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

NGUYỄN HỮU TOÀN - 52100331

BÁO CÁO CUỐI KỲ
NHẬP MÔN HỌC MÁY

Người hướng dẫn
PGS.TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

MỤC LỤC

CHƯƠNG 1. TÌM HIỂU, SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 Tìm hiểu các phương pháp Optimizer(tối ưu hóa) trong huấn luyện mô hình học máy

1.1.1 Tối ưu hóa trong huấn luyện học máy

- Tối ưu hóa học máy là quá trình liên tục để cải thiện độ chính xác của mô hình, giảm lỗi dự đoán thông qua việc điều chỉnh cấu hình mô hình. Mô hình học máy học cách ánh xạ dữ liệu đầu vào vào dự đoán đầu ra bằng cách xấp xỉ mối quan hệ giữa chúng. Mục tiêu chính là giảm thiểu sai số giữa dự đoán và thực tế.

- Tối ưu hóa đo lường thông qua hàm mất mát, thể hiện sự khác biệt giữa giá trị dự đoán và giá trị thực tế của dữ liệu. Quá trình này nhằm giảm thiểu hàm mất mát hoặc khoảng cách giữa dự đoán và thực tế, làm cho mô hình trở nên chính xác hơn.

- Việc làm sạch và chuẩn bị dữ liệu đào tạo cũng được coi là một bước quan trọng để tối ưu hóa học máy. Dữ liệu thô cần được chuyển đổi thành dữ liệu huấn luyện để mô hình có thể học được. Tuy nhiên, quan điểm tổng thể là tối ưu hóa học máy liên quan chặt chẽ đến việc cải thiện lặp lại của mô hình thông qua việc điều chỉnh siêu tham số, như tốc độ học hoặc số lượng cụm phân loại.

- Quan trọng nhất, việc chọn lựa siêu tham số tối ưu là chìa khóa để đảm bảo mô hình học máy đạt được độ chính xác và hiệu quả cao nhất. Bằng cách điều chỉnh siêu tham số này một cách khôn ngoan, chúng ta có thể đạt được sự cân bằng giữa khả năng tự tinh chỉnh của mô hình và khả năng áp dụng cho nhiều tình huống khác nhau.

- Một số yếu tố cần biết có ảnh hưởng đến bài toán tối ưu hóa trong quá trình huấn luyện học máy:

+ Learning Rate (Tốc độ học): là hệ số quyết định độ lớn của bước di chuyển tại mỗi vòng lặp của Gradient Descent (GD). Learning Rate đóng vai quyết định tốc

độ học của mô hình. Nếu quá cao, có thể bỏ qua giải pháp tốt; nếu quá thấp, có thể làm chậm quá trình học.

+ Batch, Epoch, Iteration:

- Batch: Số lượng mẫu dữ liệu được sử dụng trong mỗi vòng lặp của Gradient Descent.
- Epoch: Một epoch là một lượt qua toàn bộ tập dữ liệu huấn luyện.
- Iteration: Số lượt chạy qua toàn bộ tập dữ liệu huấn luyện trong quá trình Gradient Descent.

+ Local Minimum và Global Minimum:

- Local Minimum: Là điểm dưới đáy của một khu vực nhất định của hàm mất mát.
- Global Minimum: Là điểm dưới đáy của toàn bộ không gian của hàm mất mát.

+ Convergence Criteria: Là điều kiện dừng để xác định khi nào dừng lại trong quá trình huấn luyện. Nó có vai trò ngăn chặn quá trình luyện tập khi đã đạt được giải pháp đủ tốt, giúp tiết kiệm tài nguyên tính toán.

- Một số phương pháp tối ưu hóa trong huấn luyện mô hình học máy: Gradient Descent, Stochastic Gradient Descent (SGD), Gradient Descent with Momentum, Adam, Adagrad, RMSprop. Sau đây là tìm hiểu về từng phương phương pháp.

1.1.2 Gradient Descent (Giảm độ dốc)

1.1.2.1 Giới thiệu

- Giảm độ dốc là một thuật toán tối ưu hóa được sử dụng trong học máy để giảm thiểu hàm chi phí bằng cách điều chỉnh lặp đi lặp lại các tham số theo hướng của độ dốc âm, nhằm tìm ra bộ tham số tối ưu. Vậy hàm chi phí là gì? Hàm chi phi là một chức năng đo lường hiệu suất của một mô hình đối với bất kỳ dữ liệu nhất định nào. Hàm chi phí định lượng sai số giữa giá trị dự đoán và giá trị kỳ vọng và trình bày nó dưới dạng một số thực duy nhất.

