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

ĐỒ ÁN CUỐI KÌ
NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN ĐÌNH DANH - 52100878**
Lớp **: 21050301**
Khoá **: 25**

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

ĐỒ ÁN CUỐI KÌ
NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN ĐÌNH DANH - 52100878**
Lớp **: 21050301**
Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc tới thầy Lê Anh Cường về sự dành thời gian và tâm huyết trong việc hướng dẫn môn Nhập môn Học máy. Những giờ học với thầy không chỉ là cơ hội để em học hỏi kiến thức mới mà còn là những trải nghiệm quý báu, giúp em hiểu rõ hơn về lĩnh vực này.

Sự chân thành và sự nhiệt huyết của thầy là nguồn động viên lớn, giúp em có động lực hơn trong quá trình nghiên cứu và thực hành. Em đánh giá cao không chỉ kiến thức chuyên sâu của thầy mà còn sự truyền đạt linh hoạt và dễ hiểu, giúp em tiếp cận môn học một cách dễ dàng hơn.

Em chân thành cảm ơn thầy và trường Đại học Tôn Đức Thắng đã tạo điều kiện cho em có cơ hội học tập và phát triển. Em rất tự hào và biết ơn vì đã có thầy là người hướng dẫn.

Chúc thầy sức khỏe và thành công trong công việc giảng dạy và nghiên cứu. Em hân hạnh được làm sinh viên của thầy.

Sinh viên

Nguyễn Đình Danh

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

Tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của 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 đồ 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 đồ 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)

Nguyễn Đình Danh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Báo cáo đề tài này tập trung vào hai khía cạnh chính của học máy: tối ưu hóa quá trình huấn luyện mô hình và ứng dụng của Continual Learning và Test Production trong việc giải quyết các bài toán cụ thể.

Tối Ưu Hóa Quá Trình Huấn Luyện Mô Hình:

- Nghiên cứu và so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy.
- Đánh giá hiệu suất và ổn định của các Optimizer phổ biến như Adam, SGD, và RMSprop.
- Xem xét ảnh hưởng của các tham số hyperparameter lên quá trình học và độ chính xác của mô hình.

Continual Learning và Test Production:

- Nắm vững kiến thức về Continual Learning và áp dụng chúng trong xây dựng giải pháp học máy.
- Nghiên cứu và đánh giá cách Continual Learning giúp mô hình duy trì và cập nhật kiến thức khi có dữ liệu mới.
- Tìm hiểu về Test Production và cách nó hỗ trợ việc xây dựng mô hình có khả năng dự đoán tốt trên dữ liệu mới và đối mặt với môi trường thực tế.

Bằng cách kết hợp nghiên cứu sâu sắc về các phương pháp tối ưu hóa và áp dụng linh hoạt của Continual Learning và Test Production, đề tài này hứa hẹn mang lại cái nhìn tổng thể và ứng dụng thực tế vững chắc về cả hai lĩnh vực quan trọng trong học máy.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 - TÌM HIỂU VỀ OPTIMIZER, CONTINUAL LEARNING VÀ TEST PRODUCTION TRONG HỌC MÁY

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

Trong lĩnh vực học máy, quá trình huấn luyện mô hình là một giai đoạn quan trọng để đạt được hiệu suất tối ưu trên tập dữ liệu huấn luyện. Trong quá trình này, Optimizer đóng vai trò quan trọng trong việc tối ưu hóa các tham số của mô hình, đồng thời giúp mô hình hội tụ nhanh chóng và ổn định. Trên thực tế, có nhiều phương pháp Optimizer khác nhau được sử dụng phổ biến, mỗi phương pháp có ưu điểm và hạn chế riêng.

Bài báo cáo này tập trung vào việc tìm hiểu và so sánh các phương pháp Optimizer phổ biến nhất trong huấn luyện mô hình học máy. Chúng ta sẽ xem xét một số phương pháp Optimizer quan trọng sau đây:

1.1.1. Gradient descent

1.1.1.1. Giới thiệu

Gradient Descent hoạt động dựa trên ý tưởng cơ bản của việc di chuyển ngược chiều của đạo hàm của hàm mất mát để tìm giá trị tối ưu. Quá trình này được thực hiện bằng cách cập nhật các tham số theo hướng ngược với gradient với một tỷ lệ học (learning rate) đã cho. Mục tiêu của GD là tìm ra các tham số tối ưu nhằm giảm thiểu giá trị của hàm mất mát.

