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

NGUYỄN QUANG HUY - 52000057

BÁO CÁO CUỐI KÌ
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

NGUYỄN QUANG HUY - 52000057

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

Để hoàn thành bài báo cáo này, tôi xin gửi lời cảm ơn đến TS Lê Anh Cường đã hướng dẫn cho tôi đi từ những cái cơ bản nhất của môn Nhập môn học máy để có thể có đủ kiến thức hoàn thành bài cáo cáo cuối kì này, xin chân thành cảm ơn thầy!

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

Tác giả

Nguyễn Quang Huy

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

Tôi xin cam đoan đây là công trình nghiên cứu của 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 Báo cáo 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 Báo cáo 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

Nguyễn Quang Huy

TÓM TẮT

Bài báo cáo này tập trung vào quá trình tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy, các vấn đề về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

ABSTRACT

This report focuses on the investigation and comparison of various optimizer methods in training machine learning models, issues related to Continual Learning, and Test Production when constructing a machine learning solution for addressing a specific problem.

MỤC LỤC

DANH MỤC BẢNG BIỂU

PHẦN 1. OPTIMIZER

1.1 Giới thiệu

Optimizer là một thành phần quan trọng trong quá trình đào tạo các mô hình học máy. Vai trò chính của nó là giảm thiểu giá trị của hàm chi phí hoặc tổn thất bằng cách điều chỉnh các tham số của mô hình. Quá trình tối ưu hóa nhằm mục đích tìm ra bộ tham số tối ưu mang lại hiệu suất tốt nhất cho mô hình trong nhiệm vụ nhất định.

Trong bối cảnh học máy, vấn đề tối ưu hóa thường được trình bày như sau: đưa ra một mô hình có các tham số có thể điều chỉnh và tập dữ liệu huấn luyện với các giá trị đích tương ứng, mục tiêu là tìm các giá trị tham số giúp giảm thiểu hàm chi phí hoặc tổn thất được xác định trước. Hàm chi phí này định lượng sự khác biệt giữa kết quả đầu ra dự đoán của mô hình và giá trị mục tiêu thực tế.

Trình tối ưu hóa đạt được mức giảm thiểu này bằng cách điều chỉnh lặp đi lặp lại các tham số mô hình theo hướng ngược lại với độ dốc của hàm chi phí đối với các tham số đó. Độ dốc cung cấp thông tin về mức tăng dốc nhất của hàm chi phí và việc di chuyển theo hướng ngược lại cho phép trình tối ưu hóa giảm xuống mức tối thiểu cục bộ hoặc toàn cục của hàm chi phí.

1.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một thuật toán tối ưu hóa mạnh mẽ thường được sử dụng trong học máy và học sâu để giảm thiểu chi phí hoặc hàm mất mát trong quá trình đào tạo mô hình. Nó là một biến thể của thuật toán tối ưu hóa giảm độ dốc đưa tính ngẫu nhiên vào quy trình, làm cho nó hiệu quả hơn và phù hợp hơn với các tập dữ liệu lớn.

Dưới đây là bảng phân tích các khái niệm chính liên quan đến Stochastic Gradient Descent:

1.2.1 Gradient Descent (GD)

Gradient Descent là một thuật toán tối ưu hóa lặp lại được sử dụng để giảm thiểu hàm bằng cách điều chỉnh các tham số của hàm đó một cách lặp đi lặp lại. Đó

là một thuật toán nền tảng trong học máy và tối ưu hóa, thường được sử dụng trong các mô hình đào tạo để giảm thiểu hàm chi phí hoặc tổn thất. Ý tưởng cơ bản là tiến tới mức tối thiểu của hàm bằng cách thực hiện các bước tỷ lệ với âm của gradient của hàm tại điểm hiện tại.

- Giảm độ dốc truyền thống liên quan đến việc cập nhật các tham số mô hình (trọng số và độ lệch) theo hướng ngược lại với độ dốc của hàm chi phí đối với các tham số đó.
- Các điểm gradient theo hướng tăng mạnh nhất của hàm chi phí và việc di chuyển ngược lại sẽ giúp đạt mức tối thiểu.

1.2.2 Batch Gradient Descent (BGD)

Batch gradient Descent là một biến thể của thuật toán tối ưu hóa gradient Descent trong đó toàn bộ tập dữ liệu huấn luyện được sử dụng để tính toán độ dốc của hàm chi phí hoặc tổn thất đối với các tham số mô hình trong mỗi lần lặp. Nó thường được gọi là giảm độ dốc "vanilla" hoặc "cổ điển". Mục tiêu của thuật toán là tìm mức tối thiểu toàn cục của hàm chi phí bằng cách điều chỉnh lặp lại các tham số theo hướng ngược lại với gradient.

