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

BIỆN NGỌC CANG - 52000827

BÁO CÁO 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

BIỆN NGỌC CANG - 52000827

BÁO CÁO CUỐI KỲ
NHẬP MÔN HỌC MÁY

Người hướng dẫn
TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Lời đầu tiên em xin chân thành cảm ơn quý khoa, nhà trường, thầy cô đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài này. Trong suốt quá trình học tập ở trường, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của thầy cô và bạn bè.

Chúng em xin chân thành cảm ơn thầy Lê Anh Cường, đã truyền đạt cho em những kiến thức quý giá thông qua môn học này. Khoảng thời gian học tập đầy giá trị này giúp chúng em có đủ hành trang cho tương lai sau này. Một lần nữa chúng em xin cảm ơn thầy và kính chúc thầy Cường sức khỏe dồi dào, gặp nhiều may mắn và có một cái Tết 2024 thật sum vầy bên gia đình!

TP. Hồ Chí Minh, ngày 15 tháng 12 năm 2023.

Tác giả

(Ký tên và ghi rõ họ tên)

Biện Ngọc Cang

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 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 15 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Biện Ngọc Cang

TÓM TẮT

Thời đại công nghệ đang diễn ra với tốc độ nhanh chóng, những công nghệ mới đang ngày càng được phát triển và ứng dụng, trong đó có lĩnh vực Trí tuệ nhân tạo và sâu hơn là Học máy. Trong bài báo cáo cuối kì này, sinh viên cần thực hiện 2 phần; phần bài làm cá nhân và phần bài làm nhóm. Trong đó, bài báo cáo dưới đây thuộc về phần bài làm cá nhân.

Trong phần bài làm cá nhân, sinh viên cần tìm hiểu và trình bày về các phương pháp tối ưu hóa trong việc huấn luyện mô hình học máy, countinual learning và test production (Link Github bài làm cá nhân tại đây và bài làm chung tại đây).

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

∇J	Gradient
β	Hệ số giảm trọng số.
η	Tỉ lệ học (Learning Rate)
$E[g^2]$	Trung bình bình phương của gradient
θ	Tham số của mô hình

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

1.1 Gradient Descent

Gradient Descent là một phương pháp quan trọng trong tối ưu hóa mô hình máy học. Gradient Descent sử dụng đạo hàm để xác định hướng và độ lớn của bước di chuyển để giảm thiểu giá trị của hàm mất mát.

1.1.1 Nguyên lí

Nguyên lý cơ bản của Gradient Descent (GD) là một phương pháp tối ưu hóa được sử dụng để điều chỉnh các tham số của mô hình máy học sao cho giảm thiểu hàm mất mát (loss function).

Các bước thực hiện phương pháp Gradient Descent:

Bước 1: Khởi tạo tham số: Khởi tạo các tham số của mô hình, chẳng hạn như trọng số và bias.

Bước 2: Chọn hyperparameters (tham số): Chọn giá trị cho các hyperparameters như learning rate (α), là một tham số quan trọng quyết định độ lớn của bước cập nhật.

Bước 3: Tính Gradient: Tính toán gradient của hàm mất mát theo các tham số.

$\nabla J=[\ \partial J/\partial\theta_1\ ,\ \partial J/\partial\theta_2\ ,\ldots,\ \partial J/\partial\theta_n]$

Bước 4: Cập nhật tham số: Cập nhật các tham số của mô hình bằng cách di chuyển ngược hướng của gradient với một bước có kích thước là learning rate:

$\nabla\theta = \theta - \alpha\cdot\nabla J$

Bước 5: Lặp Lại Quá Trình: Lặp lại quá trình tính toán và cập nhật cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp, giảm gradient đủ nhỏ, hoặc đạt được giá trị mất mát mong muốn).

1.1.2 Ưu-nhược điểm và dữ liệu thích hợp

Ưu Điểm:

- Đơn giản: Dễ hiểu và triển khai.

- Phổ biến: Là một trong những phương pháp tối ưu hóa phổ biến và được sử dụng nhiều.

Nhược Điểm:

- Chọn kích thước bước khó khăn: Việc chọn kích thước bước (learning rate) phù hợp có thể là một thách thức.
- Khả năng hội tụ trong các trường hợp phức tạp chậm: Trong một số trường hợp, có thể yêu cầu nhiều thời gian để hội tụ đến điểm tối ưu.

Dữ liệu phù hợp: Phi tuyến tính, hội tụ nhanh chóng đến giá trị tối ưu, không thay đổi quá nhanh theo thời gian, có nhiều biến đầu vào.

1.2 Batch Gradient Descent

1.2.1 Nguyên lí

Batch Gradient Descent điều chỉnh tham số của mô hình bằng cách sử dụng thông tin từ toàn bộ tập dữ liệu huấn luyện trong mỗi lần cập nhật. Mục tiêu là hội tụ đến giá trị tối ưu của hàm mất mát, nơi mô hình có thể dự đoán tốt trên dữ liệu mới.

Các bước thực hiện Batch Gradient Descent:

Bước 1: Khởi tạo tham số: Khởi tạo các tham số của mô hình, chẳng hạn như trọng số và bias.

Bước 2: Chọn hyperparameters: Chọn giá trị cho các hyperparameters như learning rate (α), là một tham số quan trọng quyết định độ lớn của bước cập nhật.

Bước 3: Lặp qua toàn bộ tập huấn luyện mỗi lần cập nhật tham số. Điều này bao gồm tính toán gradient cho toàn bộ dữ liệu.

Bước 4: Tính Gradient: Tính toán gradient của hàm mất mát theo các tham số.

