TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ HK1 NĂM HỌC 2023 - 2024

BÁO CÁO CUỐI KỲ MÔN
NHẬP MÔN HỌC MÁY

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

Người thực hiện: **DƯƠNG THANH QUÝ – 52000591**

Lớp : **20050401**

Khóa : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ HK1 NĂM HỌC 2022 - 2023

BÁO CÁO CUỐI KỲ MÔN
NHẬP MÔN HỌC MÁY

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

Người thực hiện: **DƯƠNG THANH QUÝ – 52000591**

Lớp : **20050401**

Khóa : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Trước tiên, tôi xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến Giảng viên Lê Anh Cường. Thầy là người đã luôn hỗ trợ và hướng dẫn tận tình cho tôi trong suốt quá trình giảng dạy và là nền tảng để tôi hoàn thành bài tập với đề tài cuối kỳ môn ***"Nhập môn Học máy"***.

Tiếp theo, tôi xin gửi lời cảm ơn đến khoa Công Nghệ Thông Tin trường Đại học Tôn Đức Thắng vì đã tạo điều kiện cho tôi được học tập và nghiên cứu môn học này. Khoa đã luôn sẵn sàng chia sẻ các kiến thức bổ ích cũng như chia sẻ các kinh nghiệm tham khảo tài liệu, giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu mà còn giúp ích cho việc học tập và rèn luyện trong quá trình thực hành tại trường Đại học Tôn Đức Thắng nói chung.

Cuối cùng, sau khoảng thời gian học tập trên lớp, tôi đã hoàn tất bài báo cáo nhờ vào sự hướng dẫn, giúp đỡ và những kiến thức học hỏi được từ Quý thầy cô. Do giới hạn về mặt kiến thức và khả năng lý luận nên tôi vẫn còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của Quý thầy cô giáo để bài báo cáo của tôi được hoàn thiện hơn. Hơn nữa, nhờ những góp ý từ thầy cô và các bạn hữu, tôi sẽ hoàn thành tốt hơn ở những bài nghiên cứu trong tương lai. Tôi mong Quý thầy cô và các bạn bè – những người luôn quan tâm và hỗ trợ chúng tôi – luôn tràn đầy sức khỏe và sự bình an.

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

Tác giả

(Ký tên và ghi rõ họ tên)

ĐỒ ÁN / BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Khóa luận/Đồ án tốt nghiệp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung Khóa luận/Đồ án tốt nghiệp của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

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

Tác giả

(Ký tên và ghi rõ họ tên)

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

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

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

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

(kí và ghi họ tên)

MỤC LỤC

MACHINE LEARNING

DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU

BÀI 1

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

1.1.1 Tìm hiểu về Optimizer

Quá trình tối ưu hóa không chỉ dừng lại ở việc điều chỉnh trọng số và bias trong mạng neural mà còn là một quy trình phức tạp gồm một chuỗi các bước để "hướng dẫn" mô hình trong quá trình "học" từ dữ liệu.

Khi xây dựng mạng neural, chúng ta thường khởi tạo trọng số và bias một cách ngẫu nhiên. Tuy nhiên, để mô hình có thể hiểu và dự đoán chính xác dữ liệu thì cần điều chỉnh các tham số này. Thuật toán tối ưu hóa chính là yếu tố quyết định cách mà chúng ta thay đổi các tham số này.

Thuật toán tối ưu hóa giúp mô hình "học" thông qua việc điều chỉnh trọng số và bias dựa trên dữ liệu. Thay vì thử nghiệm ngẫu nhiên, chúng ta sử dụng các phương pháp thông minh để cải thiện từng tham số qua mỗi bước.

Ví dụ, các thuật toán tối ưu hóa như Gradient Descent và các biến thể của nó (như Stochastic Gradient Descent, Mini-batch Gradient Descent) giúp xác định hướng và độ lớn của điều chỉnh trọng số và bias để mô hình tiến gần đến mức tối ưu. Chúng hoạt động dựa trên việc tính toán gradient của hàm mất mát, từ đó xác định hướng cần di chuyển và độ lớn của bước đi trong không gian các tham số.

Sử dụng thuật toán tối ưu hóa không chỉ giúp mô hình học hiệu quả hơn mà còn tiết kiệm tài nguyên và thời gian. Thay vì lãng phí với việc thử nghiệm ngẫu nhiên, chúng ta sử dụng thông tin từ dữ liệu thực tế để cập nhật mô hình. Điều này giúp mô hình học nhanh hơn và hội tụ đến kết quả tối ưu tốt hơn

Ưu điểm:

Thuật toán Gradient Descent đơn giản và dễ hiểu, là bước đầu tiên quan trọng trong việc hiểu và áp dụng cho mạng neural.