- Trong phương pháp giảm độ dốc hàng loạt, toàn bộ tập dữ liệu được sử dụng để tính toán độ dốc của hàm chi phí trước khi cập nhật các tham số mô hình.
- Mặc dù cách tiếp cận này đảm bảo sự hội tụ đến mức tối thiểu toàn cầu nhưng nó có thể tốn kém về mặt tính toán, đặc biệt đối với các tập dữ liệu lớn

1.2.3 Stochastic Gradient Descent (SGD)

Stochastic gradient Descent (SGD) là một biến thể của thuật toán tối ưu hóa gradient Descent truyền thống được sử dụng để đào tạo các mô hình học máy. Không giống như Batch gradient Descent, tính toán độ dốc của hàm chi phí bằng cách sử dụng toàn bộ tập dữ liệu huấn luyện trong mỗi lần lặp, SGD cập nhật các tham số mô hình chỉ bằng cách sử dụng một điểm dữ liệu được chọn ngẫu nhiên duy nhất (hoặc một tập hợp con nhỏ gọi là lô nhỏ) tại một thời điểm.

Dưới đây là các đặc điểm và bước chính của Stochastic Gradient Descent:

- Khởi tạo:
    - Bắt đầu với dự đoán ban đầu cho các tham số của mô hình (θ)
- Quá trình lặp lại:
    - Trong mỗi lần lặp, xáo trộn ngẫu nhiên tập dữ liệu huấn luyện.
    - Đối với mỗi điểm dữ liệu i (hoặc lô điểm dữ liệu nhỏ):
        - Tính toán độ dốc (đạo hàm riêng) của hàm chi phí hoặc hàm tổn thất đối với từng tham số chỉ sử dụng điểm dữ liệu hiện tại (hoặc lô nhỏ).

          $$\nabla J(\theta; x^{(i)}, y^{(i)})$$

        - Cập nhật các tham số bằng cách sử dụng gradient được tính toán.

          $$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \nabla J(\theta_{\text{old}}; x^{(i)}, y^{(i)})$$

          Trong đó:

          θ_{new} là vectơ tham số được cập nhật.

          θ_{old} là vectơ tham số hiện tại.

          α là tốc độ học tập, một siêu tham số xác định kích thước của các bước.
- Lặp lại:

  Lặp lại quá trình lặp cho đến khi hội tụ hoặc đạt tiêu chuẩn dừng. Sự hội tụ xảy ra khi sự thay đổi trong hàm chi phí hoặc các tham số trở nên đủ nhỏ.
- Tỷ lệ học tập (α):

  Tốc độ học tập là một siêu tham số quan trọng trong SGD, tương tự như Batch gradient Descent. Nó ảnh hưởng đến kích thước của các bước thực hiện theo hướng gradient. Tốc độ học tập cần phải được lựa chọn cẩn thận; quá nhỏ có thể dẫn đến hội tụ chậm và quá lớn có thể dẫn đến dao động hoặc phân kỳ.

SGD đặc biệt hữu ích khi xử lý các tập dữ liệu lớn, vì nó chỉ xử lý một điểm dữ liệu (hoặc một lô nhỏ) tại một thời điểm, giúp tính toán hiệu quả hơn so với Batch gradient Descent. Tuy nhiên, bản chất ngẫu nhiên của các bản cập nhật có thể gây ra nhiễu, dẫn đến sự hội tụ thất thường hơn so với sự hội tụ mượt mà hơn của Batch

gradient Descent. Giảm dần độ dốc hàng loạt nhỏ là một sự thỏa hiệp kết hợp một số ưu điểm của cả Giảm dần độ dốc hàng loạt và ngẫu nhiên.

1.3 Adagrad (Adaptive Gradient Algorithm)

Adagrad là một thuật toán tối ưu hóa được thiết kế để đào tạo các mô hình machine learning, đặc biệt là trong bối cảnh deep learning. Nó được đề xuất bởi Duchi, Hazan và Singer vào năm 2011. Adagrad được biết đến với phương pháp tiếp cận tốc độ học thích ứng, trong đó nó điều chỉnh tốc độ học cho từng tham số riêng lẻ dựa trên thông tin độ dốc lịch sử.

