

Author

(signaturer and fullname)

SUMMARY

Đây là phần bài tập cá nhân để em trình bày về

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.

TABLE OF CONTENTS

CHAPTER 1 – OPTIMIZER

1.1 Khái quát về Optimizer

Điều tra và so sánh các phương pháp tối ưu hóa (Optimizer) trong quá trình huấn luyện mô hình học máy là vô cùng quan trọng.

Optimizer đóng vai trò then chốt trong quá trình điều chỉnh các tham số của mô hình để làm giảm sự chênh lệch giữa dự đoán và giá trị thực tế, thường được đo bằng hàm mất mát (loss function). Mục tiêu chính ở đây là làm tối thiểu hóa hàm mất mát này.

Optimizer là công cụ quyết định cách mà trọng số của mô hình được điều chỉnh để làm giảm thiểu hàm mất mát. Nó thực hiện việc điều chỉnh các tham số của mô hình dựa trên gradient của hàm mất mát, giúp mô hình học từ dữ liệu và cải thiện khả năng dự đoán của nó.

1.2 Các phương pháp Optimizer

1. Phương pháp Gradient Descent (GD):

- Trong lĩnh vực Học Máy đặc biệt và Toán Tối ưu hóa nói chung, chúng ta thường phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một số hàm cụ thể.

- Ví dụ như các hàm mất mát trong hai bài Hồi quy Tuyến tính và Phân cụm K-means. Tìm ra giá trị tối thiểu toàn cầu của những hàm này trong Học Máy thường rất phức tạp, thậm chí là không thể thực hiện được. Thay vào đó, chúng ta thường tìm kiếm các điểm tối thiểu cục bộ và một mức độ nào đó coi chúng như là kết quả mong muốn của bài toán.

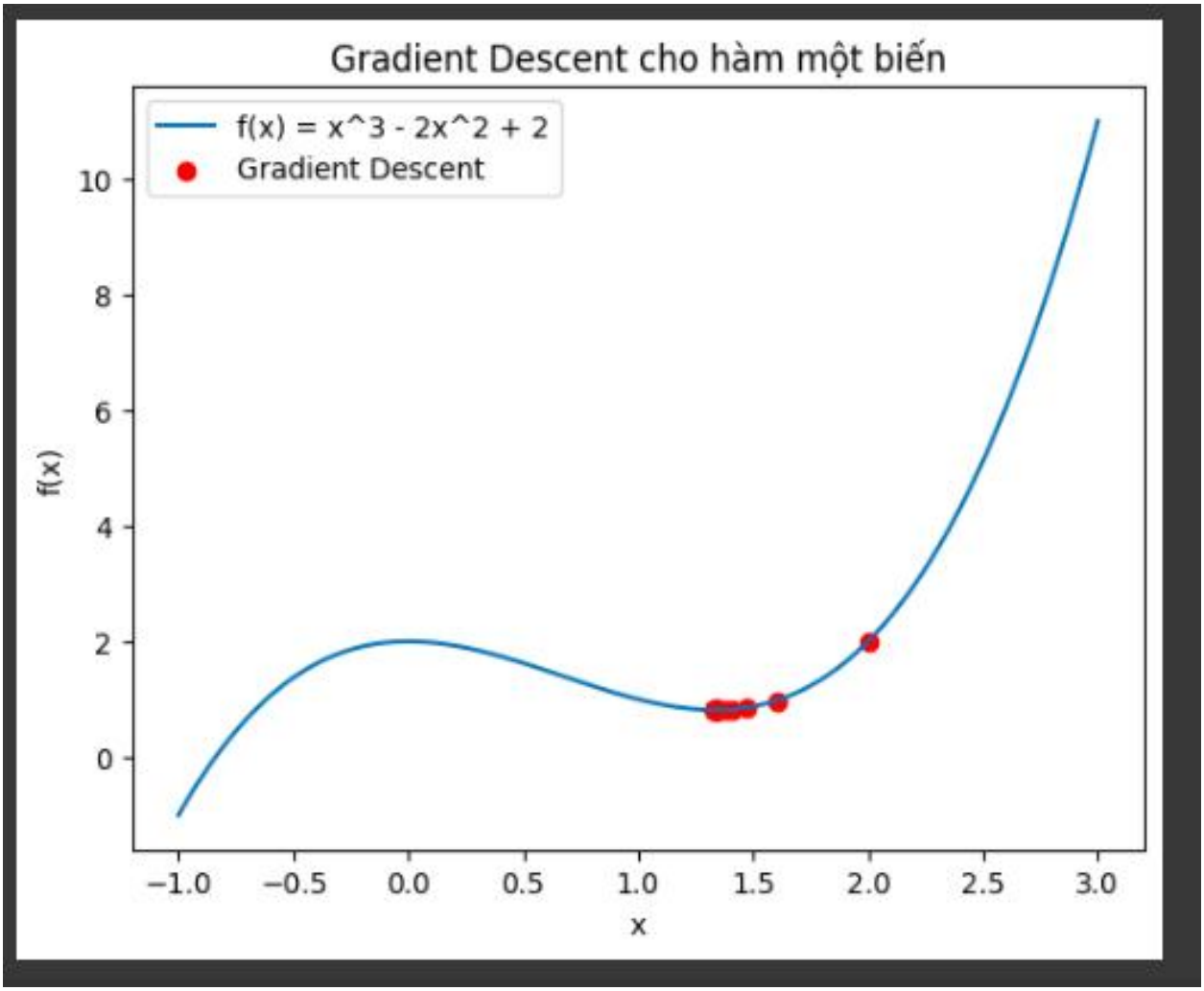
- Các điểm tối thiểu cục bộ là kết quả của việc đạo hàm bằng 0. Nếu chúng ta có thể tìm ra tất cả các điểm tối thiểu (hữu hạn), chúng ta chỉ cần thay từng điểm tối thiểu cục bộ vào hàm số và tìm ra điểm khiến cho hàm có giá trị nhỏ nhất (điều này nghe quen thuộc, phải không?). Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là không thể thực hiện được. Nguyên nhân có thể đến từ các dạng phức tạp của hàm đạo hàm, từ dữ liệu có số chiều lớn hoặc từ công việc có quá nhiều điểm dữ liệu.