$\nabla J=[\ \partial J/\partial \theta 1\ ,\ \partial J/\partial \theta 2\ ,\ldots,\ \partial J/\partial \theta n]$

Bước 5: Cập nhật tham số: Cập nhật các tham số của mô hình bằng cách di chuyển ngược hướng của gradient với một bước có kích thước là learning rate:

$\nabla \theta = \theta - \alpha \cdot \nabla J$

Bước 6: Lặp lại quá trình: Lặp lại quá trình bước 3-5 và cập nhật cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp, giảm gradient đủ nhỏ, hoặc đạt được giá trị mất mát mong muốn).

1.2.2 Ưu-nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Độ chính xác cao: Batch Gradient Descent thường dẫn đến độ chính xác cao hơn vì nó sử dụng toàn bộ tập dữ liệu để cập nhật tham số.
- Hội tụ ổn định: Do sử dụng toàn bộ tập dữ liệu, Batch Gradient Descent có thể hội tụ một cách ổn định đến giá trị tối ưu.
- Hiệu quả trên dữ liệu nhỏ: Trong trường hợp dữ liệu nhỏ có thể fit vào bộ nhớ, Batch Gradient Descent có thể hoạt động hiệu quả.
- Được ưa chuộng trong học sâu: Trong các mô hình học sâu, Batch Gradient Descent thường được ưa chuộng do có thể tận dụng khả năng tính toán của GPU để xử lý toàn bộ tập dữ liệu.

Nhược Điểm:

- Tính tính toán cao: Batch Gradient Descent yêu cầu tính toán đạo hàm của toàn bộ tập dữ liệu, điều này có thể làm chậm quá trình đào tạo đối với dữ liệu lớn.
- Không hiệu quả cho dữ liệu lớn: Khi tập dữ liệu không thể fit vào bộ nhớ, Batch Gradient Descent trở nên không hiệu quả vì đòi hỏi nhiều tài nguyên tính toán.
- Khả năng mất thông tin với dữ liệu động: Nếu dữ liệu thay đổi theo thời gian, Batch Gradient Descent có thể mất đi thông tin về biến động nếu không được cập nhật thường xuyên..
- Yêu cầu bộ nhớ lớn: Cần phải lưu toàn bộ tập dữ liệu trong bộ nhớ, điều này có thể trở thành vấn đề khi làm việc với dữ liệu lớn.

Dữ liệu phù hợp đối với phương pháp: dữ liệu nhỏ và sạch, không nhiễu và không chứa quá nhiều giá trị ngoại lệ, không hoặc ít thay đổi theo thời gian.

1.3 Stochastic Gradient Descent (SGD)

1.3.1 Nguyên lí

Mục tiêu của SGD là tối ưu hóa hàm mất mát, biểu diễn sự chênh lệch giữa dự đoán của mô hình và giá trị thực tế trên tập dữ liệu huấn luyện. Thay vì tính gradient trên toàn bộ tập dữ liệu, SGD giảm độ phức tạp tính toán bằng cách chỉ sử dụng một lượng nhỏ dữ liệu trong mỗi lần cập nhật. Quá trình này được lặp lại qua nhiều mini-batch cho đến khi đạt được điều kiện dừng.

Các bước thực hiện SGD:

Bước 1: Khởi tạo tham số: Khởi tạo các tham số của mô hình, bao gồm trọng số và độ chệch (bias).

Bước 2: Chọn kích thước mẫu (batch size): Chọn một số lượng mẫu ngẫu nhiên từ tập dữ liệu để tạo thành một batch. Kích thước batch là một tham số quan trọng trong SGD.

Bước 3: Tính giá trị dự đoán: Sử dụng các tham số hiện tại của mô hình để tính toán giá trị dự đoán cho mỗi mẫu trong batch.

Bước 4: Tính toán hàm mất mát: Tính toán giá trị của hàm mất mát, đo lường sự chênh lệch giữa giá trị dự đoán và giá trị thực tế.

Bước 5: Tính gradient của hàm mất mát: Tính toán gradient của hàm mất mát đối với từng tham số. Điều này đại diện cho hướng và độ lớn cần điều chỉnh các tham số để giảm mất mát.

Bước 6: Cập nhật tham số: Sử dụng gradient tính được để cập nhật các tham số của mô hình. Công thức cập nhật thường sử dụng một tỷ lệ học (learning rate) để kiểm soát kích thước của bước cập nhật.

Tham số mới=Tham số cũ −Tỉ lệ học×Gradient

Bước 7: Lặp lại quá trình: Lặp lại các bước 2-6 cho đến khi đạt được điều kiện dừng hoặc một số lượng vòng lặp đã đủ.

1.3.2 Ưu-nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Hiệu quả cho dữ liệu lớn: SGD thích hợp cho việc đào tạo mô hình trên dữ liệu lớn, vì nó chỉ yêu cầu một lượng nhỏ dữ liệu trong mỗi lần cập nhật.
- Khả năng đối phó với dữ liệu nhiễu: Do việc chọn ngẫu nhiên mini-batch, SGD có khả năng tránh được nhiễu và giúp mô hình chống lại điểm cực tiểu địa phương.
- Tích hợp tốt với học sâu: Trong học sâu, SGD được sử dụng phổ biến giúp đào tạo mô hình nhanh chóng.
- Cập nhật thường xuyên: Việc cập nhật tham số sau mỗi mini-batch giúp SGD có thể hội tụ nhanh chóng và thích hợp cho việc đào tạo online.
- Ngốn ít tài nguyên tính toán: Vì chỉ sử dụng một lượng nhỏ dữ liệu, SGD ngốn ít tài nguyên tính toán hơn so với Batch Gradient Descent.