1.1.1.2. Công thức chung và các bước cụ thể (dùng cho cả Gradient descent đơn và đa biến)

Công thức chung cập nhật cho mỗi tham số $\boldsymbol{\theta_i}$ trong mô hình được mô tả như sau:

$$\boldsymbol{\theta_{i\_new} = \theta_{i\_old} - \alpha . \frac{\partial J}{\partial \theta_{i\_old}}}$$

Trong đó:

- $\boldsymbol{\theta_{i\_new}}$ là giá trị tham số mới.
- $\boldsymbol{\theta_{i\_old}}$ là giá trị tham số cũ.
- $\boldsymbol{\alpha}$ là learning rate - tốc độ học.
- **J** là hàm mất mát.
- $\frac{\partial J}{\partial \theta_{i\_old}}$ là đạo hàm riêng của hàm mất mát **J** theo $\boldsymbol{\theta_{i\_old}}$.

1.1.1.3. Các bước cụ thể:

1. Chọn giá trị khởi tạo cho tham số θ và learning rate $\boldsymbol{\alpha}$.
2. Xác định hàm mất mát .
3. Lặp qua các bước cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp tối đa hoặc sự hội tụ đủ):
   - Tính đạo hàm của hàm mất mát $\boldsymbol{J}$ theo từng tham số $\boldsymbol{\theta_i}$.
   - Cập nhật tham số bằng công thức $\boldsymbol{\theta_{i\_new} = \theta_{i\_old} - \alpha . \frac{\partial J}{\partial \theta_{i\_old}}}$ đã nêu bên trên.
   - Kiểm tra điều kiện dừng. Điều kiện dừng có thể là số lần lặp tối đa, sự hội tụ đủ, hoặc một điều kiện khác tùy thuộc vào vấn đề cụ thể.
4. Khi thuật toán kết thúc, giá trị của các tham số **θ** đã được điều chỉnh để giảm thiểu hàm mất mát và phản ánh mô hình tốt hơn với dữ liệu đào tạo.

1.1.1.4. Ưu và nhược điểm

Ưu điểm:

Gradient Descent là một thuật toán cơ bản và dễ hiểu, giúp tối ưu hóa các mô hình neural network bằng cách cập nhật trọng số sau mỗi vòng lặp.

Nhược điểm:

Với tính đơn giản, thuật toán Gradient Descent cũng đi kèm với một số hạn chế đáng lưu ý:

- Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate, có thể ảnh hưởng đến hiệu suất của thuật toán.
- Trong trường hợp hàm mất mát có nhiều global minimum, kết quả cuối cùng phụ thuộc vào điểm khởi tạo ban đầu, có thể dẫn đến sự không đồng nhất trong kết quả.
- Tốc độ học quá lớn có thể làm cho thuật toán không hội tụ và quanh quẩn xung quanh điểm cực tiểu vì bước nhảy quá lớn. Ngược lại, tốc độ học quá nhỏ có thể ảnh hưởng đến tốc độ training, làm chậm quá trình học.

1.1.2. Stochastic Gradient Descent (SGD)

1.1.2.1. Giới Thiệu

Stochastic Gradient Descent (SGD) là một thuật toán tối ưu hóa thường được sử dụng trong học máy và deep learning. Nó là một biến thể của thuật toán Gradient Descent (GD) nhằm giảm độ phức tạp tính toán khi đối mặt với dữ liệu lớn.

Thuật toán SGD khác với GD truyền thống ở cách tính toán gradient và cập nhật tham số. Thay vì tính toán gradient trên toàn bộ tập dữ liệu huấn luyện, SGD chỉ sử dụng một mẫu dữ liệu ngẫu nhiên (mini-batch) để tính gradient và cập nhật tham số mô hình ở mỗi bước lặp.

1.1.2.2. Các bước cụ thể:

- Khởi tạo giá trị ban đầu: Đặt giá trị ban đầu cho các tham số trong mô hình.
- Xác định hàm mất mát .

- Lặp lại các bước sau cho đến khi đạt điều kiện dừng:

a. Lựa chọn một mini-batch ngẫu nhiên từ tập dữ liệu huấn luyện. Mini-batch thường có kích thước nhỏ hơn so với toàn bộ tập dữ liệu.

b. Tính toán giá trị dự đoán của mô hình dựa trên các tham số hiện tại và mini-batch được chọn.

c. Tính toán gradient của hàm mất mát (đạo hàm của hàm mất mát) theo từng tham số, sử dụng mini-batch đó. Gradient được tính toán bằng cách áp dụng quy tắc chuỗi đạo hàm (chain rule) cho từng mẫu dữ liệu trong mini-batch.

d. Cập nhật các tham số bằng cách di chuyển ngược hướng gradient với một tỷ lệ học (learning rate) đã chọn:

$$\theta_{i\_new} = \theta_{i\_old} - \alpha . \frac{\partial J}{\partial \theta_{i\_old}}$$

- Lặp lại quá trình trên (số lượng epoch) cho đến khi đạt được điều kiện dừng (ví dụ: đạt đủ số lần lặp, hàm mất mát không thay đổi đáng kể, hoặc đạt được độ chính xác mong muốn).