- Công thức hàm chi phí:

+ Đối với bài toán Hồi quy tuyến tính (Linear Regression):

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

+ Đối với bài toán Hồi quy logistic nhị phân (Logistic Regression):

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

+ Trong đó:

- m là số lượng mẫu.
- $h_\theta(x^{(i)})$ là giá trị dự đoán của mô hình.
- $y^{(i)}$là giá trị thực tế.

- Hàm chi phí biểu thị sự khác biệt giữa đầu ra dự đoán của mô hình và đầu ra thực tế. Mục tiêu của việc giảm độ dốc là tìm ra tập hợp các tham số giúp giảm thiểu sự khác biệt này và cải thiện hiệu suất của mô hình.

- Thuật toán hoạt động bằng cách tính toán độ dốc của hàm chi phí. Tuy nhiên, vì mục tiêu là cực tiểu hóa hàm chi phí nên việc giảm độ dốc sẽ di chuyển theo hướng ngược lại với độ dốc, được gọi là hướng độ dốc âm.

- Bằng cách cập nhật lặp đi lặp lại các tham số của mô hình theo hướng gradient âm, độ dốc giảm dần hội tụ về phía tập tham số tối ưu mang lại chi phí thấp nhất.

- Công thức cập nhật của hàm 1 biến trong mỗi lần lặp là:

$$\theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

- Công thức cập nhật của hàm đa biến trong mỗi lần lặp là:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial \theta_j}J(\theta_0, \theta_1, \dots, \theta_n)$$

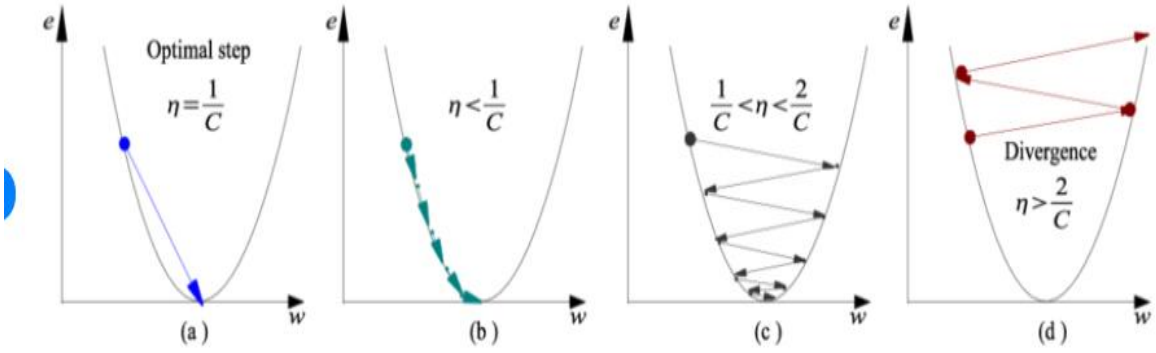
- Trong đó:

- θ_j là trọng số thứ j của mô hình.

- $\frac{\partial J(\theta)}{\partial \theta_j}$ là đạo hàm riêng của hàm chi phí của J theo tham số θ_j
- $J(\theta_0, \theta_1, \dots, \theta_n)$ là là hàm chi phí với $n + 1$ trọng số $\theta_0, \theta_1, \dots, \theta_n$
- α là learning rate (tốc độ học).
- $:=$ là toán tử gán, chỉ đơn giản là thực hiện phép tính cập nhật.

- Tốc độ học, một siêu tham số, xác định kích thước bước được thực hiện trong mỗi lần lặp, ảnh hưởng đến tốc độ và độ ổn định của quá trình hội tụ.

- Ví dụ về tốc độ học tập:



1a) Tốc độ học là tối ưu, mô hình hội tụ về mức tối thiểu.