Nhược Điểm:

- Khả năng dao động: SGD có thể dao động quanh điểm tối ưu do sự ngẫu nhiên trong việc chọn mini-batch.
- Không đảm bảo hội tụ đến giá trị tối ưu toàn cục: Do tính ngẫu nhiên, SGD không đảm bảo hội tụ đến giá trị tối ưu toàn cục, nhưng thay vào đó có thể dẫn đến điểm cực tiểu địa phương.
- Yêu cầu chọn learning rate thích hợp: Cần phải chọn learning rate phù hợp để đảm bảo sự hội tụ và tránh trường hợp overshooting hoặc undershooting.
- Dễ bị ảnh hưởng bởi nhiễu: Do sự ngẫu nhiên trong việc chọn mini-batch, SGD có thể bị ảnh hưởng bởi nhiễu, đặc biệt là trên dữ liệu nhỏ hoặc có nhiễu.
- Không hiệu quả đối với dữ liệu nhỏ: Trong trường hợp dữ liệu nhỏ, SGD có thể không hiệu quả do độ dao động cao và không tận dụng được các lợi ích của Batch Gradient Descent.

Dữ liệu thích hợp: lớn, dữ liệu động, có thể thay đổi theo thời gian, có thể có nhiễu, dữ liệu yêu cầu huấn luyện nhanh hoặc huấn luyện online liên tục được cập nhật với dữ liệu mới.

1.4 RMSprop (Root Mean Square Propagation)

1.4.1 Nguyên lí

Nguyên lý cơ bản của RMSprop là điều chỉnh tỷ lệ học (learning rate) cho từng tham số của mô hình dựa trên lịch sử của các gradient gần đây. Cụ thể, RMSprop sử dụng trung bình bình phương của các gradient để thay đổi tỷ lệ học.

Các bước thực hiện RMSprop:

Bước 1: Khởi tạo các tham số: Khởi tạo các tham số của mô hình, bao gồm các trọng số (θ) và các tham số khác.

Bước 2: Khởi tạo các biến cho RMSprop: Khởi tạo biến $E[g^2]$ lưu trữ trung bình bình phương của gradient.

Bước 3: Thiết lập các hyperparameters: Đặt giá trị cho các siêu tham số như tỷ lệ học (η) và hệ số giảm trọng số (β).

Bước 4: Lặp lại qua từng mini-batch: Tính gradient (∇J) của hàm mất mát đối với các tham số.

Bước 5: Cập nhật $E[g^2]$ Sử dụng công thức sau để cập nhật trung bình bình phương của gradient: $E[g^2]=\beta E[g^2]+(1-\beta)(\nabla J)^2$

Bước 6: Cập nhật tham sốSử dụng công thức sau để cập nhật các tham số:

$\theta=\theta - \eta /(E[g^2]+\epsilon).\nabla J$

ϵ lúc này là một giá trị nhỏ để tránh chia cho 0.

Bước 7: Lặp lại quá trình: Lặp lại các bước trên với các mini-batch khác hoặc toàn bộ dữ liệu.

1.4.2 Ưu nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Điều chỉnh tỷ lệ học tự động: RMSprop có khả năng tự động điều chỉnh tỷ lệ học cho từng tham số của mô hình dựa trên lịch sử của gradient. Điều này giúp nó thích ứng tốt với các tình huống trong đó gradient thay đổi đáng kể.
- Hiệu suất tốt trong các bài toán không đồng nhất: RMSprop thường hoạt động hiệu quả hơn so với các thuật toán tối ưu hóa khác trong các bài toán có hàm mất mát không đồng nhất hoặc có các tham số có độ biến động lớn.
- Không yêu cầu tham số điều chỉnh nhiều: RMSprop ít yêu cầu sự điều chỉnh tham số hơn so với một số thuật toán khác

Nhược điểm:

- Nguy cơ đột ngột giảm tỷ lệ học: Trong một số trường hợp, RMSprop có thể dẫn đến nguy cơ tỷ lệ học giảm đột ngột quá nhanh, đặc biệt khi giá trị của $E[g^2]$ là rất lớn. Điều này có thể làm chậm quá trình học.
- Không hiệu quả trên tất cả các bài toán: Mặc dù RMSprop thích ứng tốt với nhiều tình huống, nhưng nó không phải là giải pháp hoàn hảo cho mọi bài toán. Có những trường hợp nó không hiệu quả hoặc thậm chí dẫn đến hiện tượng không ổn định.
- Tùy chỉnh các siêu tham số: Mặc dù ít cần điều chỉnh hơn so với một số thuật toán khác, nhưng vẫn có một số siêu tham số cần được điều chỉnh, như tỷ lệ học (η) và hệ số giảm trọng số (β).

Dữ liệu thích hợp: thường là những bài toán mà hàm mất mát có tính chất không đồng nhất, nghĩa là gradient thay đổi đáng kể tại các điểm khác nhau của không gian tham số; dữ liệu trong các bài toán học tăng cường; xử lí ảnh và thị giác máy tính.

1.5 Momentum

1.5.1 Nguyên lí

Nguyên lý cơ bản của thuật toán Momentum là sử dụng một đối tượng giữ đà (momentum) để theo dõi hướng và tốc độ của quá trình tối ưu hóa.

Bước 1: Khởi tạo các tham số: Khởi tạo các tham số của mô hình, bao gồm các trọng số (θ) và các tham số khác.

Bước 2: Khởi tạo moment (động lượng): Khởi tạo moment bậc nhất (m) bằng 0 hoặc giá trị khác nhau tùy thuộc vào chiến lược khởi tạo.