Công cụ mạnh mẽ giúp tối ưu hóa mô hình neural network bằng cách cập nhật trọng số sau mỗi vòng lặp.

Nhược điểm:

Phụ thuộc lớn vào nghiệm khởi tạo ban đầu và learning rate, gây ảnh hưởng lớn đến kết quả cuối cùng.

Vấn đề khi một hàm số có hai điểm global minimum, dẫn đến kết quả khác nhau tùy thuộc vào điểm khởi tạo ban đầu.

Tốc độ học quá lớn hoặc quá nhỏ đều gây ảnh hưởng tiêu cực đến quá trình huấn luyện, khiến thuật toán không hội tụ hoặc làm chậm quá trình này.

1.2.1 Các thuật toán tối ưu

1.2.1.1 Gradient Descent (GD)

Trong thực tế, khi làm các bài toán tối ưu, chúng ta thường tìm kiếm giá trị nhỏ nhất của một hàm số cụ thể. Điều này sẽ phải xem xét đến nhiều yếu tố phức tạp. Đạo hàm của hàm số chính là một khía cạnh then chốt trong việc này. Để tìm ra giá trị nhỏ nhất, chúng ta thường tập trung vào các điểm mà đạo hàm của hàm số đạt giá trị bằng 0. Tuy nhiên, việc tính toán đạo hàm không phải lúc nào cũng đơn giản, đặc biệt là đối với các hàm số có nhiều biến.

Thay vào đó, một phương pháp thay thế phổ biến là sử dụng Gradient Descent, hay "giảm dần độ dốc". Ý tưởng chính là chúng ta chọn một điểm khởi đầu ngẫu nhiên và sau đó điều chỉnh nó dần dần qua mỗi vòng lặp (có thể gọi là epoch) bằng việc di chuyển theo hướng ngược với độ dốc của hàm số, với tốc độ di chuyển được xác định bởi "learning rate". Công thức cụ thể để cập nhật giá trị mới được thể hiện bằng:

xnew = xold - learningrate.gradient(x)

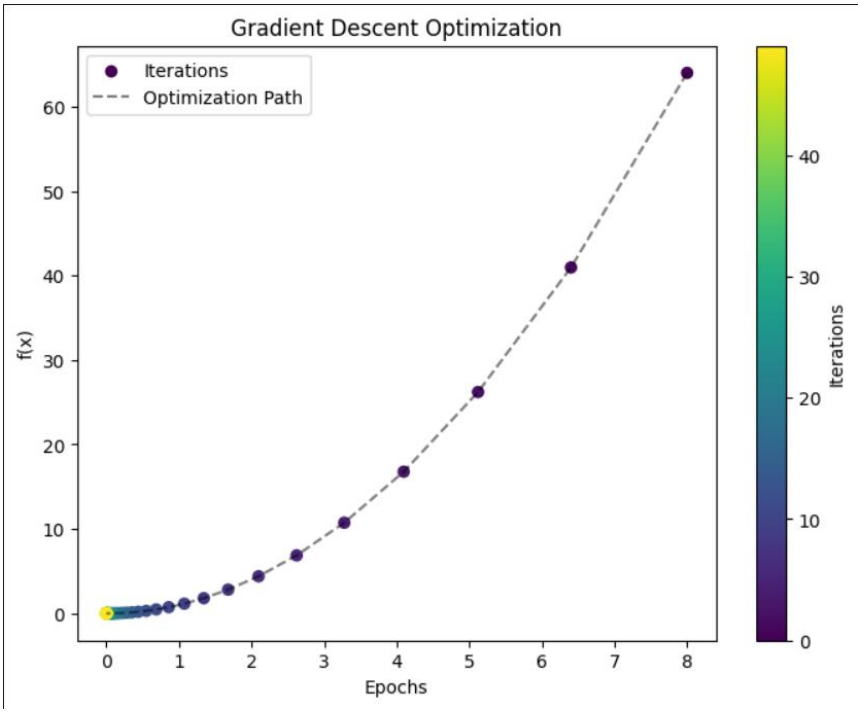
1.2.1.1.1 Gradient cho hàm 1 biến

Trong việc phân tích Gradient descent, chúng ta nhận ra rằng quá trình này phụ thuộc vào một loạt yếu tố phức tạp. Điều này bao gồm việc chọn điểm x ban đầu, một lựa chọn đơn giản nhưng ảnh hưởng đáng kể đến quá trình hội tụ. Thậm chí, việc điều chỉnh tốc độ học (learning rate) cũng có thể tạo ra những biến đổi không ngờ trong quá trình này.