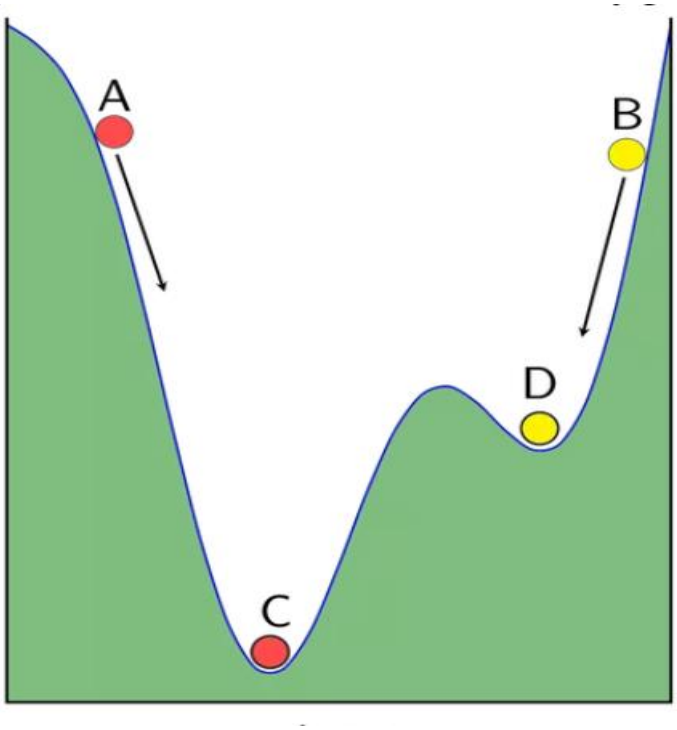
2b) Tốc độ học quá nhỏ, tốn nhiều thời gian hơn nhưng lại hội tụ ở mức tối thiểu. Đây là một trong số các hạn chế của phương pháp giảm độ dốc - hội tụ chậm.

3c) Tốc độ học cao hơn giá trị tối ưu, vượt quá nhưng hội tụ ($1/C < \eta < 2/C$).

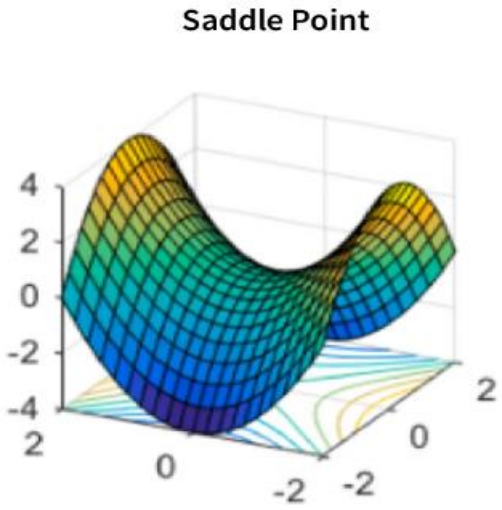
4d) Tốc độ học rất lớn, vượt quá và phân kỳ, rời xa mức tối thiểu, hiệu quả học tập giảm.

- Có một số vấn đề khi sử phương pháp giảm độ dốc:

+ Tối thiểu địa phương (Local minimum): Giảm độ dốc có thể bị kẹt trong cực tiểu cục bộ (gobal minimum), các điểm không phải là mức tối thiểu toàn cầu (local minimum) của hàm chi phí nhưng vẫn thấp hơn các điểm xung quanh. Điều này có thể xảy ra khi hàm chi phí có nhiều đáy và thuật toán bị kẹt trong một đáy thay vì đạt mức tối thiểu toàn cục.



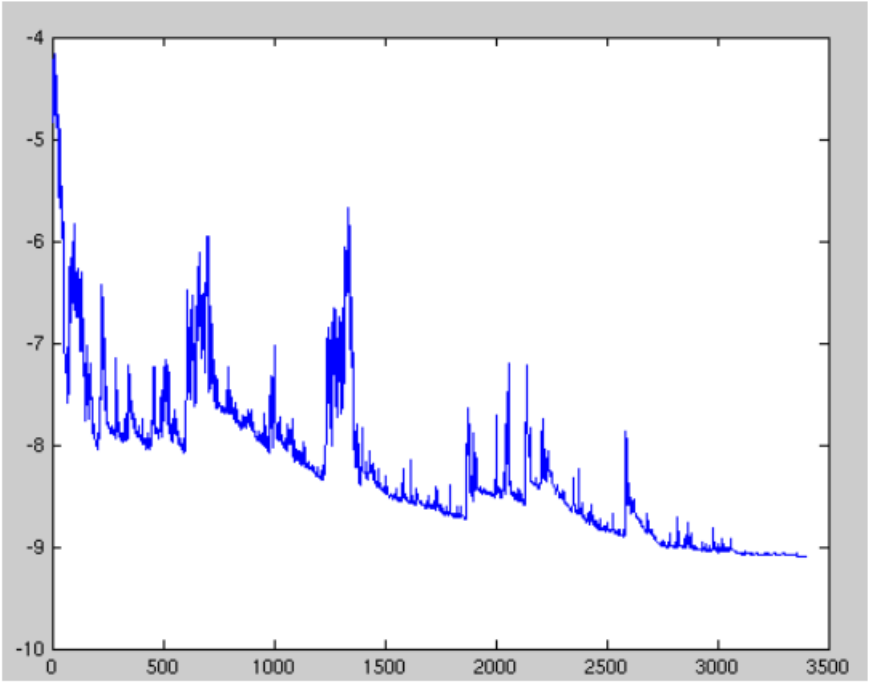
+ Điểm yên ngựa: là một điểm trong hàm chi phí trong đó một chiều có giá trị cao hơn các điểm xung quanh và chiều kia có giá trị thấp hơn. Độ dốc giảm dần có thể bị kẹt tại các điểm này vì độ dốc theo một hướng hướng tới giá trị thấp hơn, trong khi các độ dốc theo hướng khác hướng tới giá trị cao hơn.