Bước 3: Thiết lập tỷ lệ học và hệ số giảm trọng số: Đặt giá trị cho tỷ lệ học (η) và hệ số giảm trọng số (β).

Bước 4: Lặp qua từng mini-batch hoặc toàn bộ dữ liệu: Tính gradient (∇J) của hàm mất mát đối với các tham số.

Bước 5: Cập nhật moment: Sử dụng công thức sau để cập nhật moment: $m=\beta m+(1-\beta)\nabla J$

Bước 6: Cập nhật tham số: Sử dụng công thức sau để cập nhật các tham số: $\theta=\theta-\eta m$

Bước 7: Lặp lại quá trình: Lặp lại các bước trên với các mini-batch khác hoặc toàn bộ dữ liệu.

1.5.2 Ưu nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Tăng tốc độ học: Momentum giúp tăng tốc độ học của mô hình bằng cách giữ đà (momentum) từ các bước trước đó, giúp vượt qua sự đánh mất độ đạo hàm và giảm bớt độ dao động trong quá trình tối ưu hóa.
- Khả năng vượt qua điểm cực tiểu cục bộ: Đối với các bài toán có nhiều điểm cực tiểu cục bộ, Momentum có khả năng vượt qua chúng và tiếp tục tìm kiếm trong không gian tham số.
- Giảm dao động: Momentum giúp giảm độ dao động của quá trình tối ưu hóa, giúp mô hình hội tụ nhanh hơn.

Nhược điểm:

- Quá trình học có thể quá nhanh: Trong một số trường hợp, Momentum có thể làm cho quá trình học tiến triển quá nhanh và vượt qua điểm tối ưu.
- Khả năng lặp đi lặp lại: Nếu tỷ lệ học quá lớn, mô hình có thể lặp đi lặp lại quanh điểm tối ưu hoặc không hội tụ.
- Yêu cầu lựa chọn tham số tốt: Cần lựa chọn tham số β một cách cẩn thận để đảm bảo sự ổn định và hiệu suất tốt của thuật toán.

Dữ liệu phù hợp cho Momentum là các dạng dữ liệu có đặc tính không đồng nhất, dữ liệu cho bài toán có độ biến động lớn.

1.6 Adam (Adaptive Moment Estimation)

1.6.1 Nguyên lí

Adam kết hợp hai ý tưởng chính từ RMSprop và Momentum để tạo ra một phương pháp hiệu quả cho việc tối ưu hóa hàm mất mát.

Các bước thực hiện Adam:

Bước 1: Khởi tạo các tham số: Khởi tạo các tham số của mô hình, bao gồm các trọng số (θ) và các tham số khác.

Bước 2: Khởi tạo moment bậc nhất và bậc hai: Khởi tạo moment bậc nhất (m) và moment bậc hai (v) bằng 0 hoặc giá trị khác nhau tùy thuộc vào chiến lược khởi tạo.

Bước 3: Thiết lập tỷ lệ học và các hệ số giảm trọng số: Đặt giá trị cho tỷ lệ học (η), hệ số giảm trọng số cho moment bậc nhất (β_1), và hệ số giảm trọng số cho moment bậc hai (β_2).

Bước 4: Khởi tạo biến đếm:

Khởi tạo biến đếm (t) bằng 0.

Bước 5: Lặp qua từng mini-batch hoặc toàn bộ dữ liệu:

Tính gradient (∇J) của hàm mất mát đối với các tham số.

Bước 6: Cập nhật moment bậc nhất và bậc hai: Sử dụng công thức sau để cập nhật moment bậc nhất và moment bậc hai:

$m=\beta_1 m+(1-\beta_1)\nabla J$

$v=\beta_2 v+(1-\beta_2)(\nabla J)^2$

Bước 7: Hiệu chỉnh độ chệch (bias correction):

Thực hiện bước hiệu chỉnh độ chệch để giảm ảnh hưởng của giá trị khởi tạo 0 cho $\hat{m}$ và $\hat{v}$

$\hat{m}=m/(1-\beta_1^t)$

$\hat{v}=v/(1-\beta_2^t)$

Bước 8: Cập nhật tham số:

Sử dụng công thức sau để cập nhật các tham số

$\theta = \theta - \eta/(\sqrt{\hat{v}}+\epsilon\hat{m})$

Trong đó:

ϵ là một giá trị nhỏ để tránh chia cho 0.

Bước 9: Cập nhật biến đếm:

Tăng giá trị biến đếm (t) lên 1: 1t=t+1.

Bước 10: Lặp lại quá trình:

Lặp lại các bước trên với các mini-batch khác hoặc toàn bộ dữ liệu.

1.6.2 Ưu nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Hiệu suất và linh hoạt: Adam thường hiệu quả và linh hoạt trên nhiều loại bài toán học máy, đặc biệt là trong các mô hình sâu và phức tạp.
- Tự điều chỉnh tỷ lệ học: Adam tự điều chỉnh tỷ lệ học (learning rate) cho từng tham số dựa trên lịch sử của gradient, giúp nó phù hợp với nhiều tình huống và giảm nguy cơ tỷ lệ học giảm quá nhanh.
- Hoạt động tốt với dữ liệu lớn: Adam thường hoạt động tốt với các tập dữ liệu lớn và có đặc tính không đồng nhất.

