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

ĐẶNG THỊ MINH THƯ - 52100843

YÊU CẦU 1

DỰ ÁN 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

ĐẶNG THỊ MINH THƯ - 52100843

YÊU CẦU 1

DỰ ÁN CUỐI KỲ
NHẬP MÔN HỌC MÁY

Người hướng dẫn
PGS.TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin chân thành cảm ơn Khoa Công nghệ thông tin, Trường đại học Tôn Đức Thắng đã tạo điều kiện học tập thuận lợi để em có thể hoàn thành bài tiểu luận này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Lê Anh Cường đã truyền đạt kiến thức, những kinh nghiệm quý giá và hướng dẫn em trong quá trình làm bài.

Em đã cố gắng vận dụng những kiến thức đã học trong học kỳ vừa qua để hoàn thành bài báo cáo. Tuy nhiên, những kiến thức và kỹ năng về môn học này vẫn còn nhiều hạn chế do đó bài báo cáo của em khó tránh khỏi những sai sót. Kính mong quý thầy/ cô xem xét và góp ý giúp bài báo cáo của em được hoàn thiện hơn

TP. Hồ Chí Minh, ngày 23 Tháng 12 năm 2024.

Tác giả

(Ký tên và ghi rõ họ tên)

Thư

Đặng Thị Minh Thư

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS.TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án 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 23 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Thư

Đặng Thị Minh Thư

TÊN ĐỀ TÀI

TÓM TẮT

Bài báo cáo trình bày yêu cầu một của bài dự án cuối kỳ môn Nhập môn học máy. Trong phần này trình bày về những tìm hiểu, nghiên cứu của em về các phương pháp optimizer trong huấn luyện mô hình học máy và tìm hiểu về Continual Learning và Test Production khi xây dựng giải pháp học máy.

Bài báo cáo gồm hai phần chính:

Phần 1: Tìm hiểu về các phương pháp tối ưu trong huấn luyện mô hình học máy

Hiện nay có nhiều phương pháp tối ưu hóa, tuy nhiên trong phần này em sẽ tìm hiểu và so sánh một số phương pháp tối ưu hóa phổ biến nhất, bao gồm:

Gradient Descent, Stochastic Gradient Descent, Momentum

AdaGrad, Adadelta, RMSProp, Adam

Phần 2: Tìm hiểu về Continual learning và Test product

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

GD	Gradient Descent
SGD	Stochastic Gradient Descent
CL	Continual Learning
ML	Machine Learning

CHƯƠNG 1. TÌM HIỂU CÁC PHƯƠNG PHÁP TỐI ƯU TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 Giới thiệu về Optimizer

Optimizer (hay trình tối ưu hóa) là các thuật toán điều chỉnh các tham số của mô hình trong quá trình đào tạo (training) để giảm thiểu hàm mất mát (loss function). Chúng cho phép mạng lưới thần kinh (neural network) học hỏi từ dữ liệu bằng cách cập nhật lặp đi lặp lại các trọng số (weight) và độ lệch (bias). Về cơ bản, thuật toán tối ưu là cơ sở để xây dựng mô hình neural network với mục đích "học" được các features (hay pattern) của dữ liệu đầu vào, từ đó có thể tìm 1 cặp weights và bias phù hợp để tối ưu hóa model.

Thuật toán tối ưu hóa là phương pháp tối ưu hóa giúp cải thiện hiệu suất của mô hình. Các thuật toán tối ưu hóa hoặc trình tối ưu hóa này ảnh hưởng rộng rãi đến độ chính xác và tốc độ đào tạo của mô hình học.

Mỗi thuật toán tối ưu hóa có các quy tắc cập nhật, tốc độ học tập và động lượng cụ thể để tìm các tham số mô hình tối ưu nhằm cải thiện hiệu suất.

Có hai số liệu chính cần xem xét khi xác định hiệu quả của trình tối ưu hóa:

- Tốc độ hội tụ, nghĩa là tốc độ đạt được cực tiểu của hàm mất mát.
- Khái quát hóa mô hình, nghĩa là mô hình hoạt động tốt như thế nào trên dữ liệu mới chưa được nhìn thấy.

1.2 Các phương pháp Optimizer phổ biến

1.2.1 Gradient Descent

Gradient Descent là thuật toán tối ưu hóa cơ bản nhất nhưng được sử dụng nhiều nhất. Nó được sử dụng rất nhiều trong các thuật toán phân loại (classification) và hồi quy tuyến tính (linear regression). Lan truyền ngược (backpropagation) trong mạng nơ-ron cũng sử dụng thuật toán gradient descent.

Gradient Descent (tiếng việt là Giảm dần độ dốc) là thuật toán tối ưu hóa phụ thuộc vào đạo hàm bậc nhất của hàm mất mát (loss function). Nó tính toán xem nên thay đổi trọng số (weight) theo cách nào để hàm có thể đạt cực tiểu (minima).