1.1.2.3. Ưu và nhược điểm:

Ưu điểm

- Tiết kiệm thời gian tính toán, SGD chỉ tính toán gradient trên một mini-batch nhỏ, giúp giảm độ phức tạp tính toán so với Gradient Descent truyền thống, đặc biệt là khi làm việc với dữ liệu lớn.
- Khả năng xử lý dữ liệu lớn. Vì SGD chỉ yêu cầu một lượng nhỏ dữ liệu để tính toán gradient nên có thể huấn luyện các mô hình trên dữ liệu khổng lồ mà không cần tốn nhiều tài nguyên tính toán.
- Hội tụ nhanh. Với mỗi bước cập nhật, SGD sử dụng gradient dựa trên một mẫu ngẫu nhiên, giúp thuật toán thích nghi nhanh hơn với dữ liệu và có khả năng hội tụ nhanh đến điểm tối ưu cục bộ.

Nhược điểm:

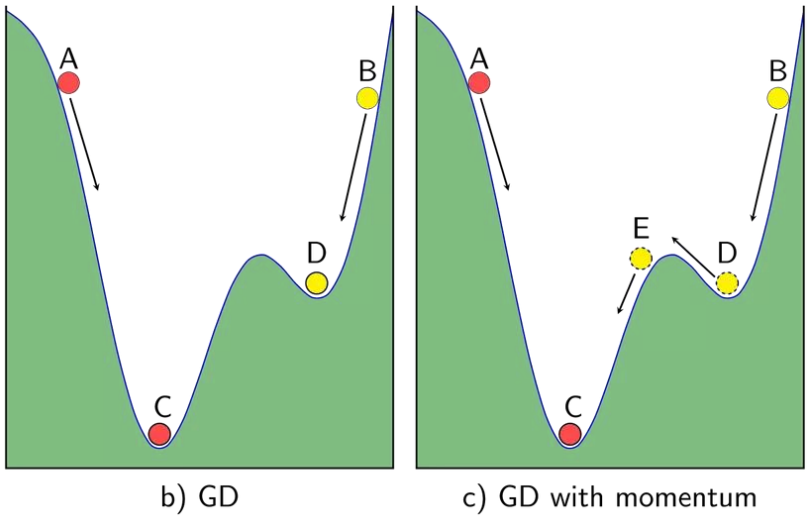
Vẫn mang những nhược điểm của gradient descent truyền thống và chưa thể khắc phục được về learning rate, điểm dữ liệu ban đầu…

1.1.3. Momentum

Để khắc phục các hạn chế trên của thuật toán Gradient Descent, ta có thể kết hợp gradient descent với Momentum.

1.1.3.1. Giới thiệu

Trước khi giải thích trong lĩnh vực Machine learning, ta hãy nhìn Momentum - "đà" - dưới góc nhìn vật lý:



Hình 1: Momentum

Để hiểu Gradient with Momentum, hãy xem xét từ góc độ vật lý: Giả sử ta thả hai viên bi từ hai điểm khác nhau A và B. Khi đó, viên bi tại A sẽ trượt xuống đến điểm C, trong khi viên bi tại B sẽ trượt xuống đến điểm D. Tuy nhiên, ta không muốn viên bi tại B dừng lại ở điểm D (điểm cực tiểu cục bộ), mà mong muốn nó tiếp tục lăn tới điểm C (điểm cực tiểu toàn cục). Để thực hiện điều này, ta cung cấp cho viên bi B một vận tốc ban đầu đủ lớn để có thể vượt qua điểm E và đạt tới điểm C.

Dựa trên ý tưởng này, thuật toán Momentum được tạo ra, nơi chúng ta tạo ra "quán tính" để viên bi - tức quá trình tối ưu - vượt qua các điểm cực tiểu cục bộ và tiến gần đến điểm cực tiểu toàn cục.

1.1.3.2. Công thức tổng quát

$$v_t = \beta v_{t-1} - (1-\beta).\frac{\partial J}{\partial \theta_i}$$

$$\theta_{i\_new} = \theta_{i\_old} - \alpha.v_t$$

Trong đó:

- $\boldsymbol{\theta_{i\_new}}$ là giá trị tham số mới.
- $\boldsymbol{\theta_{i\_old}}$ là giá trị tham số cũ.
- $\boldsymbol{\alpha}$ là learning rate - tốc độ học.
- **J** là hàm mất mát.
- $\frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \theta_i}}$ là đạo hàm riêng của hàm mất mát **J** theo $\boldsymbol{\theta_i}$.
- $\boldsymbol{v_t}$ là giá trị momentum tại vòng lặp thứ t
- $\boldsymbol{\beta}$ là hệ số momentum

1.1.3.3. Các bước thực hiện

1. Chọn giá trị khởi tạo cho tham số θ và learning rate $\boldsymbol{\alpha}$.
2. Xác định hàm mất mát .
3. Lặp qua các bước cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp tối đa hoặc sự hội tụ đủ):
   - Tính đạo hàm của hàm mất mát.
   - Cập nhật giá trị momentum $\boldsymbol{v_t}$
   - Cập nhật giá trị của tham số $\boldsymbol{\theta_i}$.
   - Kiểm tra điều kiện dừng.