- Ngoài ra Gradient Descent còn có các biến thể: Stochastic Gradient Descent (SGD), Gradient Descent with Momentum. Các biến thể trên ra đời nhằm khắc phục các hạn chế và tối ưu phương pháp Gradient Descent.

1.1.3 Stochastic Gradient Descent (SGD)

- Giảm tốc độ hàng loạt là một biến thể của Gradient Descent truyền thống nhưng thực hiện cập nhật trọng số sau mỗi điểm dữ liệu, thay vì sau mỗi lượt duyệt qua toàn bộ tập dữ liệu. Do đó, nó thường nhanh hơn nhiều và cũng có thể được sử dụng để học trực tuyến. SGD thực hiện cập nhật thường xuyên với phương sai cao khiến hàm mục tiêu dao động mạnh như trong hình.



- Trong khi độ dốc giảm dần hàng loạt hội tụ đến mức tối thiểu của lưu vực mà các tham số được đặt trong đó, thì sự biến động của SGD, một mặt, cho phép nó nhảy đến mức tối thiểu cục bộ mới và có khả năng tốt hơn. Mặt khác, điều này cuối cùng sẽ làm phức tạp thêm sự hội tụ đến mức tối thiểu chính xác, vì SGD sẽ tiếp tục tăng quá mức. Khi chúng ta giảm dần tốc độ học, SGD thể hiện hành vi hội tụ tương tự như giảm độ dốc hàng loạt, gần như chắc chắn hội tụ đến mức tối thiểu cục bộ hoặc toàn cục để tối ưu hóa không lồi và lồi tương ứng.

- Ví dụ cho mỗi điểm dữ liệu (x_i, y_i)với i là chỉ số của mẫu dữ liệu ta có công thức cập nhật sau:

$$\theta := \theta - \alpha \nabla J(\theta; x_i, y_i)$$

- Trong đó:

+ $\nabla J(\theta; x_i, y_i)$ là gradient của hàm chi phí $J(\theta)$ theo trọng số θ tính toán dựa trên mẫu dữ liệu (x_i, y_i).

+ α là learning rate (tốc độ học).

1.1.4 Gradient Descent with Momentum

1.1.4.1 Giới thiệu

- Giảm dần độ dốc dựa trên động lượng là một biến thể của thuật toán tối ưu hóa giảm độ dốc có thêm thuật ngữ động lượng vào quy tắc cập nhật.

- Động lượng được tính toán dưới dạng trung bình động của các gradient trong quá khứ và trọng số của các gradient trong quá khứ được kiểm soát bởi một siêu tham số có tên Beta.

- Động lượng giúp đẩy nhanh quá trình tối ưu hóa bằng cách cho phép các bản cập nhật tích lũy theo hướng giảm dần nhanh nhất. Điều này có thể giúp giải quyết một số vấn đề với việc giảm độ dốc vanilla, chẳng hạn như dao động, hội tụ chậm và bị kẹt ở cực tiểu cục bộ. Bằng cách sử dụng phương pháp giảm độ dốc dựa trên động lượng, có thể huấn luyện các mô hình học máy hiệu quả hơn và đạt được hiệu suất tốt hơn.

- Công thức cập nhật trọng số mới sẽ tích hợp vào nó một phần của gradient trước đó:

$$v = \beta v + (1 - \beta) \nabla J(\theta)$$

$$\theta := \theta - \alpha v$$

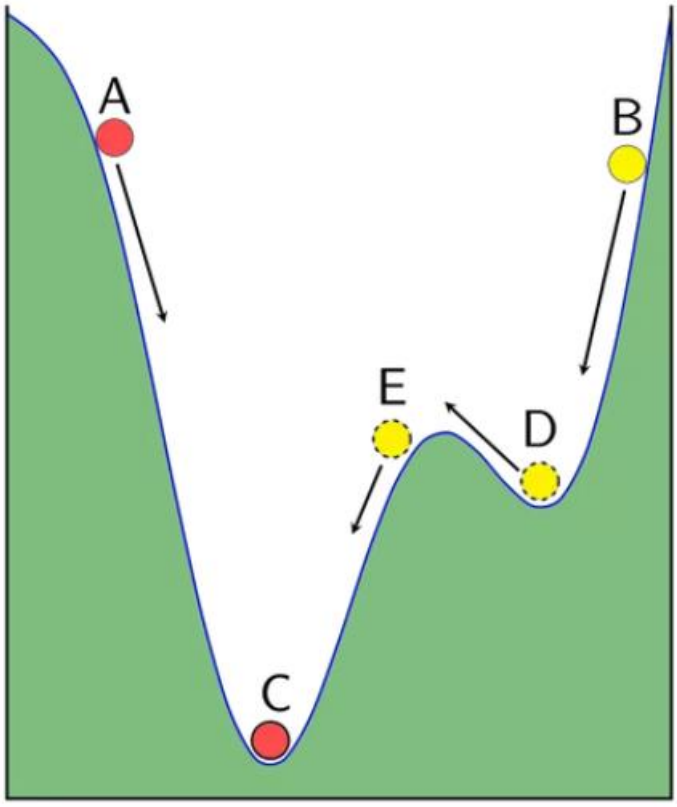
- Trong đó:

+ β là momentum được giới thiệu để đại diện cho đà của quá khứ.

+ v là vectơ momentum.

+ $\nabla J(\theta)$ là gradient của hàm chi phí $J(\theta)$ theo trọng số θ.

+ α là learning rate.



c) GD with momentum

- Theo hình ta thấy khi sử dụng động lượng cho phương pháp giảm độ dốc thì nó đã giúp quả bóng vượt qua được điểm D (gobal minimum) để đi đến điểm C (lobal minimum).

- Tính chất:

+ Khi gradient thay đổi hướng, momentum giúp tránh tình trạng quá giảm tốc độ hoặc dao động mạnh.

+ Có thể hiểu momentum như là một "trí nhớ" của quá khứ để ổn định hướng di chuyển.

1.1.5 Adagrad

- AdaGrad là một nhóm các thuật toán gradient phụ để tối ưu hóa ngẫu nhiên. Các thuật toán thuộc họ đó tương tự như độ dốc. Tên của AdaGrad xuất phát từ Adaptative Gradient. Nó điều chỉnh tốc độ học cho từng đặc điểm tùy thuộc vào hình

dạng ước tính của bài toán; đặc biệt, nó có xu hướng gán tỷ lệ học cao hơn cho các tính năng không thường xuyên, điều này đảm bảo rằng các cập nhật tham số phụ thuộc ít hơn vào tần suất mà phụ thuộc nhiều hơn vào mức độ liên quan. Vì lý do này, nó rất phù hợp để xử lý dữ liệu thưa thớt. Adagrad đã cải thiện đáng kể tính mạnh mẽ của SGD và sử dụng nó để đào tạo mạng lưới thần kinh quy mô lớn tại Google.

- Công thức cập nhật của Adagrad cho từng tham số w trong mỗi bước lặp là:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} g_t$$

- Trong đó:

+ w_t là giá trị tham số tại bước thời gian t

+ G_t là tổng bình phương của gradient cho tham số w tính đến thời điểm t.

+ α là learning rate, quyết định độ lớn của bước cập nhật.

+ g_t là gradient

+ ϵ là một số nhỏ (thường được sử dụng để tránh việc chia cho 0).

1.1.6 RMSprop

- RProp, hay chúng tôi gọi là tuyên truyền trung bình gốc có khả năng phục hồi, là thuật toán được sử dụng rộng rãi để học có giám sát với các mạng chuyển tiếp nguồn cấp dữ liệu nhiều lớp. Đây là một biến thể của phương pháp Gradient Descent dựa trên việc điều chỉnh learning rate của từng trọng số theo tỉ lệ của giá trị gradient. Đặc biệt là khi áp dụng cho các mạng nơ-ron sâu.

- 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.

- Công thức cập nhật:

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

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

- Trong đó:

+ θ_t là trọng số cần cập nhật,

+ g_t là gradient.

+ $E[g^2]_t$ là giá trị trung bình bình phương của gradient.

+ α là learning rate.

+ $\in$ là một số nhỏ (thường được sử dụng để tránh việc chia cho 0).