Cách thức hoạt động của Gradient Descent như sau:

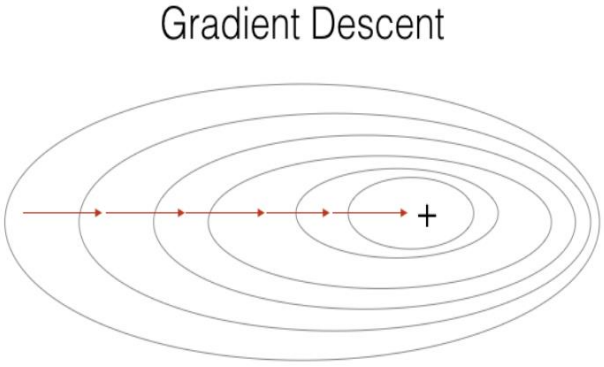
1. Tính toán gradient: Gradient Descent bắt đầu bằng cách tính toán gradient của hàm mất mát theo từng tham số của mô hình. Gradient cho biết hướng và độ lớn của cách thay đổi các tham số để giảm thiểu hàm mất mát.
2. Cập nhật tham số: Dựa trên gradient, Gradient Descent sẽ cập nhật các tham số của mô hình theo một bước nhất định. Bước này thường được gọi là "gradient descent" (giảm dần độ dốc). Kích thước của bước phụ thuộc vào một giá trị gọi là learning rate.
3. Lặp lại: Quá trình tính toán gradient và cập nhật tham số được lặp lại nhiều lần (được gọi là "epochs") cho đến khi hàm mất mát đạt giá trị tối thiểu hoặc không còn giảm đáng kể nữa.

Công thức cập nhật tham số của Gradient Descent:

$$\theta_{new} = \theta_{old} - \alpha . \nabla \theta J(\theta_{old})$$

Trong đó:

- θ_{new} là tham số mới
- θ_{old} là tham số cũ
- α là learning rate
- $\nabla \theta J(\theta_{old})$ là gradient của hàm mất mát theo tham số θ_{old}



Hình 1.1 Minh họa đường đi thuật toán Gradient Descent

Gradient descent phụ thuộc vào nhiều yếu tố : như nếu chọn điểm x ban đầu khác nhau sẽ ảnh hưởng đến quá trình hội tụ; hoặc tốc độ học (learning rate) quá lớn hoặc quá nhỏ cũng ảnh hưởng: nếu tốc độ học quá nhỏ thì tốc độ hội tụ rất chậm ảnh hưởng đến quá trình training, còn tốc độ học quá lớn thì tiến nhanh tới đích sau vài vòng lặp tuy nhiên thuật toán không hội tụ, quanh quẩn quanh đích vì bước nhảy quá lớn.

Ưu điểm:

- Tính toán dễ dàng.
- Dễ để thực hiện.
- Dễ hiểu.

Nhược điểm:

- Vì đơn giản nên thuật toán Gradient Descent còn nhiều hạn chế như phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.
- Trọng số được thay đổi sau khi tính toán độ dốc trên toàn bộ tập dữ liệu. Vì vậy, nó có thể hội tụ chậm, đặc biệt vối các mô hình phức tạp hoặc tập dữ liệu lớn, mất nhiều thời gian để hội tụ về mức tối thiểu
- Yêu cầu bộ nhớ lớn để tính toán độ dốc trên toàn bộ tập dữ liệu.

1.2.2 Stochastic Gradient Descent

SGD (tiếng việt là Giảm dần độ dốc ngẫu nhiên) là một biến thể của phương pháp Gradient Descent, thực hiện cập nhật các tham số mô hình (weight) dựa trên độ dốc của hàm mất mát được tính toán trên một tập hợp con được chọn ngẫu nhiên của dữ liệu huấn luyện, thay vì trên toàn bộ tập dữ liệu.

Trong Gradient Descent thì sau mỗi epoch, chúng ta sẽ cập nhật trọng số (weight) một lần. Còn trong SGD nếu mỗi epoch có N điểm dữ liệu thì ta sẽ cập nhật trọng số N lần.

Cách thức hoạt động của SGD như sau:

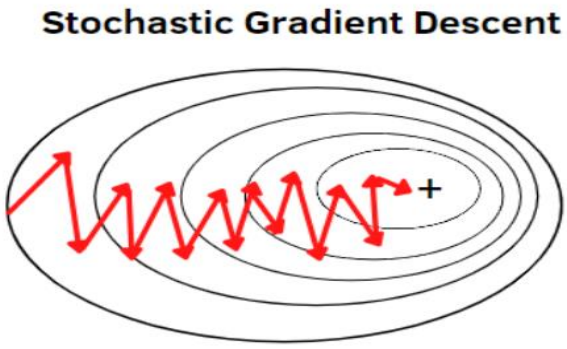
1. Tạo một mẫu dữ liệu ngẫu nhiên của dữ liệu huấn luyện (gọi là mini-batch)
2. Tính toán gradient của hàm mất mát theo mẫu dữ liệu ngẫu nhiên
3. Cập nhật các tham số của mô hình theo gradient đã tính toán

4. Lặp lại bước 1-3 cho đến khi thuật toán hội tụ hoặc đạt đến tiêu chí dừng được xác định trước.

Công thức cập nhật tham số của SGD:

$$\theta_{new} = \theta_{old} - \alpha.\nabla\theta J(x_i, \theta_{old})$$

Trong đó: x_i là mẫu dữ liệu ngẫu nhiên



Hình 1.2 Hình minh họa đường đi thuật toán Stochastic Gradient Descent

SGD sẽ làm giảm đi tốc độ của 1 epoch. Tuy nhiên nhìn theo 1 hướng khác, SGD sẽ hội tụ rất nhanh chỉ sau vài epoch.

So sánh đường đi của GD và SGD ta thấy, SGD có nhiều đường zig zắc hơn. Bởi vì, một điểm dữ liệu trong SGD chỉ là ngẫu nhiên và không thể đại diện cho toàn bộ tập dữ liệu.