4. Khi thuật toán kết thúc, giá trị của các tham số $\boldsymbol{\theta}$ đã được điều chỉnh để giảm thiểu hàm mất mát.

1.1.3.4. Ưu nhược điểm

Ưu điểm:

Thuật toán tối ưu Momentum giúp khắc phục vấn đề của Gradient Descent, nơi mà việc tìm kiếm global minimum có thể bị gián đoạn ở local minimum.

Nhược điểm:

Mặc dù Momentum giúp "hòn bi" - tức quá trình tối ưu hoá - vượt qua đồi và tiến gần đến điểm đích, nhưng khi đã gần đích, nó vẫn tiếp tục giao động qua lại trước khi dừng hoàn toàn. Hiện tượng này có thể được giải thích bằng việc "hòn bi" vẫn giữ lại một lượng đà, và do đó, nó tiếp tục di chuyển với dao động trước khi dừng hẳn.

1.1.4. Adagrad

1.1.4.1. Giới thiệu

Adagrad đặt một cách tiếp cận động (adaptive) cho việc quản lý tỷ lệ học (learning rate) trong quá trình đào tạo mô hình. Trong khi các thuật toán trước đó sử dụng learning rate cố định (hằng số) cho toàn bộ quá trình, Adagrad xem xét từng tham số riêng lẻ và điều chỉnh learning rate cho mỗi tham số theo lịch sử của gradient liên quan.

Điều này có nghĩa là những tham số nào gặp gradient lớn hơn sẽ có learning rate giảm, còn những tham số có gradient nhỏ hơn sẽ có learning rate tăng lên. Điều này giúp quá trình đào tạo trở nên linh hoạt hơn và có hiệu suất tốt hơn đối với các tham số có đóng góp lớn hơn vào quá trình học. Adagrad có thể tự động điều chỉnh learning rate tùy thuộc vào độ quan trọng của từng tham số trong mô hình.

1.1.4.2. Công thức cập nhật

$$\boldsymbol{\theta_{i\_new}} = \boldsymbol{\theta_i} - \frac{\boldsymbol{\alpha}}{\sqrt{\boldsymbol{G_i} + \boldsymbol{\epsilon}}} \cdot \frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \theta_i}}$$

Trong đó:

- $\boldsymbol{\theta_{i\_new}}$ là giá trị tham số mới.
- $\boldsymbol{\theta_i}$ là giá trị tham số cũ.
- $\boldsymbol{\alpha}$ là learning rate - tốc độ học.
- **J** là hàm mất mát.
- $\frac{\partial J}{\partial \theta_i}$ là đạo hàm riêng của hàm mất mát **J** theo $\boldsymbol{\theta_i}$.
- $\boldsymbol{G_i}$ là tổng bình phương của đạo hàm của tham số từ vòng lặp đầu tiên đến vòng lặp thứ i.

1.1.4.3. Ưu và nhược điểm

Ưu điểm:

Adagrad giúp giảm bớt bước điều chỉnh learning rate bằng tay trong quá trình đào tạo. Việc chỉ cần thiết lập tốc độ học mặc định và thuật toán sẽ tự động điều chỉnh nó tùy thuộc vào lịch sử của gradient.

Nhược điểm:

Một yếu điểm của Adagrad là sự tích tụ của bình phương gradient theo thời gian dẫn đến việc tốc độ học giảm dần. Điều này có thể làm cho quá trình đào tạo trở nên chậm lại hoặc thậm chí đóng băng nếu tổng bình phương gradient trở nên quá lớn. Điều này có thể làm cho tốc độ học quá nhỏ và mô hình không còn học được nữa.

1.1.5. RMSprop

1.1.5.1. Giới thiệu

RMSprop (Root Mean Square Propagation) là một thuật toán tối ưu hóa gradient descent được thiết kế để giải quyết vấn đề của Adagrad liên quan đến việc giảm dần tỷ lệ học theo thời gian. RMSprop sử dụng một cơ chế chuẩn hóa để điều chỉnh learning rate cho từng tham số riêng lẻ, tạo ra sự linh hoạt và hiệu quả hơn trong quá trình đào tạo mô hình.

Adagrad tích tụ toàn bộ lịch sử của gradient, điều này dẫn đến việc tỷ lệ học giảm dần nhanh chóng và có thể làm cho quá trình đào tạo trở nên chậm lại. RMSprop giải quyết vấn đề này bằng cách thay đổi cách tính tổng bình phương gradient.