Khi tốc độ học quá nhỏ, chúng ta rơi vào tình trạng hội tụ chậm, làm chậm quá trình training và kéo dài thời gian cần thiết để đạt được mục tiêu. Ngược lại, với tốc độ học quá lớn, chúng ta có thể đi nhanh tới đích ngắn sau vài vòng lặp. Tuy nhiên, điều này cũng đồng nghĩa với việc thuật toán không hội tụ mà đi lung tung quanh mục tiêu do bước nhảy quá lớn.

Điều này chỉ là một phần nhỏ trong việc hiểu rõ cách Gradient descent hoạt động. Sự phức tạp và đa dạng của các yếu tố này làm cho quá trình tối ưu hóa hàm mất mát

trở nên thú vị và phức tạp hơn, đồng thời cũng làm tăng sự khó khăn khi điều chỉnh các tham số để đạt được hiệu suất tối ưu.



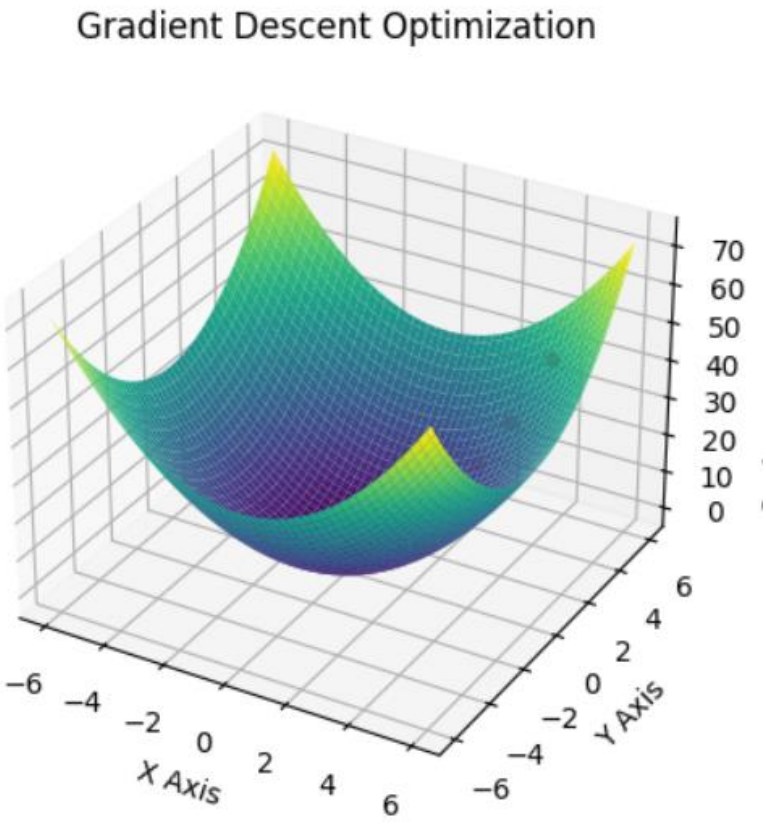
Hình 1: Gradient cho hàm 1 biến

1.2.1.1.1 Các thông tin ảnh hưởng trực tiếp Gradient descent cho hàm nhiều biến

Tại mỗi bước di chuyển, thuật toán Gradient Descent cho hàm nhiều biến phải đối mặt với việc tính toán gradient theo từng chiều của không gian đa chiều. Mỗi chiều này đại diện cho một hướng tối ưu hoá riêng, một cách như đang phải duy trì sự cân đối giữa các biến thể khác nhau. Điều này tạo ra một độ phức tạp và đa dạng trong cách thức điều chỉnh tham số để giảm thiểu hàm mất mát.

Tốc độ học, một yếu tố chủ chốt trong thuật toán Gradient Descent, vẫn đóng vai trò quan trọng. Quá nhỏ, nó làm cho thuật toán di chuyển rất chậm và có thể mắc kẹt tại các điểm cực tiểu cục bộ; quá lớn, nó khiến thuật toán dao động hoặc không hội tụ.