Ưu điểm:

- Việc cập nhật thường xuyên các tham số giúp mô hình hội tụ trong thời gian ngắn hơn.
- Yêu cầu ít bộ nhớ hơn vì không cần lưu trữ giá trị của loss function
- Thuật toán giải quyết được đối với cơ sở dữ liệu lớn mà GD không làm được. SGD trong học sâu là thích hợp hơn nếu dữ liệu lớn và thời gian tính toán lớn.

Nhược điểm:

- Phương sai cao trong các tham số mô hình.

- Thay vì toàn bộ tập dữ liệu, ta chỉ lấy các tập nhỏ của tập dữ liệu để lặp lại. Vì vậy, đường đi được thuật toán chấp nhận bao gồm nhiễu. Do đó, SGD sử dụng số lần lặp cao hơn để đạt được cực tiểu cục bộ.
- Thuật toán vẫn chưa giải quyết được 2 nhược điểm lớn của gradient descent (learning rate, điểm dữ liệu ban đầu). Vì vậy ta phải kết hợp SGD với 1 số thuật toán khác như: Momentum, AdaGrad,..

1.2.3 Momentum

Momentum được phát minh để giảm phương sai cao trong SGD và làm giảm sự hội tụ. Nó tăng tốc độ hội tụ theo hướng có liên quan và giảm sự biến động theo hướng không liên quan.

Thuật ngữ momentum (động lượng) có thể được hiểu là vận tốc của trình tối ưu hóa. Trình tối ưu hóa tích lũy động lượng khi nó di chuyển xuống dốc và giúp làm giảm dao động trong quá trình tối ưu hóa. Điều này có thể giúp trình tối ưu hóa hội tụ nhanh hơn và đạt mức tối thiểu cục bộ tốt hơn.

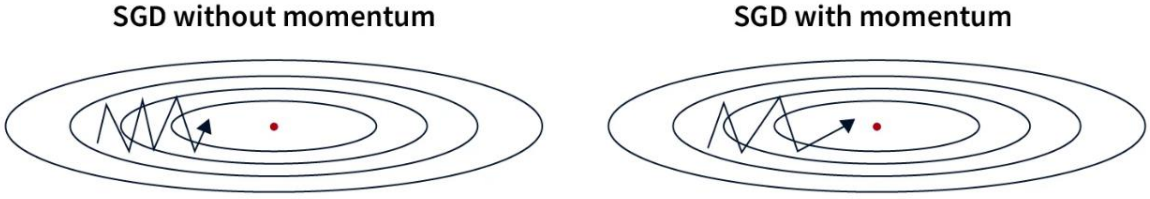
Một siêu tham số nữa được sử dụng trong phương pháp này được gọi là động lượng được ký hiệu bằng μ

$$v_{new} = \mu . v_{old} + \alpha . \nabla J(\theta)$$

$$\theta_{new} = \theta_{old} - v_{new}$$

Trong đó:

- v_{old} : momentum tại thời điểm t-1
- v_{new} : momentum tại thời điểm t
- μ: hệ số momentum (giá trị thường chọn là 0.9 hoặc giá trị tương đương)



Hình 1.3 Hình so sánh khi không có và có momentum

Từ hình mình họa trên, ta thấy được ở SGD không dùng momentum thì hội tụ nhưng không phải tại điểm global minimum, còn ở SGD với momentum phương trình hội tụ tại điểm global minimum.

Tối ưu hóa momentum đặc biệt hữu ích trong các tình huống trong đó bối cảnh tối ưu hóa nhiễu hoặc khi độ dốc thay đổi nhanh chóng. Nó cũng có thể giúp quá trình tối ưu hóa diễn ra suôn sẻ và ngăn trình tối ưu hóa bị kẹt ở mức tối thiểu cục bộ.

Ưu điểm:

- 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.
- Nó có thể hoạt động tốt với dữ liệu thưa thớt.
- Tự động điều chỉnh tốc độ học tập dựa trên các cập nhật tham số.

Nhược điểm:

- Có thể hội tụ quá chậm đối với một số vấn đề.
- Có thể ngừng học hoàn toàn nếu learning rate trở nên quá nhỏ.

Bảng 1.1: Bảng so sánh Gradient Descent, Stochastic Gradient Descent, Momentum

Đặc điểm	**Gradient Descent**	**Stochastic Gradient Descent**	**Momentum Optimizer**
Cách tính đạo hàm	Tính đạo hàm của hàm mất mát tại tất cả các điểm dữ liệu trong tập dữ liệu	Tính đạo hàm của hàm mất mát tại một điểm dữ liệu ngẫu nhiên trong tập dữ liệu	Tính đạo hàm của hàm mất mát tại một điểm dữ liệu ngẫu nhiên trong tập dữ liệu và tích lũy các cập nhật trong quá trình huấn luyện
Ưu điểm	Đơn giản, dễ hiểu, hiệu quả	Hiệu quả, có thể áp dụng cho các tập dữ liệu lớn	Tăng tốc độ hội tụ, ít bị mắc kẹt trong các điểm tối thiểu cục bộ
Nhược điểm	Có thể bị mắc kẹt trong các điểm tối thiểu cục bộ	Có thể bị nhiễu bởi các điểm dữ liệu ngẫu nhiên	Khó điều chỉnh các tham số
Ứng dụng	Phổ biến trong nhiều lĩnh vực, bao gồm học máy, thị giác máy tính, xử lý ngôn ngữ tự nhiên	Phổ biến trong học máy, đặc biệt là trong các tập dữ liệu lớn	Phổ biến trong học máy, đặc biệt là trong các tập dữ liệu lớn và phức tạp

1.2.4 AdaGrad

Thuật toán Adagrad (Adaptive Gradient Algorithm) điều chỉnh tốc độ học (learning rate) của từng tham số của mạng nơ-ron một cách thích ứng trong quá trình huấn luyện. Cụ thể, nó chia tỷ lệ tốc độ học của từng tham số dựa trên độ dốc (gradient) lịch sử được tính toán cho tham số đó. Nói cách khác, các tham số có độ dốc lớn sẽ có tốc độ học nhỏ hơn, trong khi các tham số có độ dốc nhỏ sẽ có tốc độ học lớn hơn. Điều này giúp ngăn tốc độ học giảm quá nhanh đối với các tham số thường xuyên xảy ra và cho phép hội tụ quá trình đào tạo nhanh hơn.