1.1.5.2. Công thức

$$v_t = \beta v_{t-1} - (1-\beta).\left(\frac{\partial J}{\partial \theta_i}\right)^2$$

$$\theta_{i,t} = \theta_{i,t-1} - \frac{\alpha}{\sqrt{v_t + \epsilon}}.\frac{\partial J}{\partial \theta_i}$$

1.1.5.3. Ưu và nhược điểm

Ưu điểm:

Một ưu điểm rõ nổi của RMSprop là khắc phục vấn đề tốc độ học giảm dần theo thời gian của Adagrad. Vấn đề này có thể làm cho quá trình đào tạo trở nên chậm dần và thậm chí dẫn đến tình trạng đóng băng, làm giảm hiệu suất của mô hình.

Nhược điểm:

Thuật toán RMSprop có thể chỉ đạt được nghiệm cục bộ (local minimum) thay vì nghiệm toàn cục (global minimum) như thuật toán Momentum. Do đó, để tận dụng ưu điểm của cả hai thuật toán, người ta thường kết hợp chúng lại với nhau để tạo ra thuật toán tối ưu hóa Adam. Sự kết hợp này giúp cân bằng giữa tốc độ học và khả năng tránh được các điểm cực tiểu địa phương.

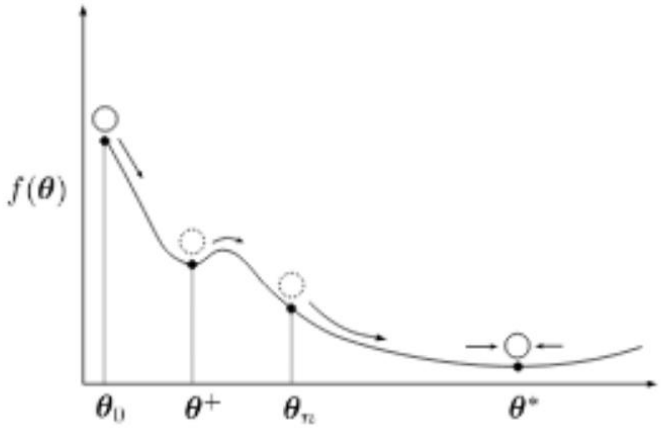
1.1.6. Adam

1.1.6.1. Giới thiệu

Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa gradient descent kết hợp sự mạnh mẽ của hai thuật toán khác là Momentum và RMSprop. Adam được thiết kế để cân bằng giữa tốc độ học và khả năng vượt qua các điểm cực tiểu địa phương.

Nếu mô tả theo hiện tượng vật lý, Momentum có thể được xem như một quả cầu lao xuống dốc, tăng tốc và vượt qua các chướng ngại địa phương. Trái lại, Adam giống

như một quả cầu nặng với ma sát, giúp dễ dàng vượt qua các điểm cực tiểu địa phương và khiến cho quá trình hội tụ nhanh chóng khi tiến tới global minimum.



Hình 2: Minh hoạ thuật toán Adam

1.1.6.2. Công thức

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

Hình 3: Các công thức cho thuật toán Adam.

1.1.7. So sánh các thuật toán

Dưới đây là bảng so sánh các thuật toán trên:

Thuật toán	**Ý nghĩa**	**Ưu điểm**	**Nhược điểm**
Gradient Descent (GD)	Thuật toán tối ưu hóa dựa trên việc cập nhật trọng số theo hướng ngược của gradient.	- Đơn giản và dễ hiểu.	- Cần phải chọn tỷ lệ học tập (learning rate) phù hợp để đảm bảo thuật toán hội tụ.
Stochastic	Phiên bản ngẫu nhiên của	- Hiệu quả tính	- Thuật toán có thể

Thuật toán	**Ý nghĩa**	**Ưu điểm**	**Nhược điểm**
Gradient Descent (SGD)	Gradient Descent, chỉ cập nhật trọng số dựa trên một điểm dữ liệu ngẫu nhiên trong quá trình huấn luyện.	toán và tiết kiệm bộ nhớ hơn so với GD khi xử lý tập dữ liệu lớn.	dao động xung quanh điểm tối ưu và khó đạt đến điểm tối ưu chính xác.
Adagrad	Tối ưu hóa tỷ lệ học tập (learning rate) cho từng trọng số bằng cách điều chỉnh tỷ lệ học tập dựa trên lịch sử các gradient đã tính toán cho trọng số đó.	- Có khả năng tự động điều chỉnh tỷ lệ học tập để phù hợp với từng trọng số.	- Quá trình tích lũy bình phương gradient có thể dẫn đến một tỷ lệ học tập quá nhỏ và dừng sớm quá trình huấn luyện.
RMSprop	Cải tiến của Adagrad, giới hạn việc tích lũy bình phương gradient trong quá khứ bằng cách sử dụng trọng số giữ cho việc tích lũy.	- Giảm hiện tượng tỷ lệ học tập quá nhỏ trong quá trình huấn luyện.	- Cần phải điều chỉnh tham số tỷ lệ học tập.
Adam	Kết hợp cả Momentum và RMSprop. Tích lũy động lượng và điều chỉnh tỷ lệ học tập cho từng trọng số dựa trên bình phương gradient.	- Kết hợp lợi ích của cả Momentum và RMSprop.	- Cần phải điều chỉnh tham số tỷ lệ học tập và động lượng.