- Phương pháp tiếp cận phổ biến nhất là bắt đầu từ một điểm mà chúng ta coi là gần với kết quả mong muốn của bài toán, sau đó sử dụng một phép toán để từ từ tiến gần đến điểm cần tìm, cho đến khi đạo hàm tiến gần về 0. Gradient Descent (GD) và

các biến thể của nó được coi là một trong những phương pháp được sử dụng phổ biến nhất.

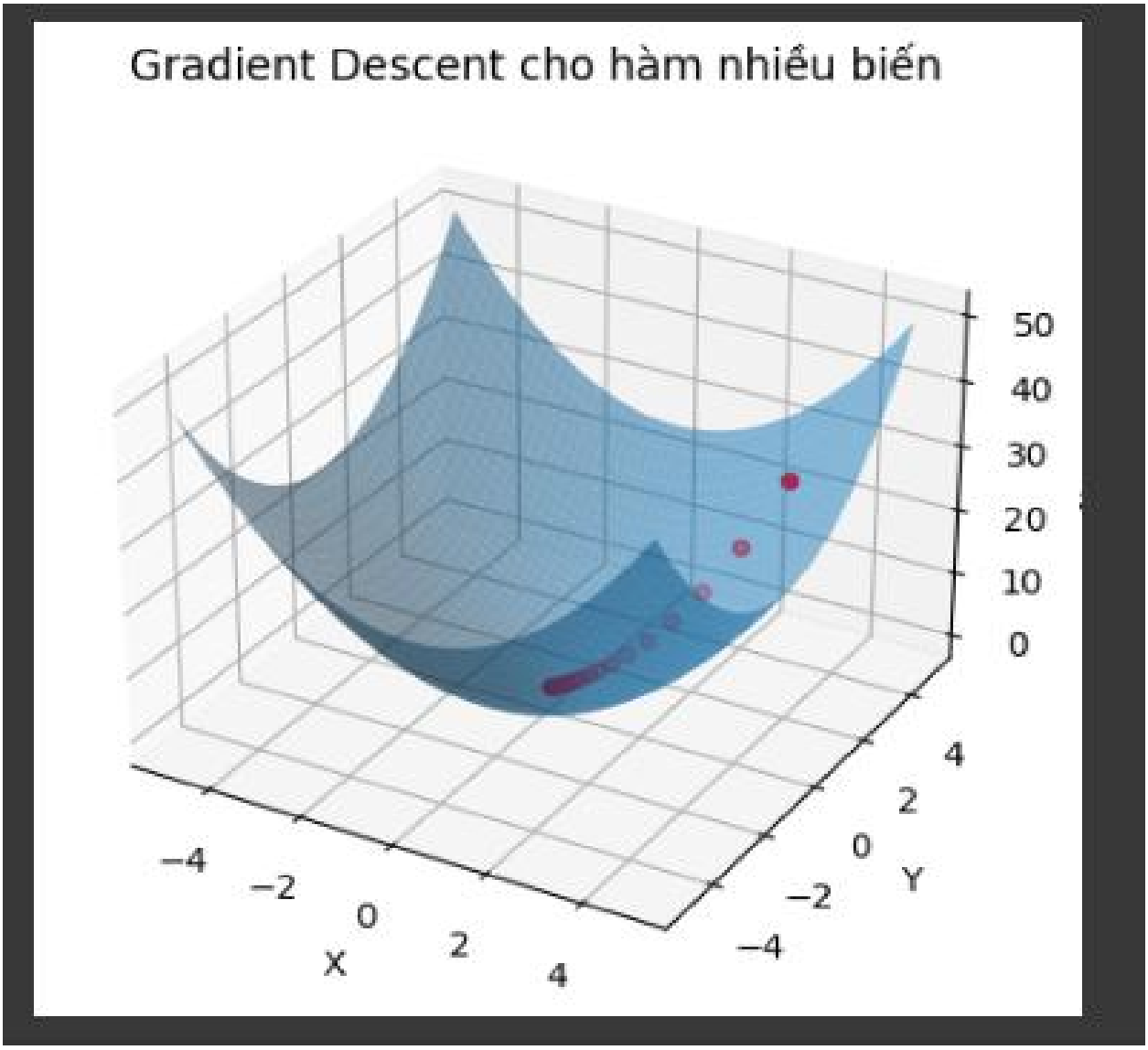
- Công thức: xnew = xold - learningrate.gradient(x)

a. Gradient cho hàm 1 biến :



- Điểm khởi đầu khác nhau sẽ có tác động đến quá trình hội tụ của thuật toán. Tốc độ học (learning rate) quá lớn hoặc quá nhỏ đều có ảnh hưởng đáng kể: nếu tốc độ học quá nhỏ, việc hội tụ sẽ diễn ra rất chậm và ảnh hưởng đến quá trình huấn luyện; ngược lại, nếu tốc độ học quá lớn, thuật toán có thể tiến nhanh đến điểm cần đạt sau vài vòng lặp nhưng không hội tụ do bước nhảy quá lớn, vòng lặp không thể tiến gần hơn tới điểm cần đạt và dễ bị lạc vào xung quanh điểm đó.

b. Gradient descent cho hàm nhiều biến :



Ưu điểm:

- Đơn giản và hiệu quả: Phương pháp Gradient Descent đơn giản và dễ triển khai, đặc biệt hiệu quả trong việc tối ưu hóa các hàm mục tiêu đa chiều.

- Tìm điểm cực tiểu địa phương: Gradient Descent có khả năng tìm điểm cực tiểu địa phương của hàm mục tiêu, giúp thuật toán hội tụ đến điểm tối ưu.