Các bước của thuật toán như sau:

1. Khởi tạo. Khởi tạo một tập hợp các tham số mô hình θ ngẫu nhiên và một hằng số nhỏ ϵ để tránh chia cho 0
2. Tính toán gradient. Tính toán gradient của hàm mất mát (J) theo từng tham số mô hình ∇J(θ) là đạo hàm của loss function đối với các tham số đã cho tại thời điểm t cho trước

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

3. Cập nhật tham số. Cập nhật từng tham số mô hình

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii} + \epsilon}} . g_{t,i}$$

Trong đó:

α : learning rate đã được sửa đổi với tham số đã cho

4. Lặp lại. Lặp lại các bước 2 và 3 cho đến khi hàm mất mát đạt giá trị tối thiểu mong muốn hoặc không còn thay đổi đáng kể.

Ưu điểm

- AdaGrad điều chỉnh tốc độ học cho từng tham số để cho phép cập nhật Không cần điều chỉnh thủ công tốc độ học tập.
- Hữu ích để xử lý dữ liệu thưa thớt, trong đó một số tính năng đầu vào có tần suất thấp hoặc bị thiếu.

Nhược điểm:

- Tính toán tốn kém vì cần phải tính đạo hàm bậc hai.
- Tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm learning rate cực kì nhỏ, làm việc training trở nên đóng băng.

AdaGrad có một số nhược điểm, do đó nhiều sửa đổi và cải tiến khác nhau đã được đề xuất.

- RMSprop: Để giữ tốc độ học không giảm quá thấp, biểu mẫu này thêm số hạng phân rã vào gradient bình phương.
- AdaDelta: Thay vì sử dụng tổng toàn cục, nó giải quyết vấn đề tốc độ học tập giảm sút một cách đơn điệu của AdaGrad bằng cách sử dụng mức trung bình giảm dần của các gradient bình phương trước đó.
- Adam: Để quản lý quá trình cập nhật hiệu quả hơn, phần mở rộng của RMSprop kết hợp các thuật ngữ động lượng.

1.2.5 AdaDelta

Adadelta sinh ra để làm giảm nhược điểm của Adagrad (việc làm thay đổi learning rate theo tính đơn điệu giảm). Nó giới hạn sự tích lũy của độ biến thiên tới một giới hạn nhất định.

Để làm được điều trên nó đưa ra khái niệm, running average phụ thuộc vào trung bình trước và độ dốc hiện tại.

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

γ : xấp xỉ 0.9

Nó còn đưa ra moment thứ 2 là bình phương của tham số update

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2$$

Tóm tắt công thức

$$g_{t+1} = \gamma g_t + (1-\gamma)\nabla \mathcal{L}(\theta)^2$$

$$x_{t+1} = \gamma x_t + (1-\gamma)v_{t+1}^2$$

$$v_{t+1} = -\frac{\sqrt{x_t + \epsilon}\,\delta L(\theta_t)}{\sqrt{g_{t+1} + \epsilon}}$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$

1.2.6 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.

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

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

Ưu điểm :

Ưu điểm rõ nhất của RMSprop là giải quyết được vấn đề tốc độ học giảm dần của Adagrad (vấn đề tốc độ học giảm dần theo thời gian sẽ khiến việc training chậm dần, có thể dẫn tới bị đóng băng)

Nhược điểm :

Thuật toán RMSprop có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum như Momentum. Vì vậy người ta sẽ kết hợp cả 2 thuật toán Momentum với RMSprop cho ra 1 thuật toán tối ưu Adam.

1.2.7 Adam

Adam (Adaptive Moment Estimation) là một trong những trình tối ưu hóa được ưa thích nhất và là sự lựa chọn tổng thể tốt nhất. Nó sử dụng mức trung bình giảm dần theo cấp số nhân của độ dốc và bình phương của độ dốc để xác định thang đo được cập nhật. Nó cũng sử dụng thuật ngữ momentum (động lượng) để giúp trình

tối ưu hóa di chuyển hiệu quả hơn thông qua loss function. Quy tắc cập nhật có thể được viết như sau:

1. Khởi tạo đường trung bình động của khoảnh khắc thứ nhất và thứ hai (m và v) về 0.
2. Tính toán gradient của hàm mất mát (J) theo từng tham số mô hình (∇J(θ)).
3. Cập nhật biến m và v

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

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

m, v lần lượt là vector động lượng và vận tốc

The values for β1 is 0.9 , 0.999 for β2, and (10 x exp(-8)) for ‘ϵ’.

