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

PHẠM HOÀN PHÚC - 52000704

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

PHẠM HOÀN PHÚC - 52000704

BÁO CÁO CUỐI KỲ
NHẬP MÔN HỌC MÁY

THÀNH PHỐ HỒ CHÍ MINH, NĂM 202

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. 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 22 tháng 12 năm 2023.

Tác giả

Phạm Hoàn Phúc

MỤC LỤC

CHƯƠNG 1. MỞ ĐẦU

Bài báo cáo này sẽ gồm các phần như sau

- Trình bày về nội dung, so sánh, đánh giá các phương pháp Optimizer trong học máy. (làm cá nhân)

- Trình bày về các nội dung tìm hiểu về Continual Learning và Test Production. (làm cá nhân)

CHƯƠNG 2. CÁC PHƯƠNG PHÁP OPTIMIZER

2.1 Khái niệm

Trong machine learning, các phương pháp tối ưu (optimizer) được định nghĩa là các thuật toán điều chỉnh các thuộc tính của mô hình như trọng số (weight) và bias trong quá trình huấn luyện để tối thiểu hàm mất mát, giúp cải thiện độ chính xác cho các dự đoán của mô hình. Các phương pháp tối ưu khác nhau sẽ có các ưu nhược điểm khác nhau, việc chọn phương pháp tối ưu sẽ ảnh hưởng rất nhiều đễn hiệu suất mô hình.

2.2 Các phương pháp

2.2.1 Gradient Descent

Đối với bài toán tối ưu hàm mất mát (loss function), giá trị nhỏ nhất của một hàm số là điểm mà đạo hàm bằng 0. Tuy nhiên hầu hết các trường hợp thì việc tính phương trình đạo hàm bằng 0 rất phức tạp, thậm chí là bất khả thi. Thay vào đó người ta tìm điểm có giá trị gần với điểm cực tiểu nhất và xem nó như nghiệm của bài toán.

Gradient Descent là một phương pháp cơ được dùng để giải quyết vấn đề này. Gradient Descent xuất phát từ điểm gần với điểm cực tiểu sau đó dùng phép lặp để tiến dần đến điểm cần tìm. Công thức:

w_new = w_old – n * f’(x)

Trong đó w là tham số của mô hình, learning rate n là khoảng cách hay bước nhảy mà thuật toán sẽ điều chỉnh tham số sau mỗi vòng lặp. Thuật toán sẽ dừng lại khi w không còn cải thiện được nữa hoặc cải thiện rất ít.

Gradient descent sẽ phụ thuộc vào nhiều yếu tố như việc lựa chọn tham số bắt đầu của mô hình sẽ ảnh hưởng đến tốc độ hội tụ của bài toán. Việc lựa chọn learning rate cũng là một vấn đề khi lựa chọn learning rate quá nhỏ sẽ làm cho tốc độ hội tụ chậm, ảnh hưởng đến quá trình training và lựa chọn learning rate quá lớn sẽ làm cho thuật toán không hội tụ được vì cứ quanh quẩn xung quanh đích

Ưu điểm:

- Dễ hiểu, dễ triển khai

Nhược điểm

- Phụ thuộc vào việc lựa chọn tham số ban đầu w và learning rate
- Không hoạt động tốt với các hàm có nhiều cực tiểu
- Không hiệu quả với các bộ dữ liệu có kích thước lớn và phức tạp

2.2.2 Stochastic Gradient Descent (SGD)

Phương pháp này là một biến thể của Gradient Descent. Thay vì chọn cả tập dữ liệu để thực hiện tính toán cho mỗi vòng lặp thì thuật toán này sẽ ngẫu nhiên chọn một phần dữ liệu để tính toán. SGD Descent đầu tiên cần lựa chọn tham số ban đầu và learning rate, sau đó ngẫu nhiên chọn các tập con để thực hiện tính toán cho đến khi đến điểm cực tiểu của hàm số.

Do đó, SGD sử dụng nhiều vòng lặp hơn Gradient Descent truyền thống. Tuy nhiên, SGD hội tụ rất nhanh, chỉ cần vài vòng lặp là đã đến điểm cực tiểu. Bên cạnh đó, chi phí tính toán của SGD vẫn tối ưu hơn so với Gradient Descent truyền thống do không lặp qua cả dataset.