- Áp dụng rộng rãi: Thuật toán này có thể áp dụng cho nhiều mô hình machine learning khác nhau và trong nhiều lĩnh vực khác nhau.

Nhược điểm:

- Dễ rơi vào điểm cực tiểu địa phương: Gradient Descent dễ dàng rơi vào điểm cực tiểu địa phương thay vì điểm cực tiểu toàn cục, đặc biệt khi hàm mục tiêu không phải lúc nào cũng là một hàm lồi.

- Phụ thuộc vào learning rate: Việc chọn learning rate (tốc độ học) là yếu tố quan trọng. Nếu chọn learning rate quá lớn, thuật toán có thể không hội tụ hoặc dao động quanh điểm cực tiểu. Ngược lại, nếu chọn learning rate quá nhỏ, thuật toán có thể hội tụ rất chậm.

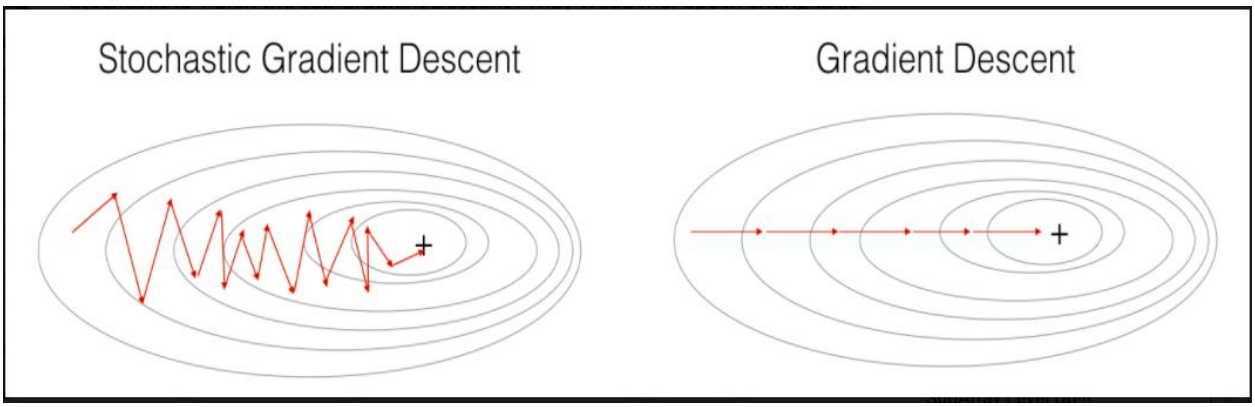
- Khó xử lý với các hàm không liên tục hoặc không khả vi: Gradient Descent yêu cầu hàm mục tiêu phải liên tục và khả vi tại mọi điểm trong không gian của nó, điều này có thể là một hạn chế đối với các bài toán có hàm mục tiêu phức tạp.

2. Stochastic Gradient Descent (SGD):

Thay vì sử dụng toàn bộ tập huấn luyện, Stochastic Gradient Descent (SGD) lấy một phần tử ngẫu nhiên từ tập huấn luyện để tính toán lại vector độ dốc chỉ dựa trên điểm dữ liệu đó, lặp lại quá trình này cho đến khi kết thúc. Việc tính toán dựa trên một điểm dữ liệu giúp thuật toán hoạt động nhanh hơn vì chỉ cần xử lý ít dữ liệu trong mỗi vòng lặp, điều này cũng hỗ trợ huấn luyện mô hình với dữ liệu lớn vì mỗi vòng lặp chỉ tập trung vào một điểm dữ liệu duy nhất.

Tuy nhiên, tính ngẫu nhiên của dữ liệu trong quá trình huấn luyện khiến hàm chi phí của SGD biến động không đồng đều, không giảm như Batch GD, mà biến đổi theo thời gian. Khi thuật toán tiệm cận cực tiểu, giá trị hàm chi phí không ổn định mà liên tục biến đổi. Mặc dù khi đạt điều kiện dừng, ta có thể có được bộ tham số cuối cùng khá tốt, nhưng chưa chắc đã tối ưu hóa thực sự.

Điều này có thể giúp thuật toán tránh khỏi cực tiểu địa phương khi hàm chi phí thay đổi liên tục, cho phép SGD có khả năng tìm kiếm cực trị toàn cục hơn Batch Gradient Descent. Tuy nhiên, việc chọn ngẫu nhiên dữ liệu cũng có nghĩa là thuật toán không bao giờ đạt được cực tiểu của hàm chi phí.