Điều này làm tăng thêm sự phức tạp và đa dạng trong việc điều chỉnh các tham số trong không gian đa chiều, đặt ra thách thức lớn trong việc tối ưu hóa hàm mất mát. Các chiều đa dạng này tạo ra một cảnh khó khăn hơn trong việc điều chỉnh các thông số để đạt được hiệu suất tối ưu, tương tự như việc tối ưu hóa hàm một biến nhưng với sự phức tạp và biến động đáng kể hơn.



Hình 2: Gradient cho hàm nhiều biến

1.2.1.2 Stochastic Gradient Descent (SGD)

Thuật toán SGD không chỉ cập nhật trọng số (Weight) sau mỗi epoch như trong Gradient Descent, mà trong mỗi epoch, chúng ta sẽ cập nhật trọng số N lần, tương ứng với N điểm dữ liệu. Điều này mang lại một mặt của SGD làm giảm tốc độ của mỗi epoch, nhưng mặt khác, nó cũng đồng thời hội tụ rất nhanh chỉ sau vài epoch.

Công thức của SGD tương tự như GD, nhưng việc thực hiện trên từng điểm dữ liệu tạo ra một đường đi zig-zag. Điều này dễ hiểu vì một điểm dữ liệu không thể đại diện cho toàn bộ dữ liệu.

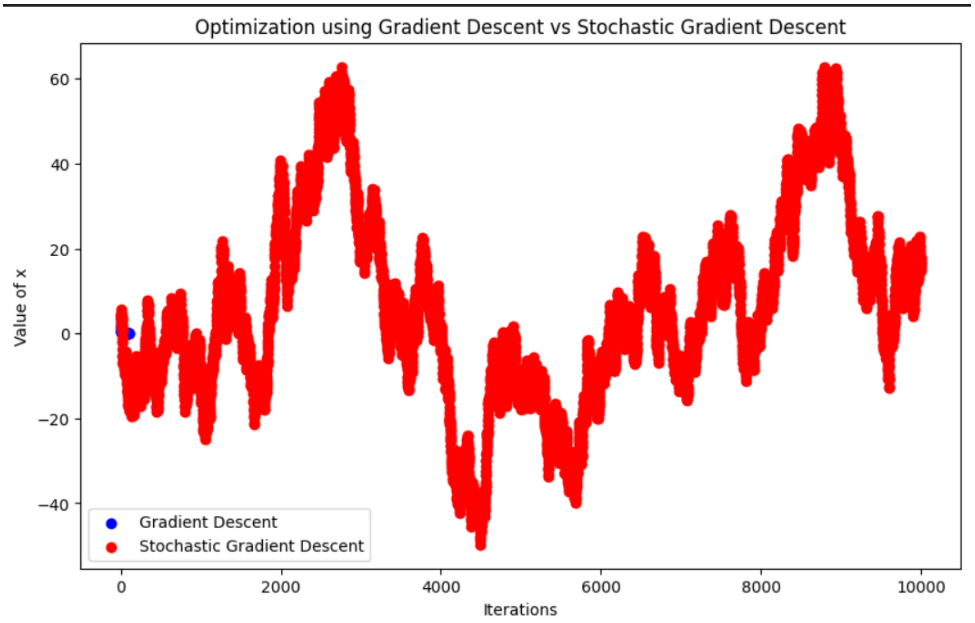
GD gặp hạn chế khi áp dụng vào cơ sở dữ liệu lớn, ví dụ như vài triệu dữ liệu, do việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên rất phức tạp và tốn kém. Ngoài ra, GD không phù hợp với việc học trực tuyến (online learning).

SGD ra đời để hạn chế vấn đề cập nhật lại toàn bộ dữ liệu, thay vào đó chỉ cần cập nhật trên từng điểm dữ liệu mới thêm vào, phù hợp với việc học trực tuyến.

Ví dụ minh họa rõ ràng cho điều này: Giả sử bạn đang xây dựng một mô hình dự đoán giá nhà từ một tập dữ liệu lớn gồm 50.000 bản ghi. Khi sử dụng Stochastic Gradient Descent (SGD) để huấn luyện mô hình, sau chỉ 5 epoch, mô hình của bạn đã có khả

năng dự đoán giá nhà với mức độ chính xác khá cao.

Trong khi đó, nếu bạn áp dụng Gradient Descent (GD) cho cùng tập dữ liệu 50.000 bản ghi, để đạt được kết quả tương tự, mô hình có thể cần tới 50 hoặc 60 epoch. Sự khác biệt rõ ràng ở đây là với SGD, mô hình đạt được mức độ chính xác mong muốn sau ít epoch hơn so với GD, thể hiện sự nhanh nhẹn và hiệu quả trong quá trình huấn luyện.