Dưới đây là các tính năng và bước chính của thuật toán Adagrad:

- Khởi tạo:
  - Bắt đầu với dự đoán ban đầu cho các tham số của mô hình (θ)
- Khởi tạo gradient bình phương tích lũy:

  Khởi tạo một vectơ r để lưu trữ gradient bình phương tích lũy. Mỗi phần tử ri tương ứng với tổng bình phương của gradient đối với tham số θi cho đến lần lặp hiện tại.

$$r_i = \sum_{t=1}^{T} (\nabla J(\theta_t)_i)^2$$

- Quá trình lặp lại:
  - Đối với mỗi lần lặp t:
  - Tính gradient của hàm chi phí hoặc tổn thất theo các tham số:

$$\nabla J(\theta_t)$$

  - Cập nhật gradient bình phương tích lũy:

$$r_{t+1} = r_t + (\nabla J(\theta_t))^2$$

Trong đó:

α là tốc độ học, có thể là hằng số cố định hoặc được chọn thích ứng.

ϵ là một hằng số nhỏ (thường được thêm vào để ổn định số học nhằm tránh chia cho 0).

- Lặp lại:
  - Lặp lại quá trình lặp cho đến khi hội tụ hoặc đạt tiêu chuẩn dừng.
- Tỷ lệ học tập thích ứng:
  - Đặc điểm chính của Adagrad là tốc độ học tập thích ứng. Các tham số đã tích lũy gradient lớn hơn trong quá khứ sẽ có tốc độ học hiệu quả nhỏ hơn và các tham số có gradient nhỏ hơn sẽ có tốc độ học hiệu quả lớn hơn. Điều này giúp tự động thích ứng với tầm quan trọng khác nhau của các thông số khác nhau.
  - Sự thích ứng này đạt được bằng cách duy trì tốc độ học riêng biệt cho từng tham số, cập nhật nó dựa trên tổng bình phương của độ dốc lịch sử.
- Xử lý dữ liệu thưa thớt:

  Adagrad hoạt động tốt với dữ liệu thưa thớt vì nó phân bổ các bản cập nhật lớn hơn một cách tự nhiên cho các tham số liên quan đến các tính năng không thường xuyên.

Mặc dù có những ưu điểm nhưng Adagrad vẫn có một số hạn chế, chẳng hạn như việc tích lũy gradient bình phương dẫn đến tốc độ học giảm dần theo thời gian, điều này có thể khiến quá trình học trở nên quá chậm trong các giai đoạn đào tạo sau này. Để giải quyết vấn đề này, các biến thể như RMSprop và Adam đã được giới thiệu để cung cấp phương pháp tiếp cận tốc độ học tập thích ứng cân bằng hơn.

1.4 RMSprop (Root Mean Square Propagation)

RMSprop là một thuật toán tối ưu hóa được thiết kế để giải quyết một số hạn chế của Adagrad, đặc biệt là vấn đề tốc độ học giảm nhanh khi xảy ra thường xuyên và có độ dốc lớn. Được đề xuất bởi Geoffrey Hinton trong một bài giảng năm 2012, RMSprop điều chỉnh tốc độ học riêng cho từng tham số bằng cách sử dụng đường trung bình động của các gradient bình phương.

Dưới đây là các tính năng và bước chính của thuật toán RMSprop:

- Khởi tạo:
  - Khởi tạo các tham số của mô hình (θ).
- Khởi tạo gradient bình phương tích lũy:
  - Khởi tạo giá trị trung bình của gradient bình phương, ký hiệu là $E[g^2]$, trong đó g là độ dốc.

$$E[g^2]t = E[g^2]_{t-1} + (1 - \beta)(\nabla J(\theta_t))^2$$

  - Ở đây, β là hệ số phân rã (thường gần bằng 1) và $\nabla J(\theta_t)$ là độ dốc của hàm chi phí hoặc hàm tổn thất đối với các tham số khi lặp t.
- Cập nhật thông số:
  - Cập nhật các tham số bằng cách sử dụng giá trị trung bình chạy được tính toán của gradient bình phương:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot \nabla J(\theta_t)$$

Trong đó:

α là tốc độ học, một siêu tham số xác định kích thước bước.

ϵ là một hằng số nhỏ (thường được thêm vào để ổn định số).

- Lặp lại:

  Lặp lại quá trình lặp cho đến khi hội tụ hoặc đạt tiêu chuẩn dừng.
- Tỷ lệ học tập thích ứng:

  Tương tự như Adagrad, RMSprop điều chỉnh tốc độ học cho từng tham số dựa trên gradient bình phương tích lũy. Tuy nhiên, RMSprop sử dụng mức trung bình giảm dần, giúp giảm thiểu vấn đề giảm tốc độ học tập quá mức.