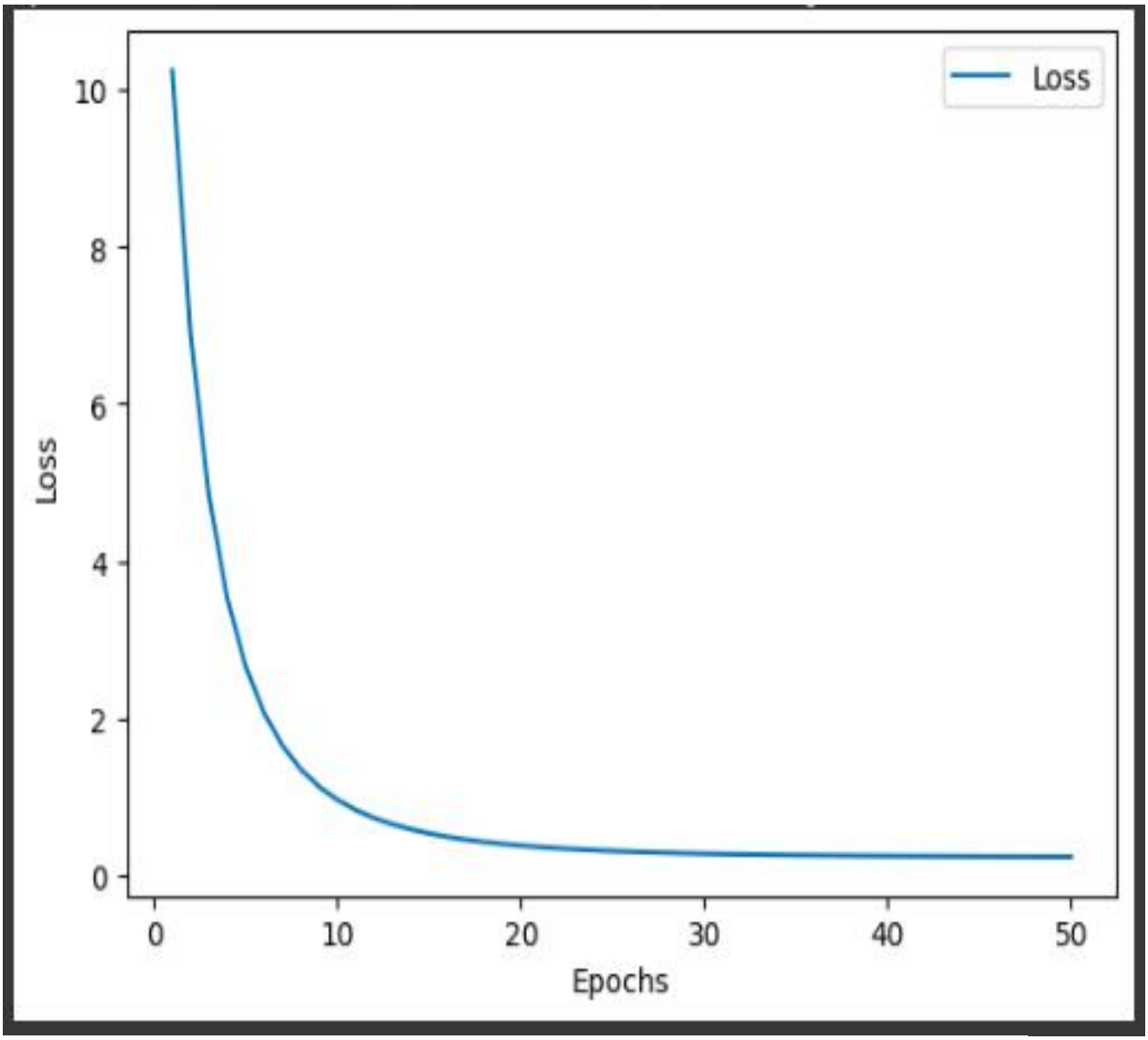
SGD được ưa chuộng vì khả năng tối ưu hiệu quả trên dữ liệu lớn. Trái với GD tính toán đạo hàm trên toàn bộ dữ liệu, SGD chỉ sử dụng một điểm dữ liệu (hoặc một vài điểm nhỏ) trong mỗi lần cập nhật, giúp giảm thiểu thời gian tính toán.

Điều này đặc biệt hữu ích trong học trực tuyến (online learning) khi dữ liệu thay đổi liên tục; SGD chỉ cần cập nhật trên dữ liệu mới mà không cần tính toán lại trên toàn bộ tập dữ liệu. Mặc dù SGD có thể di chuyển theo hình zig-zag, nó vẫn có thể nhanh chóng đạt được nghiệm tốt, đặc biệt là trên tập dữ liệu lớn, trong khi GD có thể đòi hỏi nhiều vòng lặp hơn để đạt được kết quả tương tự.

a. Giảm dần learning rate:

- Một phương pháp được đưa ra để giải quyết vấn đề này là giảm dần learning rate trong quá trình huấn luyện. Ban đầu, các vòng lặp sẽ sử dụng learning rate lớn để thoát khỏi cực tiểu địa phương, sau đó dần giảm tốc độ học để tiếp cận cực tiểu toàn cục. Hàm xác định learning rate ở mỗi vòng lặp được gọi là hàm lên lịch cho tốc độ học.