Nhược điểm:

- Yêu cầu đối số tinh chỉnh: Adam có nhiều đối số cần được tinh chỉnh, bao gồm β_1, β_2, và ϵ. Sự lựa chọn không tốt có thể dẫn đến hiện tượng không ổn định hoặc quá mức tinh chỉnh.
- Nhạy cảm với outlier: Adam có thể nhạy cảm với các giá trị gradient ngoại lệ (outlier) và có thể bị ảnh hưởng nếu các gradient có biên độ lớn.

Dữ liệu thích hợp: lớn và không đồng nhất, nơi tỷ lệ học có thể được điều chỉnh linh hoạt cho từng tham số; dữ liệu cho các bài toán tối ưu hóa phức tạp.

1.7 Adagrad (Adaptive Gradient Algorithm)

Nguyên lí

Nguyên lý cơ bản của Adagrad là điều chỉnh tỷ lệ học dựa trên lịch sử của gradient cho mỗi tham số.

Các bước thực hiện Adagrad:

Bước 1: Khởi tạo các tham số: Khởi tạo các tham số của mô hình, bao gồm các trọng số (θ) và các tham số khác.

Bước 2: Khởi tạo ma trận tổng bình phương gradient: Khởi tạo ma trận tổng bình phương gradient (G) bằng 0 hoặc giá trị khác nhau tùy thuộc vào chiến lược khởi tạo.

Bước 3: Thiết lập tỷ lệ học và giá trị epsilon: Đặt giá trị cho tỷ lệ học (η) và giá trị epsilon (ϵ).

Bước 4: Lặp qua từng mini-batch hoặc toàn bộ dữ liệu: Tính gradient (∇J) của hàm mất mát đối với từng tham số của mô hình.

Bước 5: Cập nhật tổng bình phương gradient: Sử dụng công thức sau để cập nhật ma trận tổng bình phương gradient: $G=G+(\nabla J)^2$

Bước 6: Cập nhật tham số: Sử dụng công thức sau để cập nhật từng tham số của mô hình:

$\theta=\theta- \eta /(G+\epsilon \nabla J)$

Bước 7: Lặp lại quá trình: Lặp lại các bước trên với các mini-batch khác hoặc toàn bộ dữ liệu.

1.7.1 Ưu nhược điểm và dữ liệu thích hợp

Ưu điểm:

- Tự điều chỉnh tỷ lệ học: Adagrad tự điều chỉnh tỷ lệ học cho từng tham số dựa trên lịch sử của gradient. Điều này giúp phù hợp với các tình huống trong đó các tham số có độ biến động lớn.
- Không yêu cầu lựa chọn tỷ lệ học ban đầu: Adagrad không yêu cầu người dùng phải chọn tỷ lệ học ban đầu, mà tỷ lệ này được tự động điều chỉnh dựa trên thông tin lịch sử.
- Phù hợp với các bài toán có độ chuyển động biến động lớn: Adagrad thường hoạt động tốt trên các bài toán mà các tham số có độ biến động lớn, ví dụ như các bài toán học máy sâu.

Nhược điểm:

- Độ giảm tỷ lệ học quá nhanh: Một trong nhược điểm lớn của Adagrad là sau một số bước lặp, ma trận tổng bình phương gradient có thể trở nên rất lớn, dẫn đến độ giảm tỷ lệ học nhanh chóng. Điều này có thể làm cho quá trình học trở nên chậm lại.
- Không phù hợp cho các tham số thường xuyên cập nhật: Trong một số trường hợp, khi có các tham số có gradient thường xuyên thay đổi, Adagrad có thể làm giảm quá mức tỷ lệ học, làm chậm quá trình học.

Dữ liệu thích hợp: dữ liệu với các tham số có độ biến động lớn và thường xuyên thay đổi; dữ liệu thưa, nơi một số thành phần của gradient có giá trị lớn, trong khi nhiều thành phần khác có giá trị gần 0.

1.8 Adadelta

1.8.1 Nguyên lí

Adadelta được thiết kế để giảm nhược điểm của Adagrad liên quan đến giảm tỷ lệ học quá nhanh do tích lũy bình phương gradient. Nguyên lý cơ bản của Adadelta là sử dụng trung bình trượt có trọng số của bình phương gradient để điều chỉnh tỷ lệ học

Các bước thực hiện Adadelta:

Bước 1: Khởi tạo các tham số: Khởi tạo các tham số của mô hình, bao gồm các trọng số (θ) và các tham số khác.

Bước 2: Khởi tạo ma trận trung bình trượt bình phương gradient ($E[g^2]$) và ma trận trung bình trượt bình phương của thay đổi tham số ($E[\Delta\theta^2]$) bằng 0 hoặc giá trị khác nhau tùy thuộc vào chiến lược khởi tạo.

Bước 3: Thiết lập hệ số giảm trọng số (ρ): Đặt giá trị cho hệ số giảm trọng số (ρ).

Bước 4: Lặp qua từng mini-batch hoặc toàn bộ dữ liệu: Tính gradient (∇J) của hàm mất mát đối với từng tham số của mô hình.

Bước 5: Cập nhật ma trận trung bình trượt bình phương gradient ($E[g^2]$):

Sử dụng công thức sau để cập nhật $E[g^2]$:

$E[g^2]=\rho E[g^2]+(1-\rho)(\nabla J)^2$

Bước 6: Tính toán thay đổi tham số (Δθ):

Tính toán thay đổi tham số (Δθ) bằng tỷ lệ của tham số gradient và căn bậc hai của ma trận trung bình trượt bình phương của thay đổi tham số:

$$\Delta\theta = -\frac{\sqrt{E[g2]+\epsilon}}{\sqrt{E[\Delta\theta 2]+\epsilon}}\nabla J$$

Trong đó ϵ là một giá trị nhỏ để tránh chia cho 0.