1.1.7 Adam

- Adam là một thuật toán tối ưu hóa có thể được sử dụng thay cho quy trình giảm độ dốc ngẫu nhiên cổ điển để cập nhật các trọng số mạng lặp đi lặp lại dựa trên dữ liệu huấn luyện

- Adam khác với phương pháp giảm độ dốc ngẫu nhiên cổ điển. Giảm dần độ dốc ngẫu nhiên duy trì một tốc độ học duy nhất (được gọi là alpha) cho tất cả các cập nhật trọng số và tốc độ học không thay đổi trong quá trình huấn luyện. Tốc độ học được duy trì cho từng trọng lượng và được điều chỉnh riêng biệt khi quá trình học diễn ra.

- Phương pháp này tính toán tốc độ học thích ứng riêng lẻ cho các tham số khác nhau từ ước tính khoảnh khắc đầu tiên và khoảnh khắc thứ hai của độ dốc.

- Adam đang kết hợp các ưu điểm của hai phần mở rộng khác của phương pháp giảm độ dốc ngẫu nhiên. Đặc biệt:

+ Phương pháp chuyển màu thích ứng (AdaGrad) duy trì tốc độ học trên mỗi tham số giúp cải thiện hiệu suất đối với các vấn đề có độ dốc thưa thớt.

+ Tuyên truyền bình phương trung bình gốc (RMSProp) cũng duy trì tốc độ học theo từng tham số được điều chỉnh dựa trên mức trung bình của độ lớn gần đây của độ dốc cho trọng số (ví dụ: tốc độ thay đổi của nó). Điều này có nghĩa là thuật toán thực hiện tốt các vấn đề trực tuyến và không cố định (ví dụ như nhiễu).

- Thay vì điều chỉnh tốc độ học tham số dựa trên khoảnh khắc trung bình đầu tiên (giá trị trung bình) như trong RMSProp, Adam cũng sử dụng mức trung bình của khoảnh khắc thứ hai của độ dốc (phương sai không tập trung).

- Cụ thể, thuật toán tính toán đường trung bình động hàm mũ của gradient và gradient bình phương, đồng thời các tham số beta1 và beta2 kiểm soát tốc độ phân rã

của các đường trung bình động này. Giá trị ban đầu của các đường trung bình động và giá trị beta1 và beta2 gần bằng 1,0 dẫn đến độ lệch của ước tính thời điểm về 0. Độ lệch này được khắc phục bằng cách trước tiên tính toán các ước tính sai lệch trước khi tính toán các ước tính đã điều chỉnh sai lệch.

- Trong khi momentum giống như một quả cầu lao xuống dốc, thì Adam lại giống như một quả cầu rất nặng và có ma sát (friction), nhờ vậy nó dễ dàng vượt qua local minimum và đạt tới điểm tối ưu nhất (flat minimum)

- Công thức cập nhật của Adam:

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

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

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

- Nó đạt được hiệu ứng Heavy Ball with Friction (HBF) nhờ vào hệ số $\frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}}$

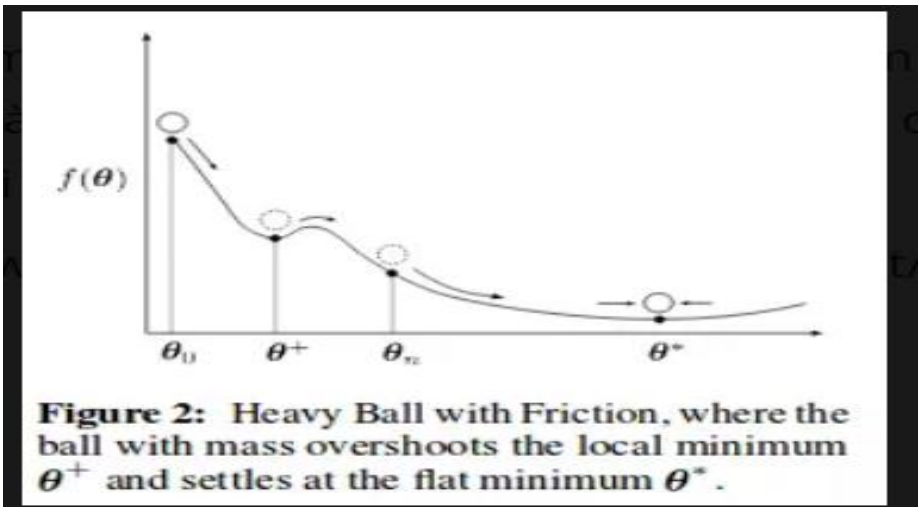


Figure 2: Heavy Ball with Friction, where the ball with mass overshoots the local minimum θ^+ and settles at the flat minimum θ^*.