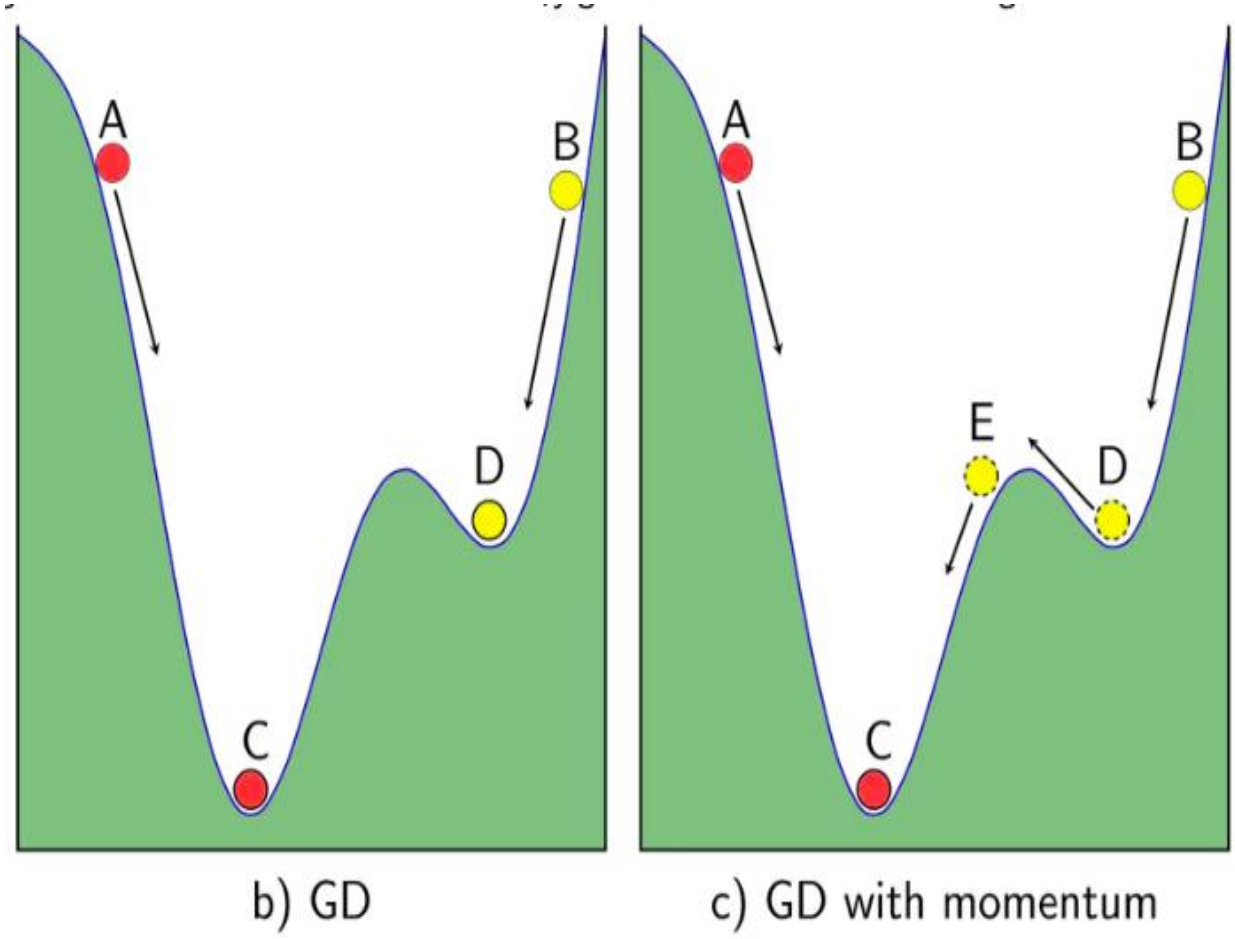
- Nếu learning rate giảm quá nhanh, thuật toán có thể dừng ở điểm cực tiểu địa phương hoặc chấm dứt trước khi đạt cực tiểu. Ngược lại, nếu learning rate giảm quá chậm, hàm chi phí có thể dao động lâu và cuối cùng dẫn đến một nghiệm tối ưu nếu việc huấn luyện dừng quá sớm.



Ưa thích tối ưu hóa Stochastic Gradient Descent (SGD) đến từ khả năng xử lý cơ sở dữ liệu lớn, điều mà Gradient Descent (GD) không thể thực hiện. Tuy nhiên, SGD vẫn phải đối mặt với hai hạn chế lớn của phương pháp gradient descent, bao gồm việc cần điều chỉnh learning rate và chọn điểm dữ liệu ban đầu. Để vượt qua những hạn chế này, thường ta kết hợp SGD với các thuật toán khác như Momentum, AdaGrad,... Những phương pháp này sẽ được giới thiệu cụ thể trong phần tiếp theo.

3. Gradient Descent với Momentum:

Để khắc phục các hạn chế trên của thuật toán Gradient Descent người ta dùng gradient descent with momentum.



Để giải thích Gradient Descent với Momentum, hãy xem xét từ góc độ vật lí: Nếu chúng ta thả hai viên bi từ hai điểm khác nhau A và B trên một dốc, viên bi tại điểm A sẽ trượt xuống và dừng ở điểm C, trong khi viên bi ở điểm B sẽ trượt xuống và dừng ở điểm D. Tuy nhiên, nếu muốn viên bi tại điểm B không dừng lại ở điểm D (điểm cực tiểu cục bộ) mà tiếp tục lăn tới điểm C (điểm cực tiểu toàn cục), ta cần cung cấp cho viên bi đó một vận tốc ban đầu đủ lớn để vượt qua điểm E và tiếp tục đến điểm C.

Dựa trên ý tưởng này, thuật toán Momentum được phát triển để cung cấp đà (momentum) cho quá trình tối ưu hóa. Bằng cách tích lũy đà từ các gradient trước đó, thuật toán giúp cho việc di chuyển trên bề mặt hàm mất mát trở nên mượt mà hơn và

có khả năng vượt qua các thung lũng và các điểm cực tiểu cục bộ để đạt tới điểm cực tiểu toàn cục.

Gradient Descent với Momentum là một phiên bản cải tiến của thuật toán Gradient Descent (GD) thông thường, nhằm cải thiện tốc độ hội tụ và tránh được những dao động không cần thiết khi tối ưu hóa.

Trong thuật toán GD cơ bản, việc cập nhật tham số dựa trên đạo hàm của hàm mất mát tại điểm hiện tại. Tuy nhiên, GD có thể di chuyển chậm hoặc bị lạc vào các điểm địa phương tối ưu. Momentum giải quyết vấn đề này bằng cách tích lũy trọng số cho các bước trước đó để tăng đà (momentum) cho việc cập nhật tham số.

Công thức cập nhật trong Gradient Descent với Momentum là:

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

$$\theta = \theta - v_t$$

Trong đó:

+ θ là tham số cần tối ưu.

+ ∇J(θ) là đạo hàm của hàm mất mát theo θ.

+ η là learning rate.

+ γ là hệ số momentum, thường nằm trong khoảng (0, 1).

+ Vt là véc-tơ momentum tại bước thời điểm t. Nó tích lũy độ lớn và hướng của các gradient trước đó. Khi gradient theo một hướng nhất định giữ không đổi hoặc thay đổi nhẹ, momentum giúp tăng tốc độ cho quá trình di chuyển và tránh bị rơi vào các điểm cục bộ tối ưu.

Ưu điểm:

- Tăng tốc độ hội tụ: Momentum giúp gia tăng đà di chuyển của thuật toán, đặc biệt hiệu quả khi đối mặt với thung lũng hoặc các điểm cực tiểu cục bộ. Điều này giúp vượt qua các thách thức và tiến gần hơn đến điểm tối ưu.