Hình 3: Biểu đồ thể hiện Gradient Descent

1.2.1.3 Momentum

Ta có: $x_{new} = x_{old} - \gamma . \vartheta + learning\ rate\ . gradient$

Ở đây:

x_{new} là tọa độ mới trong không gian tối ưu.

x_{old} là tọa độ cũ, vị trí trước đó.

γ là tham số momentum, thường là 0.9, đại diện cho sức mạnh của đà.

learning rate là tốc độ học, quyết định tốc độ di chuyển.

gradient là đạo hàm của hàm mục tiêu f, chỉ ra hướng và độ lớn của thay đổi.

Momentum không chỉ dừng lại ở việc di chuyển từ điểm này đến điểm khác. Nó là một "tinh thần" của quá trình tối ưu hóa, khiến cho bước nhảy không chỉ mạnh mẽ mà còn linh hoạt. Nó không chỉ giúp vượt qua các chướng ngại vật một cách mạnh mẽ mà còn làm tăng tính đa dạng và sự phức tạp trong cách mà thuật toán di chuyển và tiến hành tối ưu hóa. Momentum không chỉ đơn thuần là một giá trị trong phương trình mà

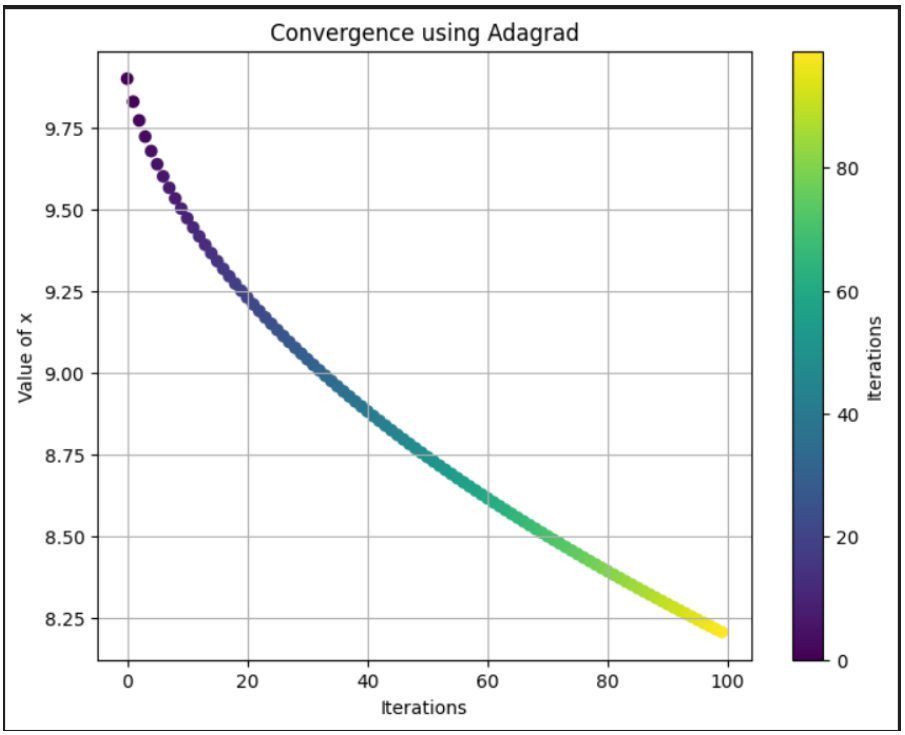
còn là nguồn cảm hứng, làm nảy sinh ra các hành động đa dạng và phức tạp hơn, tạo ra sự sôi động và đột phá cho quá trình tối ưu hóa.

1.2.1.4 Adagrad

Adagrad, một thuật toán đặc biệt trong học máy, nổi bật với sự khác biệt rõ ràng so với các phương pháp trước đây về việc điều chỉnh learning rate trong quá trình huấn luyện mô hình. Trong những thuật toán trước đó, learning rate thường được xem như một hằng số, duy trì ổn định suốt quá trình học (learning rate không thay đổi). Tuy nhiên, Adagrad lại tiếp cận khác, coi learning rate như một tham số động.