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

	Ưu điểm	Nhược điểm
Gradient Descent	- Thuật toán gradient descent cơ bản, dễ hiểu. - Thuật toán đã giải quyết được vấn đề tối ưu model neural network bằng cách cập nhật trọng số sau mỗi vòng lặp. - Hoạt động tốt trên dữ liệu nhỏ và tập dữ liệu có cấu trúc tốt.	- Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate. - Nghiệm cho ra không thống nhất. - Chậm hội tụ trên các bài toán lớn hoặc có nhiều tham số. - Quanh quẩn bên đích vì bước nhảy quá lớn - Tốc độ học nhỏ ảnh hưởng đến tốc độ training
Stochastic Gradient Descent	- Thuật toán giải quyết được đối với cơ sở dữ liệu lớn mà GD không làm được. - Cập nhật thường xuyên, giúp tránh bị mắc kẹt ở các điểm cục bộ.	- Độ dao động cao đôi khi mất kiểm soát. - Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.
Gradient Descent with Momentum	- Thuật toán tối ưu giải quyết được vấn đề Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.	- Mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn.

	- Giảm độ dao động, giúp hội tụ nhanh hơn. - Hoạt động tốt trên dữ liệu có nhiều biên độ khác nhau.	
Adagrad	- Tự động điều chỉnh tỷ lệ học tập. - Hiệu quả cho dữ liệu thưa (sparse data).	- Tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ. - Giảm tỷ lệ học tập quá nhanh.
RMSprop	- Giải quyết được vấn đề tốc độ học giảm dần của Adagrad. - Giảm tác động của gradient nhiễu. - Hiệu quả trên các bài toán với độ lớn gradient thay đổi.	- Cần phải lựa chọn tỷ lệ học tập thủ công. - Chỉ cho kết quả nghiệm là local minimum.
Adam	- Hiệu suất cao trên nhiều loại mô hình. - Tự động điều chỉnh tỷ lệ học tập.	- Có thể nhạy cảm với siêu tham số.

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

2.1 Tìm hiểu về Continual Learning

- Trước khi tìm hiểu về mô hình học liên tục (Continual learning) hay học tập suốt đời, chúng ta hãy tìm hiểu dữ liệu của nó. Trên thực tế dữ liệu hiếm khi ở trạng thái tĩnh. Hãy xem thách thức mà một hệ thống giám sát chất lượng không khí phải đối mặt. Các cảm biến gửi dữ liệu liên tục, thường xuyên đo lường và cập nhật các chỉ số chất lượng không khí một cách tức thì. Vừa xử lý lượng lớn dữ liệu liên tục vừa phải đảm bảo tính chuẩn xác của các cảm biến.

- Vì vậy vai trò của học tập liên tục là vô cùng quan trọng trong việc xử lý các dữ liệu động này. Học tập liên tục giải quyết những thách thức này bằng cách cho phép các mô hình học máy thích ứng và phát triển cùng với việc thay đổi dữ liệu và nhiệm vụ. Thay vì bắt đầu lại từ đầu với mỗi luồng dữ liệu hoặc nhiệm vụ mới, các mô hình học tập liên tục sẽ xây dựng và lưu giữ kiến thức từ những trải nghiệm trước đó. Điều này cho phép họ tích lũy kiến thức chuyên môn, thích ứng với những thách thức mới và duy trì mức hiệu suất cao trong nhiều nhiệm vụ khác nhau.

- Về bản chất, học tập liên tục cung cấp khuôn khổ cho các mô hình học máy để đạt được những gì con người làm một cách tự nhiên-học hỏi từ kinh nghiệm, thích ứng với các tình huống mới và lưu giữ kiến thức tích lũy được trong suốt cuộc đời của họ.

- Học tập liên tục là một mô hình học máy tập trung vào các mô hình đào tạo để tiếp thu kiến thức mới và thích ứng với việc thay đổi dữ liệu theo thời gian. Ngược lại với học máy truyền thống, trong đó các mô hình thường được đào tạo trên các tập dữ liệu cố định và giả định rằng việc phân phối dữ liệu không đổi, học liên tục được thiết kế để xử lý các phân phối dữ liệu đang phát triển và liên tục học hỏi từ dữ liệu mới trong khi vẫn giữ được kiến thức từ những trải nghiệm trước đó.

- Một số khái niệm và thách thức chính liên quan đến việc học tập liên tục:

+ Sự quên lãng nghiêm trọng: một trong những thách thức chính trong việc học tập liên tục là ngăn chặn tình trạng quên lãng nghiêm trọng. Đây chính là hiện tượng một mô hình quên những thông tin đã học trước đó khi bắt đầu học tập những thông tin mới.