- Giảm dao động không cần thiết: Hệ số momentum làm mịn quá trình di chuyển và giảm thiểu dao động không mong muốn, đặc biệt khi đối mặt với nhiễu trong dữ liệu.

- Hội tụ nhanh hơn: Momentum tích lũy qua các bước trước đó, giúp Gradient Descent với Momentum thường hội tụ nhanh hơn so với phương pháp cơ bản, đặc biệt trên bề mặt hàm mất mát phức tạp.

Nhược điểm:

- Điều chỉnh hệ số momentum: Việc chọn hệ số momentum có thể ảnh hưởng đến hiệu suất của thuật toán. Nếu chọn quá lớn, có thể dẫn đến việc thuật toán vượt quá mục tiêu hoặc gây ra dao động không ổn định.

- Khó khăn trong việc tinh chỉnh siêu tham số: Ngoài hệ số momentum, Gradient Descent với Momentum cũng cần điều chỉnh learning rate và các siêu tham số khác, việc điều chỉnh các siêu tham số này có thể phức tạp.

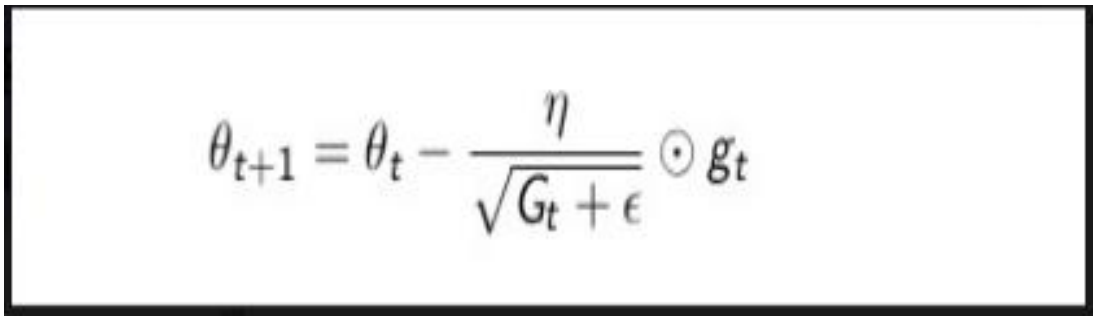
- Khả năng vượt quá mục tiêu: Đôi khi, do đà tích lũy, thuật toán có thể vượt quá mục tiêu tối ưu và không dừng lại ở điểm mong muốn.

4. Adagrad:

Adagrad là một thuật toán tối ưu hóa trong Machine Learning, điều chỉnh tỷ lệ học (learning rate) của từng tham số dựa trên lịch sử của gradient của tham số đó. Thuật toán này được xây dựng để tự động điều chỉnh learning rate để mỗi tham số có thể được cập nhật với tỷ lệ khác nhau, phụ thuộc vào mức độ biến đổi của gradient.

Trọng tâm của Adagrad là sử dụng bảng tính toán tổng bình phương của gradient đã tính toán trước đó cho mỗi tham số. Dựa trên thông tin này, learning rate được điều chỉnh để giảm khi gradient lớn và tăng khi gradient nhỏ, nhằm tối ưu hóa tỷ lệ học cho mỗi tham số.

Công thức cập nhật của Adagrad cho mỗi tham số w tại thời điểm t:



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

Trong đó:

+ n : hằng số

+ gt : gradient tại thời điểm t.

+ ϵ : hệ số tránh lỗi (chia cho mẫu bằng 0).

+ G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

Ưu điểm:

Adagrad có một ưu điểm rõ ràng là khả năng tự động điều chỉnh learning rate mà không cần can thiệp bằng tay. Bằng cách đặt tốc độ học mặc định là 0.01, thuật toán này sẽ tự động điều chỉnh mức độ này mà không cần sự can thiệp của người dùng.

Nhược điểm:

Điểm yếu của Adagrad là việc tổng bình phương của biến thiên tăng lên theo thời gian, dẫn đến việc tốc độ học giảm đến mức rất nhỏ, làm cho quá trình huấn luyện trở nên đóng băng và không hiệu quả.

5. RMSprop:

- RMSprop giải quyết vấn đề tỷ lệ học giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

$$E[g^2]_t = 0{,}9E[g^2]_{t\text{-}1} + 0{,}1g_t^2$$

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

RMSprop đem lại một ưu điểm đáng chú ý nhất, đó là khắc phục vấn đề về tốc độ học giảm dần của Adagrad. Sự giảm dần này có thể dẫn đến quá trình huấn luyện trở nên chậm dần và gây ra sự đóng băng không mong muốn. Tuy nhiên, RMSprop cũng có nhược điểm riêng của nó. Thuật toán này có thể hội tụ đến các điểm cực tiểu cục bộ thay vì điểm cực tiểu toàn cục như Momentum. Vì vậy, một giải pháp thường được áp dụng là kết hợp cả hai thuật toán, Momentum và RMSprop, để tạo ra thuật toán tối ưu hóa Adam. Phần tiếp theo sẽ trình bày về Adam và cách hoạt động của nó.

6. Adam:

Adam là sự kết hợp giữa Momentum và RMSprop. Nếu áp dụng vào hiện tượng vật lý, Momentum có thể được so sánh với việc một quả cầu lao xuống dốc, trong khi đó, Adam giống như một quả cầu với trọng lượng lớn và có ma sát. Điều này giúp nó vượt qua local minimum một cách dễ dàng hơn, tiếp cận global minimum và khi đạt được global minimum, nó không mất nhiều thời gian dao động xung quanh mục tiêu vì ma sát giúp nó dễ dàng dừng lại.

Tương tự như Adadelta và RMSprop, Adam duy trì trung bình bình phương của độ dốc quá khứ (vt) cũng như trung bình của độ dốc quá khứ (mt), tương tự như khái niệm của Momentum.