1. Cập nhật tham số

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

Công thức cập nhật biến m và v:

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

$$v_t = \beta 2 * v_t - 1 + (1 - \beta 2) * \nabla J(\theta)\text{^}2$$

- θ_{t+1}: Tập hợp các tham số của mô hình tại thời điểm t + 1.
- θ_t: Tập hợp các tham số của mô hình tại thời điểm t.
- α: Learning rate.
- ∇J(θ): Gradient của hàm mất mát J(θ) tại θ.
- v_t: Biến tích lũy của bình phương gradient của từng tham số tại thời điểm t.
- v_{t-1}: Biến tích lũy của bình phương gradient của từng tham số tại thời điểm (t-1)

2. Lặp lại:

Lặp lại các bước 2, 3, 4 cho đến khi hàm mất mát đạt giá trị tối thiểu mong muốn hoặc không còn thay đổi đáng kể.

Ưu điểm:

- Đơn giản để thực hiện
- Tính toán hiệu quả
- Yêu cầu bộ nhớ nhỏ
- Thích hợp cho các vấn đề có độ dốc rất nhiễu/hoặc thưa thớt
- Siêu tham số có khả năng diễn giải trực quan và thường yêu cầu điều chỉnh một chút

Bảng 1.2: Bảng so sang AdaGrad, AdaDelta, RMSProp, Adam

Đặc điểm	**AdaGrad**	**AdaDelta**	**RMSProp**	**Adam**
Tính chất	Tăng tốc độ hội tụ, giảm thiểu hiện tượng nhiễu	Giảm thiểu hiện tượng nhiễu	Giảm thiểu hiện tượng nhiễu	Tăng tốc độ hội tụ, giảm thiểu hiện tượng nhiễu
Ưu điểm	Tốc độ hội tụ nhanh	Tốc độ hội tụ nhanh	Tốc độ hội tụ nhanh	Tốc độ hội tụ nhanh, ít bị mắc kẹt trong các điểm tối thiểu cục bộ
Nhược điểm	Khó điều chỉnh các tham số	Khó điều chỉnh các tham số	Khó điều chỉnh các tham số	Khó điều chỉnh các tham số
Ứng dụng	Phổ biến trong các tập dữ liệu lớn và phức tạp			

1.3 Nên sử dụng Trình tối ưu hóa nào và khi nào?

Không phải mọi Trình tối ưu hóa đều tốt cho mọi tình huống mà chúng tôi cần xác định trình tối ưu hóa nào là tốt nhất theo nhiệm vụ của mình.

Một số trình tối ưu hóa hoạt động tốt hơn trong các tác vụ Thị giác máy tính, một số khác hoạt động tốt hơn khi nói đến RNN và một số trong số chúng hoạt động tốt hơn khi dữ liệu đầu vào thưa thớt.

Lợi ích chính của trình tối ưu hóa dựa trên tốc độ học thích ứng (η) là chúng ta không cần điều chỉnh tốc độ học nhưng vẫn có thể đạt được kết quả tốt nhất với các giá trị mặc định.

Việc lựa chọn trình tối ưu hóa phụ thuộc vào một số yếu tố, bao gồm:

- Kích thước của tập dữ liệu: Các trình tối ưu hóa như Gradient Descent và Stochastic Gradient Descent có thể hiệu quả với các tập dữ liệu nhỏ, nhưng có thể chậm với các tập dữ liệu lớn. Các trình tối ưu hóa như AdaGrad, AdaDelta, RMSProp và Adam có thể hiệu quả hơn với các tập dữ liệu lớn.
- Tính phức tạp của hàm mục tiêu: Các trình tối ưu hóa như Gradient Descent và Stochastic Gradient Descent có thể hiệu quả với các hàm mục tiêu đơn giản, nhưng có thể kém hiệu quả với các hàm mục tiêu phức tạp. Các trình tối ưu hóa như AdaGrad, AdaDelta, RMSProp và Adam có thể hiệu quả hơn với các hàm mục tiêu phức tạp.
- Khả năng mắc kẹt trong các điểm tối thiểu cục bộ: Các trình tối ưu hóa như Gradient Descent và Stochastic Gradient Descent có thể dễ bị mắc kẹt trong các điểm tối thiểu cục bộ. Các trình tối ưu hóa như AdaGrad, AdaDelta, RMSProp và Adam có thể ít bị mắc kẹt trong các điểm tối thiểu cục bộ.

Dựa trên các yếu tố này, ta có thể đưa ra một số hướng dẫn chung về việc lựa chọn trình tối ưu hóa:

- Với các tập dữ liệu nhỏ, ta có thể sử dụng Gradient Descent hoặc Stochastic Gradient Descent.
- Với các tập dữ liệu lớn, ta có thể sử dụng AdaGrad, AdaDelta, RMSProp hoặc Adam.
- Với các hàm mục tiêu phức tạp, ta có thể sử dụng AdaGrad, AdaDelta, RMSProp hoặc Adam.
- Với các bài toán dễ bị mắc kẹt trong các điểm tối thiểu cục bộ, ta có thể sử dụng AdaGrad, AdaDelta, RMSProp hoặc Adam.

CHƯƠNG 2. TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION

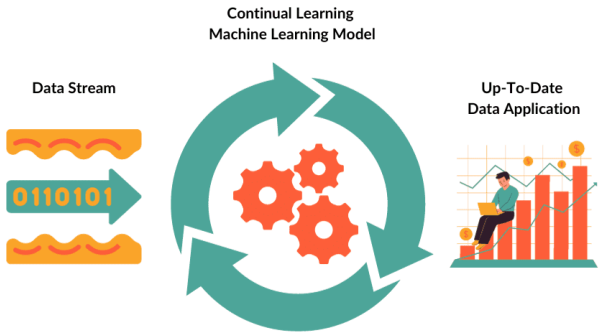
2.1 Continual Learning

2.1.1 Continual Learning là gì?

Học liên tục, còn được gọi là học máy liên tục (CML), là một quá trình trong đó một mô hình học từ các luồng dữ liệu mới mà không cần đào tạo lại.