+ Quản lý overfitting: học liên tục đối mặt với nguy cơ overfitting lớn, vì mô hình cần thích ứng với dữ liệu mới mà không ảnh hưởng đến khả năng tổng quát hóa trên dữ liệu cũ.

+ Siêu học tập: là một cách tiếp cận khác có thể giúp các mô hình thích ứng nhanh chóng với các nhiệm vụ mới.

+ Sự quên lãng nghiêm trọng là một hiện tượng xảy ra trong học máy và trí tuệ nhân tạo, đặc biệt trong các tình huống liên quan đến việc học tập liên tục hoặc suốt đời. Sự quên lãng nghiêm trọng thường có các đặc điểm sau: Sự can thiệp của nhiệm vụ, ghi đè trọng số, dung lượng cố định, học tăng dần, …

- Cách triển khai học tập liên tục:

+ Thu thập dữ liệu liên tục

+ Tiền xử lý dữ liệu

+ Mở rộng mô hình

+ Học tập liên tục

+ Đánh giá và kiểm tra

+ Lưu trữ và quản lý mô hình

+ Tối ưu hóa

- Một số thuật toán học liên tục:

+ Mạng thần kinh tiến bộ (PNN)

+ Học không quên (LwF)

+ iCaRL (Học phân loại và biểu diễn tăng dần)

2.2 Tìm hiểu về Test Production

- Mục tiêu chính của kiểm thử trong môi trường production là đảm bảo rằng hệ thống hoạt động ổn định, hiệu quả và đáp ứng đúng đối với yêu cầu của người dùngMục tiêu chính của kiểm thử trong môi trường production là đảm bảo rằng hệ

thống hoạt động ổn định, hiệu quả và đáp ứng đúng đối với yêu cầu của người dùng. Nó không chỉ giúp đảm bảo sự ổn định của hệ thống mà còn đóng vai trò quan trọng trong việc duy trì và cải thiện chất lượng sản phẩm.

- Kiểm thử ứng dụng là luôn là 1 sự ưu tiên cao đối với bất kỳ tổ chức phát triển phần mềm nào. Hầu hết các tổ chức đều ưu tiên việc kiểm thử càng nhiều càng tốt trước khi triển khai lên môi trường production để chắc chắn rằng quá trình chuyển đổi sau triển khai là trơn tru nhất có thể.

- Bạn sẽ tìm thấy một tập hợp các lỗi mà bạn không tìm thấy khi kiểm thử trong các môi trường kiểm thử khác (dev, staging, hoặc pre-prod). Các lỗi được thu thập ở trong môi trường production sẽ giúp nhóm phát triển cô lập các lỗi để cải thiện chất lượng ứng dụng, từ đó cung cấp một trải nghiệm khách hàng tốt hơn.

- Và dù kiểm thử trên môi trường có thể sẽ gặp không ít những rủi ro vì vậy nếu buộc phải kiểm thử trên môi trường production thì hãy hết sức cẩn thận.

- Ưu điểm:

+ Đóng vai trò quan trọng trong việc thu thập phản hồi sớm từ khách hàng về các tính năng mới và trải nghiệm người dùng.

+ Khách hàng có cơ hội trải nghiệm trực tiếp và đưa ra ý kiến phản hồi, giúp đội ngũ phát triển hiểu rõ hơn về cách ứng dụng được sử dụng và nhận định các điểm mạnh cũng như điều cần cải thiện.

+ Ngăn chặn thảm họa thông qua kiểm thử phục hồi và khả năng phục hồi tốt hơn.

+ Giúp đảm bảo rằng bất kỳ vấn đề nào đã được xác định và giải quyết trước khi ảnh hưởng đến người dùng cuối và trải nghiệm của họ.

+ Giúp phát hiện và giải quyết mọi tình huống có thể xảy ra, đặc biệt là những vấn đề chỉ xuất hiện khi ứng dụng đang chạy trong điều kiện thực tế.

- Nhược điểm:

+ Để lộ ra những lỗ hổng tiềm năng.

+ Không thể phục hồi sau những sự hỗn loạn bất ngờ.

+ Thời gian kiểm thử gây ra trải nghiệm không tốt cho người dùng.

+ Nếu không thực hiện đúng cách, việc kiểm thử trên môi trường production có thể tăng chi phí và rủi ro. Lỗi nghiêm trọng có thể dẫn đến mất mát về tài chính và uy tín của doanh nghiệp

TÀI LIỆU THAM KHẢO

Tiếng Việt