Bước 7: Cập nhật tham số (θ): Sử dụng công thức: $\theta=\theta+\Delta\theta$

Bước 8: Cập nhật ma trận trung bình trượt bình phương của thay đổi tham số ($E[\Delta\theta^2]$): Sử dụng công thức $E[\Delta\theta^2]$: $E[\Delta\theta^2]=\rho E[\Delta\theta^2]+(1-\rho)(\Delta\theta)^2$

Bước 9: Lặp lại các bước trên với các mini-batch khác hoặc toàn bộ dữ liệu.

1.8.2 Ưu nhược điểm và dữ liệu thích hợp

Ưu điểm

- Tự điều chỉnh tỷ lệ học: Adadelta tự điều chỉnh tỷ lệ học cho từng tham số một cách tự động dựa trên lịch sử của gradient, giúp tránh được vấn đề giảm tỷ lệ học quá nhanh như trong Adagrad.
- Không yêu cầu lựa chọn tỷ lệ học ban đầu: Khác với một số thuật toán khác, Adadelta không yêu cầu người dùng phải lựa chọn tỷ lệ học ban đầu.
- Ít tham số cần tinh chỉnh: Adadelta có ít tham số cần tinh chỉnh so với một số thuật toán khác, giúp giảm công đoạn lựa chọn tham số.

Nhược điểm:

- Không hoạt động tốt trên mọi bài toán: Mặc dù Adadelta hoạt động tốt trên nhiều loại bài toán, nhưng không phải lúc nào cũng là lựa chọn tốt nhất. Có những trường hợp nơi các thuật toán khác như Adam có thể mang lại hiệu suất tốt hơn.
- Không thích hợp cho bài toán có tham số thường xuyên thay đổi: Adadelta có thể không hiệu quả trên các bài toán mà tham số thường xuyên thay đổi đột ngột, do quá trình điều chỉnh tỷ lệ học của nó cũng diễn ra tương đối chậm.

Dữ liệu thích hợp: Adadelta thường hoạt động tốt trên các tập dữ liệu không đồng nhất và những tập dữ liệu ít biến động trong thời gian ngắn.

1.9 So sánh các phương pháp

Phương Pháp	Độ Hiệu Quả	Bộ Dữ Liệu Phù Hợp	Ưu Điểm	Nhược Điểm
GD	Trung bình	Lớn	Dễ hiểu, hội tụ ổn định	Chậm trên dữ liệu lớn, nhạy cảm với nhiễu

Phương Pháp	Độ Hiệu Quả	Bộ Dữ Liệu Phù Hợp	Ưu Điểm	Nhược Điểm
BGD	Thấp	Nhỏ hoặc Lớn	Hội tụ chính xác, ít nhạy cảm	Chậm trên dữ liệu lớn, có thể rơi vào điểm tối ưu cục bộ
SGD	Cao	Nhỏ	Hội tụ nhanh, phù hợp với dữ liệu lớn	Không ổn định, nhảy số lớn, nhạy cảm với nhiễu
RMSprop	Cao	Mọi loại	Hiệu quả trên dữ liệu không đồng nhất	Cần lựa chọn tham số, nhạy cảm với outlier
Momentum	Cao	Mọi loại	Giảm ổn định của GD, vượt qua điểm tối ưu cục bộ	Cần lựa chọn tham số, nhạy cảm với outlier
Adam	Cao	Mọi loại	Hiệu quả và linh hoạt, tự điều chỉnh tỷ lệ học	Cần lựa chọn tham số, yêu cầu bộ nhớ lớn
Adagrad	Trung bình	Mọi loại	Tự điều chỉnh tỷ lệ học, ít tham số cần tinh chỉnh	Độ giảm tỷ lệ học quá nhanh, không phù hợp cho dữ liệu thưa

Bảng 1.1 Bảng so sánh các phương pháp tối ưu hóa mô hình học máy

CHƯƠNG 2. CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual learning – Học liên tục

2.1.1 Khái niệm

Continual Learning (học liên tục) là một lĩnh vực trong Machine Learning (Máy học) mà mô hình liên tục học hỏi và thích ứng với dữ liệu mới trong khi vẫn duy trì kiến thức và kỹ năng đã có.

2.1.2 Đặc điểm

Giảm thiểu Forgetting và Overfitting: Mô hình cần giảm thiểu việc quên kiến thức từ quá khứ khi học từ dữ liệu mới và tránh tình trạng quá khớp trên dữ liệu mới, đồng thời duy trì khả năng tổng quát từ kiến thức đã học.

Adaptability (Khả năng thích ứng): Mô hình cần có khả năng thích ứng với sự thay đổi của dữ liệu mà không làm mất đi khả năng dự đoán trên các dữ liệu trước đó.

Plasticity (Khả năng định hình): Khả năng điều chỉnh cấu trúc của mô hình để nắm bắt thông tin mới từ dữ liệu.

Transfer Learning (Học chuyển giao): Sử dụng những kiến thức đã học được từ một tác vụ để giúp việc học tác vụ mới.

Memory (Bộ nhớ): Mô hình cần có khả năng lưu trữ thông tin quan trọng từ quá khứ để sử dụng trong tương lai.

Task-Incremental Learning (Học tăng cường theo tác vụ): Mô hình có khả năng học từng tác vụ một, thường thông qua việc thêm dữ liệu từ tác vụ mới mà không làm ảnh hưởng đến hiệu suất trên các tác vụ cũ.

Catastrophic Forgetting (Quên thảm họa): Tránh tình trạng quên nhanh chóng kiến thức từ tác vụ trước khi học tác vụ mới.