Do tính chất ngẫu nhiên lựa chọn tập dữ liệu để học thì hàm mất mát sẽ dao động nhiều hơn. Khiến cho SGD thiếu ổn định tuy nhiên có khả năng di chuyển đến các điểm cực tiểu khác nhiều tiềm năng. Khả năng hội tụ của SGD cũng tương tự với GD

Ưu điểm:

- Hoạt động tốt với tập dữ liệu lớn do không phải tính toán cả tập cho mỗi vòng lặp mà chỉ ngẫu nhiên chọn tập con
- Có thể hoạt động với các hàm nhiều cực tiểu

Nhược điểm

- Vẫn chưa khác phục được nhược điểm của Gradient Descent là lựa chọn tham số ban đầu và learning rate
- Do việc lựa chọn tập dữ liệu con là ngẫu nhiên nên thuật toán thiếu ổn định
- Vẫn có khả năng bị kẹt ở các cực tiểu chưa tối ưu trong các bài toán nhiều cực tiểu

2.2.3 (Stochastic) Gradient Descent với Momentum

Như ta đã biết thì Gradient Descent rất nhạy cảm với các hàm số có nhiều cực tiểu. Để khắc phục vấn đề này, người ta đề ra giải pháp là kết hợp Gradient Descent với Momentum. Tức là khi thuật toán di chuyển đến một điểm cực tiểu chưa tối ưu thì ta sẽ cấp cho nó một vận tốc ban đầu đủ lớn để cho nó vượt qua điểm đó để tiến tới điểm cực tiểu khác tối ưu hơn. Công thức:

w_new = w_old – (gama.v + n * f'(x))

Trong đó gama là một tham số thường là 0.9, v là thông tin của đà tức vận tốc tại thời điểm trước đó (ta xem vận tốc ban đầu v_0 = 0)

Ưu điểm:

- Giả quyết được vấn đề hàm có nhiều cực tiểu của gradient descent

Nhược điểm:

- Khi gần đến đích (cực tiểu tối ưu) thì sẽ mất khá nhiều thời gian dao động xung quanh điểm đó trước khi tới hẳn, lí do chính là vì có đà

2.2.4 Adagrad (Adaptive Gradient Algorithm)

Trong các optimizer trên thì learning rate luôn cố định trong suốt quá trình training và việc này có thể không phù hợp. Trong thực tế, dữ liệu sẽ có các thuộc tính có độ quan trọng khác nhau nên việc gán cùng một giá trị learning rate cho tất cả các thuộc tính là không thực tế.

Adagrad là một phương pháp tối ưu dùng learning rate khác nhau cho mỗi vòng lặp. Tốc độ biến thiên của learing rate sẽ phụ thuộc vào độ biến thiên của các tham số trong quá trình training, tham số biến thiên càng nhiều thì learning rate cũng sẽ biến thiên càng nhiều. Như vậy, với các đặc trưng có độ quan trọng thấp hay tốc độ biến thiên cao thì Adagrad sẽ cho learning rate cao để đưa tham số ứng với đặc trưng đó càng thấp. Ngược lại, với các đặc trưng có độ quan trọng cao hay tốc độ biến thiên chậm thì thuật toán sẽ giữ learning rate thấp. Công thức:

$$w_{t+1} = w_t - \frac{n}{\sqrt{G_t + \varepsilon}} \cdot g_t$$

Trong đó ε là một số dương khá nhỏ để tránh tính trạng chia cho 0, g_t là đạo hàm riêng của hàm mất mát theo w_t tại bước t, G_t là ma trận đường chéo trong đó các phần tử (i, i) là tổng bình phương của các đạo hàm tương ứng với w_t tính đến bước t.