Trong khi Momentum tương tự như quả cầu lao xuống dốc, Adam lại có đặc tính giống như một quả cầu rất nặng và có ma sát, giúp nó vượt qua local minimum một cách dễ dàng hơn và đạt tới điểm tối ưu nhất (flat minimum).

Adam có thể đạt được hiệu ứng Heavy Ball with Friction (HBF) nhờ vào hệ số (mt/ sqrt(vt)).

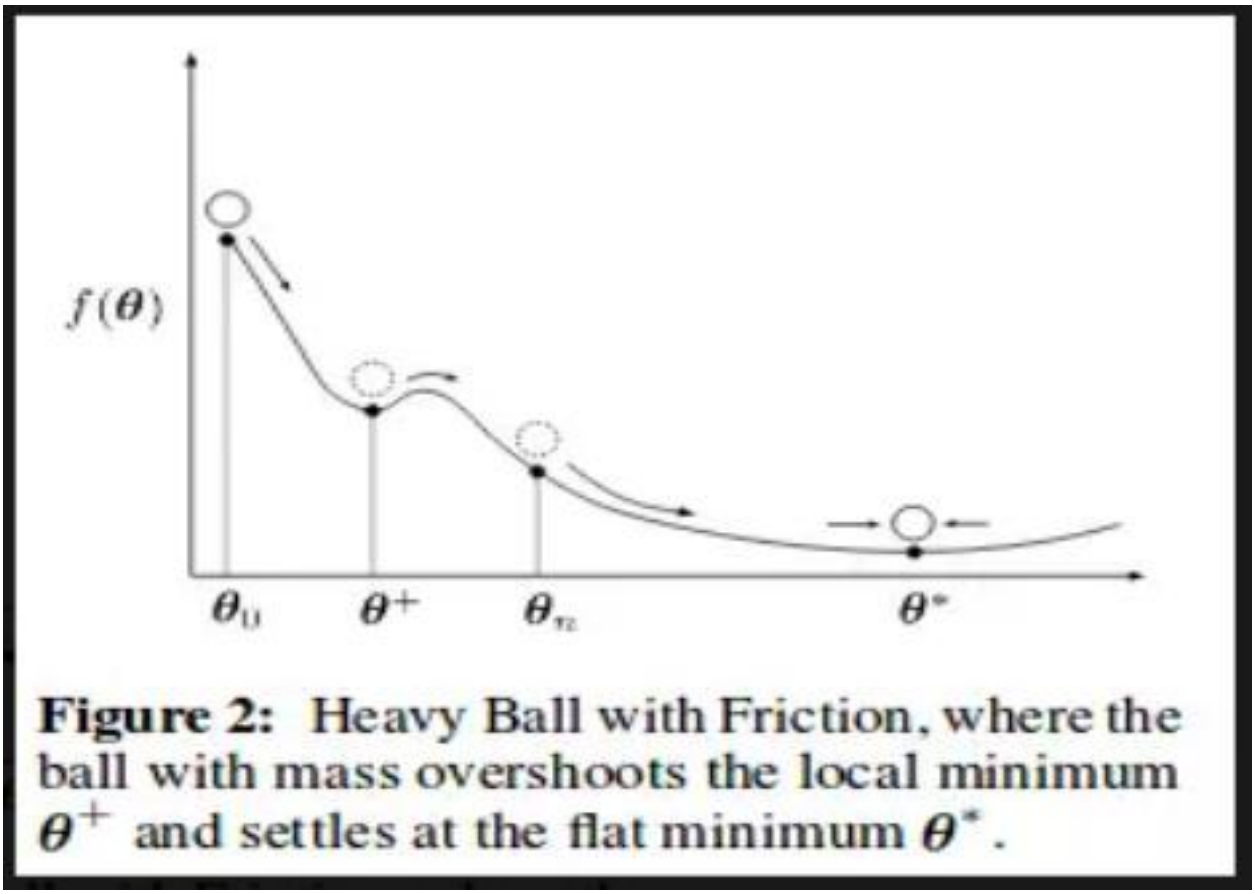


Figure 2: Heavy Ball with Friction, where the ball with mass overshoots the local minimum θ^+ and settles at the flat minimum θ^*.

Công thức update của nó là:

$$
\begin{aligned}
\boldsymbol{g}_n &\longleftarrow \nabla f(\boldsymbol{\theta}_{n-1}) \\
\boldsymbol{m}_n &\longleftarrow (\beta_1/(1-\beta_1^n))\,\boldsymbol{m}_{n-1} + ((1-\beta_1)/(1-\beta_1^n))\,\boldsymbol{g}_n \\
\boldsymbol{v}_n &\longleftarrow (\beta_2/(1-\beta_2^n))\,\boldsymbol{v}_{n-1} + ((1-\beta_2)/(1-\beta_2^n))\,\boldsymbol{g}_n \odot \boldsymbol{g}_n \\
\boldsymbol{\theta}_n &\longleftarrow \boldsymbol{\theta}_{n-1} - a\,\boldsymbol{m}_n/\left(\sqrt{\boldsymbol{v}_n} + \epsilon\right),
\end{aligned}
$$

- Tổng kết các công thức tối ưu:

Name	Update Rule
SGD	$\Delta\theta_t = -\alpha g_t$
Momentum	$m_t = \gamma m_{t-1} + (1-\gamma) g_t,$ $\Delta\theta_t = -\alpha\, m_t$
Adagrad	$G_t = G_{t-1} + g_t^2,$ $\Delta\theta_t = -\alpha\, g_t G_t^{-1/2}$
Adadelta	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\Delta\theta_t = -\alpha g_t v_t^{-1/2} D_{t-1}^{1/2},$ $D_t = \beta_1 D_{t-1} + (1-\beta_1)(\Delta\theta_t/\alpha)^2$
RMSprop	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\Delta\theta_t = -\alpha\, g_t v_t^{-1/2}$
Adam	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$ $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$ $\hat{m}_t = m_t/(1-\beta_1^t),$ $\hat{v}_t = v_t/(1-\beta_2^t),$ $\Delta\theta_t = -\alpha\, \hat{m}_t \hat{v}_t^{-1/2}$

CHAPTER 2 – ĐỊNH HƯỚNG VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG GIẢI PHÁP MACHINE LEARNING

1.1 Continual Learning

Học Liên Tục (hay còn gọi là Học Trường Sinh hoặc Học Tăng Dần) đóng vai trò quan trọng trong lĩnh vực Học Máy khi mô hình tiếp tục được huấn luyện và cải thiện theo thời gian thông qua việc tiếp nhận dữ liệu mới mà không quên đi kiến thức đã học trước đó. Điều này rất hữu ích khi dữ liệu thay đổi theo thời gian hoặc khi muốn mở rộng và cải thiện mô hình mà không cần phải huấn luyện lại từ đầu.