Regularization: Sử dụng các kỹ thuật regularization để kiểm soát việc quá mức linh hoạt của mô hình.

Dynamic Architectures (Kiến trúc động): Có khả năng thay đổi kiến trúc của mô hình tùy thuộc vào yêu cầu của dữ liệu.

Online Learning (Học trực tuyến): Mô hình có thể học ngay lập tức khi có dữ liệu mới thay vì phải huấn luyện lại toàn bộ mô hình.

2.1.3 Áp dụng vào việc xây dựng giải pháp học máy

Lợi ích của việc áp dụng Continual Learning vào việc xây dựng giải pháp học máy:

- Tính linh hoạt: Continual Learning giúp mô hình linh hoạt hóa và thích ứng với sự thay đổi liên tục trong dữ liệu mà không cần phải retrain từ đầu.
- Tiết kiệm tài nguyên: Với khả năng học từng phần dữ liệu một, mô hình không cần phải tiêu tốn tài nguyên lớn để huấn luyện lại toàn bộ mô hình khi có dữ liệu mới.
- Giảm thiểu hiện tượng Forgetting: Continual Learning giúp giảm thiểu hiện tượng quên nhanh chóng kiến thức từ tác vụ trước khi học tác vụ mới.
- Tận dụng kiến thức cũ: Mô hình có thể tận dụng kiến thức đã học từ tác vụ cũ để giúp việc học tác vụ mới hiệu quả hơn.
- Hỗ trợ học chuyển giao: Continual Learning tạo cơ hội cho học chuyển giao, nơi mô hình có thể sử dụng kiến thức từ một lĩnh vực để hỗ trợ việc giải quyết vấn đề trong lĩnh vực khác.
- Tích hợp dữ liệu mới một cách hiệu quả: Dữ liệu mới có thể được tích hợp vào mô hình một cách hiệu quả mà không làm ảnh hưởng đến hiệu suất trên dữ liệu cũ.
- Ứng dụng trong hệ thống thời gian thực: Các ứng dụng thời gian thực, nơi dữ liệu liên tục đến, có thể tận dụng lợi ích của Continual Learning để cập nhật mô hình một cách liên tục.

- Học tự động: Continual Learning làm cho quá trình học trở nên tự động và có thể duy trì hiệu suất của mô hình theo thời gian.
- Giảm thiểu rủi ro do thay đổi dữ liệu: Mô hình có khả năng thích ứng với sự đa dạng và thay đổi trong dữ liệu, giảm thiểu rủi ro do thay đổi phân phối.

Một số cách để áp dụng Continual Learning vào việc xây dựng mô hình học máy:

- Xây dựng kiến trúc động: Xây dựng một kiến trúc mô hình có khả năng thay đổi cấu trúc của nó để phản ánh sự thay đổi trong dữ liệu. Các lớp hoặc nơ-ron có thể được thêm hoặc loại bỏ tùy thuộc vào yêu cầu của tập dữ liệu mới.
- Học tích hợp dữ liệu mới: Thêm dữ liệu mới một cách liên tục và sử dụng thuật toán học tích hợp để cập nhật mô hình. Điều này giúp mô hình học từ dữ liệu mới mà không làm ảnh hưởng đến hiệu suất trên dữ liệu cũ.
- Sử dụng Elastic Weight Consolidation (EWC): EWC là một phương pháp regularization giúp giữ cho các trọng số quan trọng của mô hình được giữ nguyên khi học từ dữ liệu mới, giảm thiểu hiện tượng quên.
- Transfer learning: Sử dụng Transfer Learning để tận dụng kiến thức đã học từ mô hình trước đó và áp dụng vào mô hình mới cho các tác vụ tương tự hoặc có liên quan.
- Lưu trữ bộ nhớ ngoại vi: Xây dựng một bộ nhớ ngoại vi để lưu trữ các mẫu dữ liệu quan trọng từ quá khứ. Khi cần thiết, mô hình có thể sử dụng lại thông tin từ bộ nhớ này.
- Tích hợp Online Learning: Thực hiện học trực tuyến để mô hình có thể ngay lập tức cập nhật khi có dữ liệu mới thay vì phải retrain toàn bộ.
- Đánh giá định kỳ và điều chỉnh: Thực hiện đánh giá định kỳ trên tập dữ liệu mới và điều chỉnh mô hình nếu cần thiết để giữ cho hiệu suất cao.

Các bước áp dụng Continual Learning:

Bước 1: Chia dữ liệu: Phân chia dữ liệu thành các tập hợp nhỏ hoặc đợt dữ liệu mới một cách có tổ chức. Điều này giúp định rõ khi nào và làm thế nào dữ liệu mới sẽ được áp dụng vào mô hình.

Bước 2: Xây dựng mô hình ban đầu: Huấn luyện một mô hình ban đầu trên tập dữ liệu đầu tiên. Mô hình này sẽ được sử dụng như là một nền tảng để tích hợp kiến thức từ dữ liệu mới.

Bước 3: Lưu trữ trọng số quan trọng: Lưu trữ trọng số quan trọng của mô hình ban đầu bằng cách sử dụng các kỹ thuật như Elastic Weight Consolidation (EWC) để giảm thiểu hiện tượng quên.

Bước 4: Áp dụng dữ liệu mới: Áp dụng dữ liệu mới vào mô hình. Cập nhật trọng số của mô hình để học từ dữ liệu mới mà không làm ảnh hưởng đến hiệu suất trên dữ liệu cũ.