Ưu điểm:

- Giải quyết được vấn đề điều chỉnh learning rate cho các đặc trưng có độ quan trọng khác nhau

Nhược điểm:

- Learning rate ban đầu vẫn phải chọn thủ công
- Learning rate giảm rất nhanh, cực kì nhỏ làm cho quá trình training bị đóng băng

2.2.5 RMSProp

Để khắc phục tình trạng suy giảm learning rate của Adagrad thì RMSProp sẽ chia learning rate cho trung bình của bình phương đạo hàm. Tốc độ học thích ứng sẽ được giữ nguyên bởi trung bình bình phương sẽ

Công thức:

$$E[g^2]_t = 0.9\, E[g^2]_{t-1} + 0.1 g_t^2$$

$$w_{t+1} = w_t - \frac{n}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g_t$$

RMSProp sẽ dùng dấu của đạo hàm để quyết định việc cập nhật learning rate cho các tham số. Hai đạo hàm sẽ được so sánh dấu với nhau, nếu dấu của đạo hàm làm như nhau thì RMSProp sẽ tăng learning rate, và ngược lại sẽ giảm learning rate và tiếp tục cập nhật trọng số.

Ưu điểm:

- Giải quyết được tốc độ học giảm nhanh của Adagrad

Nhược điểm:

- Chưa xử lý được hàm có nhiều cực tiểu

2.2.6 Adam

Adam là thuật toán kết hợp giữa RMSProp và momentum. Adam không chỉ lưu trữ trung bình bình phương các đạo hàm trước đó v_t mà còn lưu trữ trung bình momentum m_t. Công thức của v_t và m_t được tính như sau:

$$m_t = m_{t-1}\beta_1 + (1 - B_1)g_t$$

$$v_t = v_{t-1}\beta_2 + (1 - B_2)g_t$$

Trong đó m_t và v_t thường được khởi tạo là các vector 0 và β_1 và β_2 là các hệ số không âm thường được chọn lần lược là 0.9 và 0.999, do đó các giá trị này thường có khuynh hướng chứa nhiều giá trị 0 do β_1 và β_2 thường có xu hướng bằng 1. Do đó ta thêm các thành phần giúp bias-corrected

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

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

Từ đó:

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

Vì kết hợp giữa momentum để dễ dàng tiến đến cực tiểu tối ưu nhất và hệ số learning rate tự thích ứng với các tham số khác nhau của RMSProp khiến cho Adam trở thành một thuật toán tối ưu mạnh mẽ và thông dụng nhất hiện nay.

2.3 Tổng kết

Các phương pháp Optimizer đề có nhược điểm và ưu điểm riêng, tùy vào mục tiêu của bài toán sẽ có những lựa chọn phù hợp khác nhau. Tuy vậy, trong các thuật toán trên, có thể xem Adam là thuật toán tối ưu nhất khi hỗ trợ tích hợp momentum lẫn learning rate tự điều chỉnh. Cùng với đó là các thử nghiệm thực tế đã cho thấy Adam có hiệu quả rất tốt, cải thiện và xử lý hầu hết các vấn đề và giới hạn của cá phương pháp còn lại nêu trên.

CHƯƠNG 3. CONTINUAL LEARNING VÀ TEST TEST PRODUCTION

3.1 Continual Learning

3.1.1 Khái niệm

Continual Learning (học liên tục) là khái niệm mà mô hình học máy sẽ không quên những gì đã học khi được tiếp xúc với dữ liệu mới. Trong nhiều trường hợp, khi một mô hình học máy được đào tạo trên một tập dữ liệu và sau đó áp dụng vào một tập dữ liệu mới, nó có thể gặp phải vấn đề quên mất kiến thức cũ khi học thông tin mới. Điều này đặt ra thách thức lớn, đặc biệt là trong các ứng dụng y tế, tài chính, và nhiều lĩnh vực khác, nơi dữ liệu thay đổi liên tục theo thời gian.