RMSprop giúp giải quyết vấn đề giảm tỷ lệ học tập ở Adagrad bằng cách đưa ra hệ số suy giảm (β) để kiểm soát sự đóng góp của các gradient bình phương trong quá khứ. Hành vi tốc độ học thích ứng này đặc biệt có lợi trong các tình huống trong đó bối cảnh của vấn đề tối ưu hóa có quy mô khác nhau dọc theo các chiều khác nhau.

Điều đáng chú ý là RMSprop là một trong những thuật toán nền tảng đặt nền móng cho các thuật toán tối ưu hóa nâng cao hơn, bao gồm cả Adam. Mặc dù RMSprop có hiệu quả nhưng có thể có trường hợp các thuật toán khác, chẳng hạn như Adam, cung cấp những cải tiến bổ sung về hiệu suất và độ bền.

1.5 Adam (Adaptive Moment Estimation)

Adam, viết tắt của Adaptive Moment Estimation, là một thuật toán tối ưu hóa kết hợp các ý tưởng từ RMSprop và Momentum. Nó được giới thiệu bởi Diederik P. Kingma và Jimmy Ba trong bài báo năm 2014 của họ có tựa đề "Adam: Phương pháp tối ưu hóa ngẫu nhiên". Adam được sử dụng rộng rãi trong việc đào tạo các mô hình machine learning, đặc biệt là trong bối cảnh deep learning.

Dưới đây là các tính năng và bước chính của thuật toán RMSprop:

- Khởi tạo:
  - Khởi tạo các tham số của mô hình (θ).
- Khởi tạo khoảnh khắc:
  - Khởi tạo hai vectơ trung bình động, m và v, để lưu trữ khoảnh khắc đầu tiên (trung bình) và khoảnh khắc thứ hai (phương sai không tập trung) của độ dốc tương ứng.

$$m_0 = 0; v_0 = 0$$

- Quá trình lặp lại:
  - Đối với mỗi lần lặp t:
    - Tính toán độ dốc của hàm chi phí hoặc tổn thất đối với các tham số: $\nabla J(\theta_t)$
    - Cập nhật ước tính thời điểm đầu tiên (m_t) và ước tính thời điểm thứ hai (v_t).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla J(\theta_t)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla J(\theta_t))^2$$

Trong đó:

β_1 và β_2 là tốc độ phân rã theo cấp số nhân (thường gần bằng 1).

- Các phiên bản đã hiệu chỉnh sai lệch của m và v được tính toán để giải quyết sai lệch khởi tạo:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} ; \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- Cập nhật các tham số bằng cách sử dụng tốc độ học thích ứng:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t$$

Trong đó:

α là tốc độ học, một siêu tham số xác định kích thước bước.

ϵ là một hằng số nhỏ (thường được thêm vào để ổn định số).

- Lặp lại:

  Lặp lại quá trình lặp cho đến khi hội tụ hoặc đạt tiêu chuẩn dừng.

- Tỷ lệ và động lực học tập thích ứng:

  Adam điều chỉnh tốc độ học cho từng tham số dựa trên cả khoảnh khắc đầu tiên (thuật ngữ động lượng) và khoảnh khắc thứ hai (phương sai không tâm) của độ dốc. Sự kết hợp này cho phép Adam thể hiện hiệu suất mạnh mẽ trong nhiều vấn đề tối ưu hóa.

Adam đã trở thành một lựa chọn phổ biến để đào tạo mạng lưới thần kinh sâu nhờ tốc độ học thích ứng, hành vi giống động lượng và tính hiệu quả trong việc xử lý độ dốc nhiễu hoặc thưa thớt. Tuy nhiên, điều quan trọng cần lưu ý là mặc dù Adam hoạt động tốt trong nhiều trường hợp nhưng không có thuật toán tối ưu hóa chung

nào phù hợp cho tất cả và hiệu suất có thể phụ thuộc vào đặc điểm cụ thể của vấn đề hiện tại.