Điều này có nghĩa là Adagrad thay đổi learning rate sau mỗi bước thời gian t trong quá trình huấn luyện. Cụ thể, với các tham số như n, gt (gradient tại thời điểm t), và ϵ (hệ số tránh lỗi để tránh việc chia cho mẫu bằng 0), Adagrad sử dụng ma trận chéo G. Mỗi phần tử trên đường chéo chính của ma trận này (i, i) tương ứng với bình phương của đạo hàm vectơ tham số tại thời điểm t.

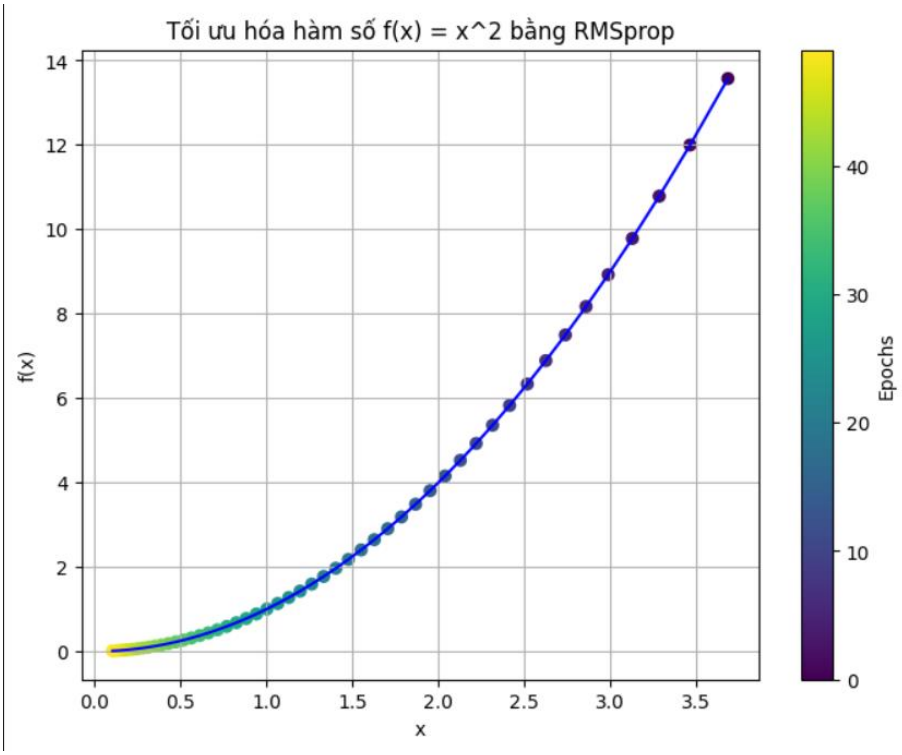
Điều này tạo ra một sự biến đổi đáng kể trong việc điều chỉnh learning rate dựa trên đạo hàm của các tham số, tăng tính linh hoạt và tối ưu hóa trong quá trình huấn luyện mô hình. Sự khác biệt này trong cách tiếp cận learning rate đã đem lại một cái nhìn mới và hiệu quả hơn về việc điều chỉnh mức độ học của mô hình trong quá trình tối ưu hóa.



Hình 4: Biểu đồ thể hiện Adagrad

1.2.1.5 RMSprop

RMSprop, một trong những thuật toán quan trọng trong học sâu, nổi bật với cách tiếp cận độc đáo giải quyết vấn đề của việc tỷ lệ học giảm dần, một hạn chế của Adagrad. Thay vì chỉ dựa vào việc chia tỷ lệ học theo bình phương gradient, RMSprop sử dụng trung bình của bình phương gradient, tạo ra một cách tiếp cận hiệu quả hơn và linh hoạt hơn trong việc điều chỉnh tỷ lệ học trong quá trình huấn luyện mô hình. Điều này tạo ra sự đa dạng và tính phức tạp trong cách thuật toán xử lý thông tin gradient, đồng thời tối ưu hóa quá trình học của mô hình một cách hiệu quả và ổn định.



Hình 5: Biểu đồ thể hiện RMSprop

1.2.1.6 Adam

Adam, như đã được đề cập trước đó, không chỉ là sự kết hợp giữa Momentum và RMSprop, mà nó còn là một hiện tượng vật lí đầy phức tạp. Nếu chúng ta nghĩ về Momentum như là một quả cầu lao xuống dốc, thì Adam lại là một quả cầu nặng có ma sát vô cùng đặc biệt. Điều này khiến cho Adam trở nên linh hoạt hơn trong việc vượt qua local minimum và đi đến global minimum. Mà khi nó đạt được global minimum, nó không chỉ dừng lại một cách nhanh chóng mà còn duy trì ổn định hơn, giống như việc quả cầu có ma sát không bao giờ mất thời gian lãng phí trong việc dao động không cần thiết xung quanh mục tiêu cuối cùng. Điều này tạo nên một đặc

điểm độc đáo và phức tạp cho Adam, tạo ra một sự khác biệt đáng kinh ngạc trong quá trình tối ưu hóa.