Bảng 1: So sánh các thuật toán Optimizer

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. Giới thiệu

Continual Learning và Test Production đều là phần quan trọng trong quá trình xây dựng và triển khai giải pháp học máy.Chúng ta sẽ xem xét chi tiết cả hai khía cạnh này để hiểu rõ hơn về cách chúng đóng góp vào sự thành công của một dự án học máy

2.2. Continual Learning

2.2.1. Giới thiệu

Học liên tục (continuous learning) là một lĩnh vực quan trọng trong machine learning, nơi mà mô hình cần được cập nhật liên tục khi có dữ liệu mới được thêm vào. Để đạt được điều này, có hai phương pháp chính được sử dụng: Stateless Retraining và Stateful Training (Fine-tuning, Incremental Learning).

2.2.2. Đặc điểm chính

- Liên Tục Cập Nhật Kiến Thức

Continual Learning là khả năng của mô hình để học từ dữ liệu mới và duy trì kiến thức đã học trước đó. Điều này giúp mô hình không bao giờ trở nên lạc hậu và có khả năng áp dụng kiến thức cũ vào các tình huống mới.

- Quản lý Quên (Catastrophic Forgetting)

Quản lý quên là một thách thức. Elastic Weight Consolidation (EWC) là một phương pháp để giữ cho mô hình không quên kiến thức đã học trước đó. Nó "tăng cường" các trọng số quan trọng để bảo vệ chúng khỏi việc thay đổi quá mức khi học kiến thức mới.

2.2.3. Các phương pháp

2.2.3.1. Stateless Retraining

- Mô hình được huấn luyện lại từ đầu khi có dữ liệu mới.

- Toàn bộ dữ liệu, cả dữ liệu cũ và mới, được sử dụng trong quá trình huấn luyện.

Ưu Điểm

- Đảm bảo rằng mô hình học được từ tất cả dữ liệu, cả dữ liệu cũ và mới.
- Không có nguy cơ quên thông tin từ dữ liệu cũ.

Nhược Điểm

Chi phí tính toán và thời gian huấn luyện có thể cao khi có nhiều dữ liệu.

Code Ví Dụ

```
# Stateless Retraining
new_data = load_new_data()
all_data = merge_old_and_new_data(old_data, new_data)
model = train_model(all_data)
```

Hình 4: Ví dụ stateless retraining

Ứng dụng:

- Phân Loại: Khi có sự thay đổi đột ngột trong dữ liệu hoặc khi dữ liệu mới có ảnh hưởng đến hiệu suất của mô hình phân loại. Ví Dụ: Mô hình phân loại tin tức trực tuyến.
- Dự Báo: Khi mô hình cần phải dự báo trên dữ liệu mới và không muốn giữ thông tin từ quá khứ. Ví Dụ: Mô hình dự báo giá cổ phiếu.

2.2.3.2. Stateful Training (Fine-tuning, Incremental Learning)

Đặc Điểm Chính

- Mô hình tiếp tục được huấn luyện trên dữ liệu mới mà không cần đặt lại toàn bộ trạng thái.
- Dữ liệu mới được sử dụng để cập nhật mô hình mà không làm ảnh hưởng đến khả năng của mô hình đối với dữ liệu cũ.

Ưu Điểm

- Tiết kiệm tài nguyên tính toán và thời gian so với huấn luyện lại toàn bộ mô hình.

- Linh hoạt và thích ứng tốt với dữ liệu động.

Nhược Điểm

- Có thể có nguy cơ quên thông tin từ dữ liệu cũ nếu không được quản lý cẩn thận.