Trái ngược với các phương pháp tiếp cận truyền thống, trong đó các mô hình được đào tạo trên một tập dữ liệu tĩnh, được triển khai và đào tạo lại định kỳ, các mô hình học liên tục cập nhật lặp đi lặp lại các tham số của chúng để phản ánh các phân phối mới trong dữ liệu.

Mô hình tự cải thiện bằng cách học hỏi từ lần lặp mới nhất và cập nhật kiến thức của nó khi có dữ liệu mới. Vòng đời của mô hình học tập liên tục cho phép các mô hình duy trì tính phù hợp theo thời gian nhờ chất lượng năng động vốn có của chúng.



Hình 2.1 Hình minh họa quá trình học liên tục

Học tập liên tục thường bị hiểu sai là một lớp thuật toán ML đặc biệt. Điều này là do một số thuật toán ML, chẳng hạn như cập nhật Bayes tuần tự và Bộ phân loại KNN, có thể được cập nhật dần dần theo thời gian khi có dữ liệu mới. Tuy nhiên, học tập liên tục có thể được áp dụng cho bất kỳ thuật toán ML được giám sát nào.

Học tập liên tục cũng thường bị hiểu sai là bắt đầu huấn luyện lại từ đầu mỗi khi có dữ liệu mới. Điều này rất nguy hiểm vì nó có thể khiến mô hình bị lãng quên

một cách thảm khốc. Khi mô hình được huấn luyện lại từ đầu, nó sẽ quên những gì đã học từ dữ liệu cũ, và điều này có thể dẫn đến giảm hiệu suất.

Hầu hết các công ty sử dụng học tập liên tục cập nhật mô hình của họ theo từng đợt nhỏ. Điều này có nghĩa là mô hình được cập nhật sau khi một số lượng dữ liệu nhất định được thu thập. Số lượng ví dụ tối ưu phụ thuộc vào nhiệm vụ cụ thể. Ví dụ, nếu mô hình được sử dụng để phát hiện các giao dịch gian lận, thì cần cập nhật mô hình thường xuyên hơn so với mô hình được sử dụng để dự đoán thời tiết.

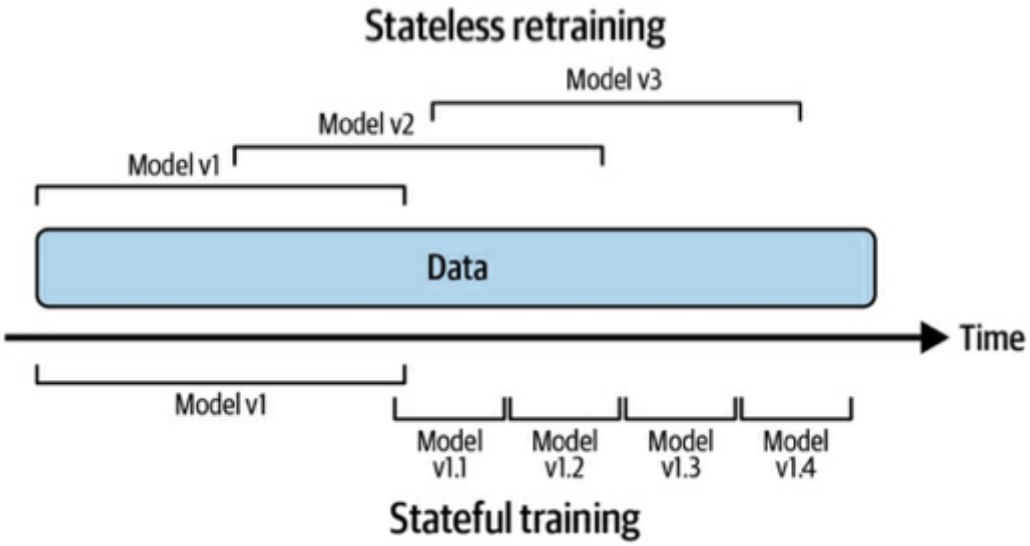
Cuối cùng, học tập liên tục đòi hỏi cả kỹ năng của nhà khoa học dữ liệu và công nghệ cơ sở hạ tầng. Nhà khoa học dữ liệu cần thiết kế và triển khai giải pháp học tập liên tục. Công nghệ cơ sở hạ tầng cần thiết để thu thập và lưu trữ dữ liệu mới, cũng như để cập nhật mô hình.

2.1.2 Ưu điểm của việc học liên tục:

Học tập liên tục có thể hữu ích cho tất cả các loại dự án dữ liệu: mô tả, chẩn đoán, dự đoán và quy định. Nó đặc biệt quan trọng trong các trường hợp liên quan đến dữ liệu thay đổi nhanh. Các lợi ích so với phương pháp học máy truyền thống bao gồm:

- Tổng quát hóa. Việc học hỏi liên tục giúp mô hình trở nên mạnh mẽ và chính xác hơn khi xử lý dữ liệu mới.
- Lưu giữ thông tin. Bằng cách sử dụng chiến lược học hỏi liên tục, mô hình sẽ xem xét kiến thức trước đây thu được trong các lần lặp lại trước đây, cho phép mô hình tích lũy thông tin theo thời gian.
- Khả năng thích ứng. Một mô hình áp dụng phương pháp học tập liên tục sẽ thích ứng với kiến thức mới - chẳng hạn như sự thay đổi khái niệm và xu hướng mới - nhờ đó có khả năng dự đoán tốt hơn về lâu dài.