1.2.1 So sánh các thuật toán tối ưu

Phương Pháp	Mô Tả	Ưu Điểm	Nhược Điểm
Gradient Descent (GD)	Phương pháp cổ điển cập nhật trọng số theo đạo hàm của hàm mất mát, điều chỉnh trọng số theo hướng ngược lại với gradient.	- Hội tụ ổn định khi hàm mất mát là convex. - Dễ hiểu và triển khai.	- Tốn thời gian với tập dữ liệu lớn. - Dễ rơi vào local minimum khi hàm mất mát không phải là convex.
Stochastic Gradient Descent (SGD)	Cập nhật trọng số sau mỗi lượt duyệt qua từng mẫu dữ liệu ngẫu nhiên, giảm thiểu tính toán.	- Hiệu quả với dữ liệu lớn. - Có thể thoát khỏi local minimum do sự ngẫu nhiên trong việc cập nhật.	- Không ổn định trong việc hội tụ, dao động quanh điểm tối ưu.
Momentum	Kết hợp gradient hiện tại với đà từ các bước trước đó để cập nhật trọng số, giúp tăng tốc độ hội tụ và tránh được dao động.	- Giảm đáng kể dao động và tăng tốc độ hội tụ. - Hiệu quả với các bề mặt hàm mất mát không đều.	- Cần điều chỉnh siêu tham số đà để đạt hiệu suất tốt nhất.
Adagrad	Điều chỉnh learning rate cho mỗi tham số theo tần suất xuất hiện của nó, giúp thích nghi với từng tham số riêng biệt.	- Hiệu quả với các tham số thưa (sparse) và dữ liệu không đồng nhất. - Không cần điều chỉnh learning rate.	- Có thể gây ra vấn đề của learning rate giảm dần quá nhanh, dẫn đến việc ngừng học sớm hoặc độ chính xác không cải thiện sau một thời gian.

RMSprop	Cải tiến của Adagrad, giảm vấn đề của learning rate giảm quá nhanh bằng cách duy trì một cửa sổ trượt của các gradient vuông của gradient.	- Hiệu quả hơn so với Adagrad với các vấn đề về learning rate.	- Vẫn cần điều chỉnh siêu tham số.
Adam	Kết hợp Momentum và RMSprop, cung cấp cân bằng giữa việc theo dõi gradient và đà từ các bước trước đó. Sử dụng cả moment của gradient và cửa sổ trượt của gradient vuông của gradient.	- Hiệu suất cao và tự điều chỉnh learning rate. - Thích hợp cho đa dạng các bề mặt hàm mất mát.	- Cần điều chỉnh các siêu tham số. - Có thể không hiệu quả trên một số vấn đề cụ thể.

Bảng 1: So sánh các thuật toán

BÀI 2

2 Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy

2.1.1 Continual Learning

Hai phương pháp chính: Stateless Retraining và Stateful Training.

Stateless Retraining đề cập đến việc huấn luyện mô hình từ đầu. Điều này đòi hỏi rất nhiều dữ liệu để mô hình có thể hiểu và thích nghi với các biến thể mới của thông tin. Ví dụ, khi áp dụng phương pháp này, mỗi lần một phiên bản mới của mô hình được tạo ra, nó sẽ được huấn luyện từ đầu, từ kiến trúc ban đầu đến các phiên bản tiếp theo.

Stateful Training tập trung vào việc mô hình tiếp tục được huấn luyện dựa trên dữ liệu mới được cung cấp. Quá trình này thường được gọi là fine-tuning hoặc incremental learning. Thay vì huấn luyện lại toàn bộ mô hình, nó tập trung vào việc cập nhật từng phần nhỏ một của mô hình dựa trên dữ liệu mới, giúp tiết kiệm dữ liệu hơn.

Còn về việc cập nhật mô hình, có ba loại cập nhật chính: Iteration trên mô hình, Iteration trên dữ liệu và Truyền thông tin kiến thức.