Code ví dụ

```
# Stateful Training (Fine-tuning)
new_data = load_new_data()
model = load_pretrained_model()
model = train_model_incrementally(model, new_data)
```

Hình 5: Code mẫu Stateful Training

Ứng dụng:

- Học Liên Tục Trong Thời Gian Thực: Khi mô hình cần được cập nhật ngay lập tức khi có dữ liệu mới và không muốn đào tạo lại toàn bộ mô hình. Ví Dụ: Hệ thống giám sát an ninh.
- Dịch Ngôn Ngữ: Trong các ứng dụng dịch ngôn ngữ, nơi ngôn ngữ thay đổi và mô hình cần cập nhật dữ liệu liên tục.Ví Dụ: Google Translate.
- Tăng Cường Mô Hình Tích Lũy Kiến Thức: Khi muốn mô hình giữ lại kiến thức học được từ dữ liệu cũ để áp dụng cho dữ liệu mới.Ví Dụ: Mô hình học từ người dùng trên các trang web.

2.3. Test Production

Để đảm bảo tính đầy đủ của việc kiểm thử mô hình trước khi triển khai rộng rãi, cần thực hiện cả đánh giá ngoại tuyến trước triển khai và kiểm thử trong môi trường thực tế. Dưới đây là một số phương pháp chính để kiểm thử:

2.3.1. Shadow Deployment:

Ý Tưởng: Triển khai mô hình thách thức (challenger) song song với mô hình hiện tại (champion). Gửi mọi yêu cầu đến cả hai mô hình, nhưng chỉ phục vụ dự đoán của mô hình champion. Lưu lại các dự đoán từ cả hai mô hình để so sánh.

Ưu Điểm:

- An toàn nhất khi triển khai mô hình mới.
- Đơn giản về khái niệm.
- Thu thập đủ dữ liệu nhanh chóng để có ý nghĩa thống kê hơn so với các chiến lược khác.

Nhược Điểm:

- Không thích hợp khi đo lường hiệu suất mô hình phụ thuộc vào quan sát cách người dùng tương tác với dự đoán.
- Tốn kém vì gấp đôi số lượng dự đoán và tài nguyên tính toán.
- Cần giải quyết các trường hợp đặc biệt khi sử dụng chế độ dự đoán trực tuyến.

2.3.2. A/B Testing:

Ý Tưởng: Triển khai mô hình thách thức cùng với mô hình champion và định tuyến một phần lượng traffic đến mô hình thách thức. Dự đoán từ mô hình thách thức được hiển thị cho người dùng. Sử dụng giám sát và phân tích dự đoán trên cả hai mô hình để xác định xem hiệu suất của mô hình thách thức có ý nghĩa thống kê hơn so với mô hình champion hay không.

Ưu Điểm:

- Cho phép thu thập đầy đủ dữ liệu về cách người dùng phản ứng với các mô hình khác nhau.
- Dễ hiểu và có nhiều thư viện, tài liệu hỗ trợ.
- Chi phí thấp vì chỉ có một dự đoán cho mỗi yêu cầu.

Nhược Điểm:

- Ít an toàn hơn so với triển khai bóng đèn. Cần đánh giá rủi ro giữa việc định tuyến traffic đến mô hình thách thức và việc thu thập đủ mẫu nhanh chóng để phân tích.
- Cần lựa chọn giữa chấp nhận rủi ro hơn (định tuyến nhiều traffic đến mô hình B) và thu thập đủ mẫu để phân tích nhanh hơn.

2.3.3. Canary Release:

Ý Tưởng: Triển khai mô hình thách thức và mô hình champion song song, nhưng bắt đầu với mô hình thách thức không nhận traffic. Dần dần chuyển traffic từ mô hình champion sang mô hình thách thức (gọi là canary). Giám sát các chỉ số hiệu suất của mô hình thách thức, nếu chúng trông tốt, tiếp tục cho đến khi toàn bộ traffic đều đến mô hình thách thức.

Ưu Điểm:

- Dễ hiểu.
- Đơn giản nhất nếu công ty đã có cơ sở hạ tầng chức năng.

Nhược Điểm:

- Mở cửa cho khả năng không nghiêm túc trong việc xác định sự khác biệt hiệu suất.
- Nếu không giám sát cẩn thận, có thể xảy ra sự cố. Là lựa chọn có khả năng không an toàn nhất, nhưng lại dễ quay lại trạng thái trước đó.

2.3.3. Thử Nghiệm Chéo (Interleaving Experiments):

Ý Tưởng: Trong A/B testing, một người dùng chỉ nhận dự đoán từ mô hình A hoặc mô hình B. Trong interleaving, một người dùng nhận dự đoán xen kẽ từ cả mô hình A và mô hình B. Đo lường sự ưa thích của người dùng với dự đoán từ mỗi mô hình để xác định hiệu suất.

Ưu Điểm:

- Interleaving có thể xác định mô hình tốt nhất với số lượng mẫu nhỏ hơn so với A/B testing truyền thống.
- Cho phép thu thập dữ liệu về cách người dùng ứng xử với dự đoán.