2.1.3 Các phương pháp học liên tục:



Hình 2.2 Hình minh họa Stateless Retraining và Statefull Training

Trong continuous learning, việc lựa chọn giữa stateless retraining và stateful training là một quyết định quan trọng ảnh hưởng đến hiệu suất và hiệu quả của mô hình. Hãy cùng phân tích sự khác biệt giữa hai cách tiếp cận này:

Stateless Retraining:

Huấn luyện lại mô hình từ đầu trên toàn bộ dữ liệu hiện có (bao gồm cả dữ liệu cũ và mới). Không ghi nhớ trạng thái hoặc kiến thức đã học trước đó.

Dễ hiểu và triển khai.

Tốn nhiều tính toán và thời gian, không phù hợp với dữ liệu thay đổi nhanh.

Có thể quên kiến thức cũ (Catastrophic Forgetting).

Nhiều công ty bắt đầu với phương pháp này.

Stateful Training:

Cập nhật mô hình dần dần với dữ liệu mới, dựa trên trạng thái của mô hình hiện tại (bao gồm kiến thức đã học trước đó). Sử dụng các kỹ thuật như regularizers, replaying, và meta-learning để duy trì kiến thức cũ.

Sử dụng ít dữ liệu hơn, học nhanh hơn, tiết kiệm tài nguyên. Giảm nguy cơ quên kiến thức cũ.

Phức tạp hơn để thiết kế và triển khai. Hiệu suất có thể thấp hơn trong một số trường hợp. Thỉnh thoảng vẫn cần huấn luyện lại stateless để hiệu chỉnh lại mô hình.

Thay đổi từ stateless sang statefull dễ dàng nếu cơ sở hạ tầng phù hợp. Thích hợp cập nhật dữ liệu vào mô hình cố định, thay đổi kiến trúc cần huấn luyện không trạng thái trước.

Kết luận:

Stateless Retraining đơn giản nhưng tốn kém, còn Stateful Training hiệu quả hơn nhưng phức tạp hơn. Lựa chọn phương pháp phụ thuộc vào nhu cầu cụ thể của ứng dụng học tập liên tục.

2.1.4 Các giai đoạn của continuous learning:

Các công ty thường hướng tới việc học tập liên tục theo bốn giai đoạn.

Giai đoạn 1: Huấn luyện lại thủ công, không trạng thái (Manual, stateless retraining)

Cập nhật mô hình chỉ khi hiệu suất giảm mạnh và có thời gian để cập nhật. Thực hiện thủ công, đơn giản nhưng tốn thời gian và công sức.

Giai đoạn 2: Huấn luyện lại tự động theo lịch trình, không trạng thái (Fixed schedule automated stateless retraining)

Cập nhật mô hình theo lịch trình cố định, thường dựa trên "cảm tính". Giúp giảm thiểu công sức thủ công, nhưng vẫn chưa linh hoạt. Cần script tự động chạy huấn luyện lại, triển khai và đánh giá mô hình. Cần thêm hệ thống lên lịch và lưu trữ mô hình.

Giai đoạn 3: Huấn luyện có trạng thái theo lịch trình cố định (Fixed schedule automated stateful training)

Cập nhật mô hình dần dần dựa trên phiên bản trước, theo dõi lịch sử phiên bản. Hiệu quả hơn, tiết kiệm tài nguyên hơn so với huấn luyện không trạng thái. Cần script và hệ thống theo dõi lịch sử mô hình, phiên bản dữ liệu. Hầu hết các hệ thống lưu trữ mô hình hiện tại chưa hỗ trợ tính năng này.

Giai đoạn 4: Học tập liên tục (Continual learning)

Cập nhật mô hình tự động dựa trên các kích hoạt: thời gian, hiệu suất, lượng dữ liệu, sự thay đổi dữ liệu. Linh hoạt nhất, nhưng cần thiết kế hệ thống kích hoạt và theo dõi hiệu quả phức tạp. Dự đoán sự thay đổi dữ liệu và ảnh hưởng đến hiệu suất là thách thức lớn.

Tóm lại, Continuous Learning tiến triển qua các giai đoạn từ thủ công đến tự động, từ đơn giản đến phức tạp, nhằm tối ưu hiệu quả và giảm thiểu công sức cập nhật mô hình.

2.1.5 Các thách thức của học liên tục

Catastrophic forgetting: Mô hình có thể quên kiến thức đã học trước đó khi được huấn luyện trên dữ liệu mới.

Tính ổn định: Mô hình có thể bị ảnh hưởng bởi dữ liệu nhiễu hoặc bất thường.

Hiệu quả tính toán: Cập nhật mô hình liên tục có thể tốn nhiều thời gian và tài nguyên tính toán.

2.1.6 Các giải pháp cho việc học liên tục

- Regularization: Kỹ thuật hạn chế sự thay đổi của mô hình khi huấn luyện trên dữ liệu mới.
- Replaying: Trình bày lại dữ liệu cũ cho mô hình trong quá trình huấn luyện với dữ liệu mới.
- Meta-learning: Huấn luyện một mô hình "học cách học" để thích nghi hiệu quả với dữ liệu mới.