Trong Học Liên Tục, có một số thách thức phải đối mặt bao gồm:

- Lãng Quên Thảm Họa: Đây là hiện tượng khi mô hình quên đi dữ liệu đã học trước đó khi nó được huấn luyện với dữ liệu mới. Điều này xảy ra do thông tin cũ bị ghi đè bởi thông tin mới trong quá trình huấn luyện.

- Sự Can Thiệp: Hiện tượng này xảy ra khi dữ liệu đã học trước đó ảnh hưởng đến khả năng học của dữ liệu mới. Dữ liệu cũ có thể gây nhiễu hoặc xung đột với dữ liệu mới, làm giảm hiệu suất học.

- Khả năng Mở Rộng: Khi số lượng nhiệm vụ hoặc dữ liệu ngày càng tăng, khả năng mở rộng của mô hình để xử lý các nhiệm vụ mới và dữ liệu lớn là một thách thức quan trọng.

Để xây dựng giải pháp Học Máy cho Học Liên Tục, có một số phương pháp và kỹ thuật quan trọng:

- Phương Pháp Dựa Trên Ghi Nhớ: Bao gồm việc lưu trữ các mẫu dữ liệu đã học trước đó và sử dụng chúng để huấn luyện lại mô hình khi có dữ liệu mới. Kết hợp dữ liệu cũ và mới giúp giảm thiểu việc quên kiến thức cũ.

- Kỹ Thuật Chính Quy Hóa: Sử dụng các kỹ thuật chính quy hóa như Elastic Weight Consolidation (EWC) và Synaptic Intelligence (SI) để bảo vệ kiến thức đã học trước đó bằng cách áp đặt ràng buộc lên các trọng số của mô hình.

- Phương Pháp Kiến Trúc: Sử dụng các phương pháp kiến trúc như Progressive Neural Networks (PNN) và Memory Aware Synapses (MAS) để duy trì và sử dụng hiệu quả kiến thức đã học trước đó.

1.2 Test Production

Kiểm Tra Sản Phẩm liên quan đến việc đảm bảo rằng mô hình Học Máy hoạt động tốt trên dữ liệu mới và không quen thuộc. Khi triển khai giải pháp Học Máy vào môi trường thực tế, việc kiểm tra sản phẩm đảm bảo rằng mô hình hoạt động như dự kiến và đáp ứng các yêu cầu về chất lượng và hiệu suất.

Các hoạt động liên quan đến Kiểm Tra Sản Phẩm bao gồm:

- Kiểm tra chất lượng: Đảm bảo rằng mô hình Học Máy đáp ứng các tiêu chí chất lượng như độ chính xác, độ phân loại chính xác, độ tin cậy và các yêu cầu khác tùy thuộc vào bài toán cụ thể. Thường sử dụng tập dữ liệu kiểm tra độc lập để đánh giá hiệu suất của mô hình.
- Đo lường hiệu suất: Đánh giá hiệu suất và độ tin cậy của mô hình trên dữ liệu thực tế. Có thể bao gồm việc thu thập dữ liệu từ hệ thống thực tế, đánh giá mô hình trên dữ liệu này và theo dõi các chỉ số hiệu suất theo thời gian.
- Kiểm tra tính ổn định: Đảm bảo rằng mô hình hoạt động ổn định trên thực tế và không gặp các vấn đề không mong muốn như lỗi, sự cố hoặc giảm hiệu suất sau một thời gian sử dụng.
- Đánh giá về tốc độ và khả năng mở rộng: Đánh giá tốc độ và khả năng mở rộng của mô hình khi áp dụng cho dữ liệu lớn hoặc trong môi trường có yêu cầu thời gian thực.

Để thành công trong kiểm tra sản phẩm, quy trình kiểm tra và đánh giá cần được thiết kế và thực hiện cẩn thận. Đồ thì, việc theo dõi và đánh giá liên tục của mô hình trong môi trường thực tế rất quan trọng để phát hiện và giải quyết các vấn đề kịp thời, đảm bảo hiệu suất và chất lượng của giải pháp học máy.

Nhìn chung, quá trình Test Production đóng vai trò quan trọng trong việc đảm bảo mô hình học máy hoạt động một cách hiệu quả và ổn định khi triển khai vào môi trường thực tế. Nó không chỉ đánh giá hiệu suất của mô hình trên dữ liệu mới mà còn đảm bảo rằng mô hình không gặp vấn đề không mong muốn và có khả năng thích nghi với môi trường thực tế. Quá trình này cần sự cẩn thận và theo dõi liên tục để đảm bảo rằng mô hình hoạt động ổn định và đáp ứng được các yêu cầu trong quá trình triển khai thực tế.

Việc thiết kế quy trình kiểm tra cũng như quy trình đánh giá cẩn thận là chìa khóa để đảm bảo mô hình học máy hoạt động hiệu quả và đáng tin cậy trong môi trường thực tế, đồng thời cung cấp thông tin quan trọng để cải thiện và tối ưu hóa mô hình trong tương lai.

Tài Liệu Tham Khảo

1. https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

2. https://ndquy.github.io/

3. https://paperswithcode.com/task/continual-learning