Nhược Điểm:

- Cài đặt phức tạp hơn A/B testing.
- Cần xem xét các trường hợp đặc biệt nếu mô hình interleaved mất thời gian phản hồi hoặc gặp sự cố.
- Tăng đáng kể nhu cầu tài nguyên tính toán vì mỗi yêu cầu cần dự đoán từ nhiều mô hình.

2.3.5. Bandits:

Ý Tưởng: Bandits là một thuật toán theo dõi hiệu suất hiện tại của mỗi biến thể mô hình và đưa ra quyết định động cho mỗi yêu cầu, xem có nên sử dụng mô hình hiện tại là tốt nhất (tận dụng kiến thức hiện tại) hay thử nghiệm các mô hình khác để có thêm thông tin về chúng (thử nghiệm để biết mô hình nào tốt hơn).

Ưu Điểm:

- Cần ít dữ liệu hơn so với A/B testing để xác định mô hình nào tốt hơn.
- Hiệu quả về dữ liệu và giảm thiểu chi phí "cơ hội".
- An toàn hơn A/B testing vì nếu một mô hình thực sự kém, thuật toán sẽ chọn nó ít hơn.

Nhược Điểm:

- Khó triển khai hơn do cần liên tục cập nhật phản hồi vào thuật toán.
- Chỉ áp dụng cho các trường hợp sử dụng dự đoán trực tuyến.
- Không an toàn như triển khai bóng đèn vì mô hình thách thức vẫn nhận traffic thực tế.

2.3.6. Độ quan trọng của Test Production

Đây là lý do tại sao nó có tính quan trọng:

- Độ chính xác của mô hình: Dữ liệu thực tế có thể khác biệt so với dữ liệu huấn luyện, dẫn đến sự giảm độ chính xác của mô hình khi triển khai. Ví dụ, mô hình phân loại hình ảnh được huấn luyện trên dữ liệu từ máy ảnh chuyên nghiệp, nhưng khi triển khai trên ảnh từ điện thoại di động, độ chính xác có thể giảm đi.
- Hiệu suất của mô hình: Mô hình có thể không đáp ứng được yêu cầu về thời gian thực hoặc tài nguyên khi triển khai. Ví dụ, mô hình dự đoán giá cổ phiếu trong thời gian thực, nhưng thời gian xử lý trễ có thể làm giảm hiệu suất của mô hình.
- Tránh các cuộc tấn công bất lợi: Mô hình có thể dễ bị tấn công, làm giảm đáng kể độ tin cậy khi triển khai. Ví dụ, mô hình xác định gương mặt trong ảnh có thể bị tấn công bằng cách thêm nhiễu đối nghịch vào ảnh.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. AWS, Overfitting là gì, https://aws.amazon.com/vi/what-is/overfitting/#:~:text=Hi%E1%BB%87n%20t%C6%B0%E1%BB%A3ng%20qu%C3%A1%20kh%E1%BB%9Bp%20x%E1%BA%A3y,nguy%C3%AAn%20nh%C3%A2n%2C%20ch%E1%BA%B3ng%20h%E1%BA%A1n%20nh%C6%B0%3A&text=K%C3%ADch%20th%C6%B0%E1%BB%9Bc%20d%E1%BB%AF%20li%E1%BB%87u%20%C4%91%C3%A0o,li%E1%BB%87u%20%C4%91%E1%BA%A7u%20v%C3%A0o%20kh%E1%BA%A3%20thi., truy cập ngày 22/12/2023
2. Funda, Bài 15: Overfitting, https://machinelearningcoban.com/2017/03/04/overfitting/, truy cập ngày 20/12/2023
3. Trí tuệ nhân tạo (02/04/2019), Vấn đề Overfitting & Underfitting trong Machine Learning, https://trituenhantao.io/kien-thuc/van-de-overfitting-underfitting-trong-machine-learning/, truy cập ngày 20/12/2023.
4. Funda (08/01/2017), Bài 6: K-nearest neighbors, https://machinelearningcoban.com/2017/01/08/knn/, truy cập ngày 20/12/2023.
5. Trần Trung Trực (17/10/2020), https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8, https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8 , truy cập ngày 20/12/2023.

Tiếng Anh

6. IBM, What is Overfitting, https://www.ibm.com/topics/overfitting#:~:text=Overfitting%20is%20a%20concept%20in,unseen%20data%2C%20defeating%20its%20purpose., accessed on 10/12/2023.
7. Jason Brownlee (2023), A Tour of Machine Learning Algorithms, https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/, accessed on 10/12/2023.

8. Samadrita G. (2023), How to Compare Machine Learning Models and Algorithms, https://neptune.ai/blog/how-to-compare-machine-learning-models-and-algorithms, accessed on 19/12/2023
9. Ajitesh K. (2020), Gradient Boosting Regression Python Examples, https://vitalflux.com/gradient-boosting-regression-python-examples/, accessed on 19/12/2023

PHỤ LỤC