2.1.7 Các ứng dụng của việc học liên tục

- Nhận dạng giọng nói: được sử dụng để cải thiện độ chính xác của nhận dạng giọng nói trong các tình huống như cuộc gọi điện thoại hoặc trợ lý giọng nói.
- Đề xuất sản phẩm: được sử dụng để đề xuất các sản phẩm liên quan và hấp dẫn hơn cho khách hàng.
- Phân tích cảm xúc: giúp hiểu rõ hơn về cảm xúc của khách hàng, từ đó cải thiện các chiến dịch tiếp thị và dịch vụ khách hàng.
- Tầm nhìn máy tính: cải thiện khả năng xác định và phân loại các đối tượng trong hình ảnh.

- An ninh mạng: phát hiện các hoạt động tấn công mạng như lừa đảo, xâm nhập và spam.
- Chăm sóc sức khỏe: cải thiện độ chính xác của chẩn đoán bệnh.
- Robot: nâng cao khả năng thích ứng và hiệu suất của robot trong các môi trường khác nhau.

Nhìn chung, học tập liên tục là một công nghệ có tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Nó có thể giúp các công ty và tổ chức cải thiện hiệu quả hoạt động và cung cấp các sản phẩm và dịch vụ tốt hơn cho khách hàng.

2.2 Test Production

Kiểm tra mô hình sản xuất (production model testing) là một bước quan trọng trong vòng đời của Machine Learning, đảm bảo mô hình hoạt động chính xác và đáng tin cậy trong môi trường thực tế.

2.2.1 Các loại kiểm tra:

- Kiểm tra tính năng: Đảm bảo mô hình hoạt động như mong đợi với các trường hợp đầu vào khác nhau.
- Kiểm tra hiệu suất: Đo lường độ chính xác, độ nhạy, độ đặc và các chỉ số hiệu suất khác của mô hình.
- Kiểm tra độ ổn định: Kiểm tra xem mô hình có bị ảnh hưởng bởi các thay đổi trong dữ liệu đầu vào hoặc môi trường sản xuất hay không.
- Kiểm tra drift: Theo dõi các thay đổi theo thời gian trong dữ liệu phân phối và hiệu suất mô hình, phát hiện hiện tượng "data drift" kịp thời.
- Kiểm tra bảo mật: Xác nhận rằng mô hình không thể bị tấn công hoặc khai thác.

2.2.2 Các phương pháp kiểm tra:

- Kiểm tra thủ công: Thực hiện thủ công các trường hợp kiểm thử khác nhau.
- Kiểm tra tự động: Sử dụng các bộ kiểm thử tự động để chạy các trường hợp kiểm thử một cách lặp lại.

- Kiểm tra A/B: So sánh hiệu suất của mô hình sản xuất với một mô hình khác trong môi trường thực tế.
- Kiểm tra giám sát: Theo dõi liên tục các hoạt động của mô hình trong sản xuất để phát hiện các vấn đề tiềm ẩn.

2.2.3 Thách thức:

Môi trường sản xuất phức tạp: Khác biệt so với môi trường phát triển, môi trường sản xuất có thể gặp nhiều sự kiện bất ngờ và dữ liệu khác biệt.

Độ trôi dữ liệu (data drift): Dữ liệu theo thời gian có thể thay đổi, khiến mô hình dần kém chính xác.

Tính sẵn sàng của hệ thống: Kiểm tra không nên ảnh hưởng đến hoạt động của mô hình trong sản xuất.

2.2.4 Lợi ích:

Ngăn ngừa lỗi trong sản xuất: Phát hiện các vấn đề tiềm ẩn trước khi ảnh hưởng đến người dùng.

Cải thiện hiệu suất mô hình: Điều chỉnh và tối ưu hóa mô hình dựa trên kết quả kiểm tra.

Xây dựng sự tin cậy vào mô hình: Đảm bảo mô hình hoạt động như mong đợi và đáng tin cậy.

2.2.5 Các bước thực hiện

Xác định các rủi ro và yêu cầu kiểm tra cụ thể.

Chọn các phương pháp kiểm tra phù hợp.

Thiết kế các bộ kiểm thử toàn diện.

Xây dựng cơ sở hạ tầng để thực hiện kiểm tra.

Giám sát liên tục kết quả kiểm tra và thực hiện hành động khắc phục.

TÀI LIỆU THAM KHẢO

Doshi, S. (2020). Various Optimization Algorithms For Training Neural Network. Retrieved from https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Malingan, N. (2023). Optimizers in Deep Learning. Retrieved from https://www.scaler.com/topics/deep-learning/optimizers-in-deep-learning/

Mhaisekar, K. (2020). AdaBelief Optimizer: fast as Adam, generalizes as well as SGD. Retrieved from https://towardsdatascience.com/adabelief-optimizer-fast-as-adam-generalizes-as-good-as-sgd-71a919597af

Brownlee, J. (2021). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Retrieved from https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Wagh, A. (2022). Gradient Descent and its Types. Retrieved from https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Bansal, S. (2020). The Math and Intuition Behind Gradient Descent. Retrieved from https://medium.datadriveninvestor.com/the-math-and-intuition-behind-gradient-descent-13c45f367a11

Jain, N. (2020). An overview of the Gradient Descent algorithm. Retrieved from https://towardsdatascience.com/an-overview-of-the-gradient-descent-algorithm-8645c9e4de1e

(N.d.). Retrieved from https://arxiv.org/pdf/2010.07468.pdf

Bindal, A. (2019). Some State of the Art Optimizers in Neural Networks. Retrieved from https://hackernoon.com/some-state-of-the-art-optimizers-in-neural-networks-a3c2ba5a5643?action=2