Bước 5: Kiểm tra hiệu suất: Đánh giá hiệu suất của mô hình trên cả dữ liệu cũ và dữ liệu mới để đảm bảo rằng nó vẫn duy trì khả năng học và không gặp hiện tượng quên.

Bước 6: Làm mềm trọng số (Distillation): Sử dụng kỹ thuật làm mềm trọng số để truyền đạt kiến thức từ mô hình cũ sang mô hình mới, giúp cải thiện sự linh hoạt và thích ứng.

Bước 7: Tích hợp Transfer Learning: Sử dụng Transfer Learning để tận dụng kiến thức đã học từ mô hình trước đó và áp dụng vào mô hình mới cho các tác vụ tương tự hoặc có liên quan.

Bước 8: Tối ưu hóa độ linh hoạt: Điều chỉnh các siêu tham số của mô hình để tối ưu hóa độ linh hoạt và khả năng thích ứng, như giảm thiểu hiện tượng quên và duy trì kiến thức quan trọng.

Bước 9: Lặp lại quy trình: Lặp lại quy trình trên với mỗi đợt dữ liệu mới để mô hình liên tục cập nhật và học từ môi trường thay đổi.

Bước 10: Kiểm tra định kỳ và điều chỉnh: Thực hiện kiểm tra định kỳ và điều chỉnh mô hình nếu cần thiết để đảm bảo duy trì hiệu suất và khả năng học.

Bước 11: Tích hợp bộ nhớ ngoại vi: Xây dựng và tích hợp bộ nhớ ngoại vi để lưu trữ các mẫu dữ liệu quan trọng từ quá khứ và sử dụng chúng khi cần thiết.

2.2 Test Production – Kiểm tra sản xuất

2.2.1 Khái niệm

Test Production là quá trình triển khai một mô hình máy học vào môi trường thực tế để thử nghiệm và đánh giá hiệu suất của mô hình trên dữ liệu thực tế từ production (sản xuất).

2.2.2 Tiêu chí

Các tiêu chí của Test Production:

- Tính tích hợp hệ thống: Mô hình cần được tích hợp một cách hoàn chỉnh và hiệu quả vào hệ thống sản xuất mà không gây ra lỗi hoặc gián đoạn.
- Tính đồng nhất và ổn định: Mô hình cần duy trì tính đồng nhất và ổn định khi chạy trên dữ liệu thực tế, tránh tình trạng quá mức tinh chỉnh cho dữ liệu kiểm thử.
- Kiểm tra hiệu suất: Đánh giá hiệu suất của mô hình trên dữ liệu thực tế để đảm bảo rằng nó đáp ứng các yêu cầu về độ chính xác và hiệu suất.
- Quản lý vấn đề thực tế: Test Production phải xác định và giải quyết các vấn đề xuất hiện trong môi trường sản xuất thực tế, như biến động của dữ liệu hoặc sự thay đổi trong môi trường.
- Tinh chỉnh linh hoạt: Mô hình cần có khả năng tinh chỉnh linh hoạt để đáp ứng các thay đổi trong dữ liệu và yêu cầu môi trường.
- Quản lý tài nguyên: Kiểm tra xem mô hình có thể hoạt động mà không gây ra vấn đề về tài nguyên hay không, đặc biệt là đối với các mô hình yêu cầu tài nguyên lớn.
- Kiểm tra an toàn và bảo mật: Test Production phải kiểm tra các khía cạnh liên quan đến an toàn và bảo mật của mô hình trong môi trường sản xuất.

- Quản lý cập nhật và đồng bộ hóa: Đảm bảo rằng quá trình cập nhật mô hình và đồng bộ hóa với dữ liệu mới diễn ra một cách hiệu quả và không gây ra sự gián đoạn.
- Giảm rủi ro Forgetting: Kiểm tra và giảm thiểu rủi ro quên thông tin quan trọng đã học từ dữ liệu cũ.
- Kiểm tra tích hợp bộ nhớ ngoại vi (External Memory): Đối với các mô hình sử dụng bộ nhớ ngoại vi, Test Production phải đảm bảo tích hợp và sử dụng nó một cách hiệu quả.

2.2.3 Áp dụng vào việc xây dựng giải pháp học máy

Các bước áp dụng Test Production vào việc xây dựng giải pháp học máy:

Bước 1: Triển khai mô hình: Đưa mô hình từ môi trường phát triển (development environment) hoặc môi trường kiểm thử (test environment) vào môi trường sản xuất.

Bước 2: Chạy mô hình trên dữ liệu thực tế: Mô hình được chạy trên dữ liệu thực tế từ production để xem xét cách nó thích ứng và hoạt động trong môi trường thực tế.

Bước 3: Đánh giá hiệu suất: Đánh giá hiệu suất của mô hình trên dữ liệu thực tế, bao gồm độ chính xác, độ tin cậy, và các thước đo hiệu suất khác.

Bước 4: Kiểm tra độ ổn định và đồng nhất: Đảm bảo rằng mô hình là ổn định và đồng nhất trong môi trường production, tránh các vấn đề như quá mức tinh chỉnh trên dữ liệu kiểm thử.

Bước 5: Giải quyết vấn đề và tối ưu hóa: Nếu phát hiện vấn đề, thực hiện các điều chỉnh cần thiết để cải thiện hiệu suất và đảm bảo mô hình hoạt động hiệu quả trong môi trường production.

Bước 6: Quản lý đồng bộ hóa và cập nhật: Quản lý việc đồng bộ hóa và cập nhật mô hình khi có sự thay đổi trong dữ liệu production hoặc khi mô hình cần được cập nhật.

Link Github bài làm cá nhân tại đây

Link Github bài làm chung nhóm tại đây