3.1.2 Vai trò

Continual Learning giúp giải quyết các vấn đề sau:

- Vấn đề xảy ra những thay đổi bất ngờ và nhanh chóng
- Vấn đề không thể có được dữ liệu huấn luyện cho một sự kiện cụ thể như các sự kiện thường không xảy ra thường xuyên, việc đó yêu cầu model phải thích ứng trong suốt thời gian diễn ra sự kiệ đó
- Vấn đề về các sự kiện chưa từng xảy ra như đưa ra dự đoán cho một người dùng chưa từng có lịch sử nào ghi nhận do đó nếu model lỗi thời thì sẽ đưa ra các dự đoán không hợp lý

3.1.3 Phương pháp

3.1.3.1 Stateless retraining

Đào tạo lại model sau một khoản thời gian với trọng số được tạo ngẫu nhiên và dữ liệu mới hơn

Tuy nhiên có thể bị trùng lặp data với dữ liệu đã được dùng để train cho model

3.1.3.2 Stateful training

Khởi tạo mô hình với trọng số từ vòng đào tạo trước và tiếp tục việc training bằng cách sử dụng dữ liệu mới chưa được nhìn thấy.

Cho phép mô hình cập nhật với lượng dữ liệu đáng kể ít hơn. Cho phép mô hình hội tụ nhanh hơn và sử dụng ít tài nguyên hơn.

3.1.3.3 Các phương pháp khác

Regularization Techniques: Sử dụng các kỹ thuật như Elastic Weight Consolidation (EWC) để giữ cho trọng số của mô hình ổn định sau khi đã học một số dữ liệu.

Memory-Augmented Networks: Sử dụng bộ nhớ ngoại vi để giữ lại thông tin quan trọng từ các nhiệm vụ trước và sử dụng chúng khi học nhiệm vụ mới.

Dynamic Architectures: Cấu trúc mô hình có thể thay đổi độ phức tạp của nó để học thông tin mới mà không làm mất thông tin cũ

Đặc trưng được tính toán để thực hiện suy luận. Một số công ty lưu trữ các đặc trưng đã tính toán cho mỗi mẫu dữ liệu để có thể tái sử dụng chúng cho việc đào tạo liên tục và tiết kiệm một số tính toán. Điều này được gọi là "log and wait".

3.2 Test Production

3.2.1 Khái niệm

Test Production là khái niệm mà trước khi mô hình học máy được triển khai ra thực tế thì nó cần được kiểm thử để đảm bảo rằng mô hình là an toàn và hoạt động ổn định

3.2.2 Phương pháp

3.2.2.1 Shadow Deployment

Triển khai mô hình mới với mô hình hiện tại Chuyển mọi yêu cầu đến cả hai mô hình, nhưng chỉ phục vụ dự đoán của mô hình hiện tại. Ghi lại các dự đoán của cả hai mô hình để sau đó so sánh chúng

3.2.2.2 Canary Release

Triển khai mô hình mới và mô hình hiện tạicùng một lúc, nhưng bắt đầu với việc không chuyển giao bất kỳ lưu lượng nào cho mô hình mới. Dần dần chuyển lưu lượng từ mô hình hiện tại sang mô hình mới (còn được gọi là canary). Theo dõi các chỉ số hiệu suất của mô hình mới, nếu chúng trông tốt, tiếp tục chuyển lưu lượng cho đến khi tất cả lưu lượng đều được chuyển đến mô hình mới.

3.2.2.3 Badit

Thuật toán Bandit là một thuật toán giữ theo dõi hiệu suất hiện tại của từng biến thể mô hình và đưa ra quyết định động cho mỗi yêu cầu về việc sử dụng mô hình có hiệu suất tốt nhất cho đến nay (tức là khai thác kiến thức hiện tại) hoặc thử nghiệm bất kỳ mô hình nào khác để có thêm thông tin về chúng (tức là khám phá trong trường hợp một trong những mô hình khác có thể tốt hơn).

3.2.2.4 A/B Testing

Triển khai mô hình mới song song với mô hình hiện tại (mô hình A) và chuyển một phần trăm lưu lượng tới mô hình mới (mô hình B). Dự đoán từ mô hình mớiđược 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 mới có độ chệch thống kê so với mô hình hiện tại không.

TÀI LIỆU THAM KHẢO

Tiếng Anh

Ayush Gupta . Access 22/12/2023. A Comprehensive Guide on Optimizers in Deep Learning. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

serodriguez68. Continual learning and test in production. Access 22/12/2023. *https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md*