- Iteration trên mô hình thường liên quan đến việc thêm tính năng mới vào kiến trúc hiện tại hoặc thay đổi kiến trúc của mô hình để có thể áp dụng Stateful Training.
- Truyền thông tin kiến thức thì liên quan đến việc chuyển giao kiến thức từ một mô hình hoặc một tác vụ sang một mô hình hoặc tác vụ khác.
- Iteration trên dữ liệu tập trung vào việc cập nhật dữ liệu mới vào mô hình hiện tại mà không thay đổi kiến trúc hay tính năng của nó. Điều này đặc biệt hữu ích trong việc áp dụng Stateful Training.

Một trong những trường hợp sử dụng phổ biến của Continual Learning là đối phó với sự thay đổi đột ngột trong phân phối dữ liệu, đặc biệt là khi các thay đổi diễn ra một cách bất ngờ. Đây cũng là cách để thích nghi với các sự kiện hiếm gặp mà mô hình trước đây chưa từng gặp phải.

Để hiểu rõ hơn về cách cập nhật mô hình, có bốn giai đoạn của Continual Learning:

Giai đoạn 1: Đào tạo lại thủ công, không lưu trạng thái trước.

Giai đoạn 2: Đào tạo lại tự động.

Giai đoạn 3: Đào tạo lại tự động, lưu trạng thái.

Giai đoạn 4: Continual Learning.

Một khía cạnh quan trọng khác là sự lặp lại giữa việc cải tiến mô hình và dữ liệu. Tuy cải tiến mô hình có thể đem lại lợi ích đáng kể, nhưng tìm kiếm kiến trúc mô hình tốt hơn đòi hỏi tài nguyên tính toán lớn, trong khi việc tối ưu hóa dữ liệu có thể đạt được hiệu quả cao hơn với tài nguyên ít hơn.

Việc triển khai và kiểm thử mô hình trong môi trường sản xuất đóng vai trò quan trọng trong việc đảm bảo hiệu suất và độ tin cậy của mô hình. Các phương pháp như Shadow Deployment, A/B Testing, Canary Release, Interleaving Experiments và Bandits đã được đề xuất để hỗ trợ quá trình triển khai mô hình một cách an toàn và hiệu quả.

2.2.1 Test in Production

Các Phương Pháp Triển Khai và Kiểm Thử Mô Hình:

Phương pháp	Mô tả
Shadow Deployment	Phương pháp này đề xuất triển khai mô hình ứng cử viên song song với mô hình hiện tại. Dữ liệu từ mọi request được đưa qua cả hai mô hình, nhưng chỉ kết quả từ mô hình hiện tại được trả về cho người dùng. Việc lưu trữ dữ liệu dự đoán từ mô hình mới cho phép phân tích và đánh giá hiệu suất trước khi thay thế mô hình hiện tại. Tuy nhiên, chi phí cho việc xử lý dữ liệu gấp đôi là một hạn chế của phương pháp này.
A/B Testing	Phương pháp này nhằm so sánh hiệu suất giữa hai biến thể của một mô hình thông qua việc chia traffic giữa chúng. Đánh giá hiệu suất dựa trên dữ liệu thu thập từ các mô hình và phản hồi từ người dùng. Đây là một thí nghiệm ngẫu nhiên đòi hỏi một mẫu số đủ lớn để có độ tin cậy về kết quả.
Canary Release	Phương pháp này giúp giảm rủi ro khi triển khai phiên bản mới bằng cách tung ra thay đổi một cách từ từ cho một phần nhỏ người dùng trước khi triển khai cho tất cả. Việc theo dõi hiệu suất của phiên bản thử nghiệm và chỉ tăng lượng traffic đến phiên bản mới khi hiệu suất đạt yêu cầu, giúp tránh được các rủi ro tiềm ẩn.

Interleaving Experiments	Phương pháp này nhằm xác định thuật toán tốt nhất với mẫu số nhỏ hơn so với A/B testing thông thường bằng cách đo lường sự ưa thích của người dùng thay vì so sánh các chỉ số cốt lõi.
Bandits	Phương pháp này sử dụng thông tin lịch sử để quyết định cách chia traffic thông minh hơn so với việc chia ngẫu nhiên như trong A/B testing. Bandits giúp tiết kiệm dữ liệu và giảm thiểu chi phí cơ hội trong quá trình kiểm thử.

Bảng 2: Các phương pháp Test trong Production

TÀI LIỆU THAM KHẢO

[1] https://machinelearningcoban.com/search/

[2] https://aiden-jeon.github.io/post/mlops/designing-ml-system/chapter-00/