Bảng 1-1 Bảng so sánh các phương pháp Optimizer

	SGD	Adagrad	RMSprop	Adam
Khả năng thích ứng của tỷ lệ học tập	Sử dụng tỷ lệ học tập cố định.	Điều chỉnh tốc độ học tập riêng cho từng tham số bằng cách tích lũy gradient bình phương	Điều chỉnh tốc độ học tập riêng lẻ với mức trung bình bình phương giảm dần theo cấp số nhân	Điều chỉnh tốc độ học tập riêng lẻ bằng cách sử dụng kết hợp các khoảnh khắc độ dốc và độ dốc bình phương
Đường trung bình động để điều chỉnh tỷ lệ học tập	Không có	Không áp dụng	Sử dụng đường trung bình động hàm mũ cho gradient bình phương	Sử dụng đường trung bình động cho cả khoảnh khắc độ dốc và độ dốc bình phương
Hiệu chỉnh sai lệch cho đường trung bình động	Không có	Không áp dụng	Áp dụng hiệu chỉnh độ lệch cho các đường trung bình động	Áp dụng hiệu chỉnh độ lệch cho cả khoảnh khắc độ dốc và độ dốc bình phương
Tích lũy độ dốc	Sử dụng một ví dụ huấn luyện tại một thời điểm (ngẫu nhiên)	Tích lũy gradient bình phương cho từng tham số	Tích lũy gradient bình phương với đường trung bình động hàm mũ	Tích lũy các khoảnh khắc độ dốc và độ dốc bình phương

				bằng các đường trung bình động
Thành phần động lượng	Có thể sử dụng một thuật ngữ xung lượng riêng	Không áp dụng	Không có	Tích hợp các khoảnh khắc độ dốc vào các cập nhật tham số
Độ nhạy cảm với siêu tham số	Nhạy cảm với việc lựa chọn tốc độ học tập	Nhạy cảm với việc lựa chọn tốc độ học tập	Ít nhạy cảm hơn với việc lựa chọn tốc độ học tập so với Adagrad	Nói chung ít nhạy cảm hơn với siêu tham số so với SGD, Adagrad và RMSprop
Mức độ phổ biến và sử dụng	Cổ điển nhưng có thể chậm trên các tập dữ liệu lớn	Ngày nay ít phổ biến hơn do vấn đề tỷ lệ học tập trở nên quá nhỏ	Phổ biến cho nhiều tác vụ khác nhau nhưng có thể yêu cầu điều chỉnh siêu tham số cẩn thận	Rất phổ biến do hiệu suất mạnh mẽ và độ nhạy thấp với siêu tham số

Tóm lại, mỗi thuật toán đều có điểm mạnh và điểm yếu và việc lựa chọn giữa chúng thường phụ thuộc vào tính chất cụ thể của nhiệm vụ, kích thước tập dữ liệu và kinh nghiệm thực nghiệm với siêu tham số. Adam thường được khuyên dùng vì hiệu suất tổng thể mạnh mẽ trong nhiều tình huống khác nhau. Tuy nhiên, bạn nên thử nghiệm nhiều trình tối ưu hóa và cấu hình để tìm ra cái nào phù hợp nhất cho một tác vụ cụ thể.

PHẦN 2. CONTINUAL LEARNING & TEST PRODUCTION

2.1 Continual learning

Continual learning, còn được gọi là lifelong learning, một phương pháp học máy cho phép các mô hình học hỏi và thích ứng liên tục từ dữ liệu mới mà chúng gặp phải theo thời gian, thay vì được đào tạo tĩnh trên một tập dữ liệu cố định. Không giống như học máy truyền thống thường giả định tất cả dữ liệu đều có sẵn cùng một lúc, các mô hình học liên tục phải thích ứng với thông tin mới trong khi vẫn giữ được những gì chúng đã học.

Điều này đặc biệt quan trọng đối với các ứng dụng trong thế giới thực, nơi dữ liệu thường xuyên thay đổi và phát triển. Ví dụ: robot cần học cách điều hướng trong môi trường mới sẽ có thể xây dựng kiến thức hiện có về tránh chướng ngại vật đồng thời kết hợp thông tin mới về cách bố trí cụ thể của môi trường mới.

Những thách thức của việc học tập liên tục:

Một trong những thách thức chính của việc học liên tục là sự quên lãng nghiêm trọng, trong đó hiệu suất của mô hình đối với các nhiệm vụ đã học trước đó giảm đáng kể khi nó được huấn luyện về các nhiệm vụ mới. Điều này xảy ra vì các tham số bên trong của mô hình được cập nhật để phù hợp với dữ liệu mới, thường làm mất đi dữ liệu cũ.

Một số phương pháp đã được phát triển để giải quyết tình trạng quên lãng nghiêm trọng, bao gồm:

- Kỹ thuật chính quy hóa: Những kỹ thuật này phạt mô hình vì quên kiến thức cũ trong quá trình đào tạo các nhiệm vụ mới.
- Phát lại dữ liệu cũ: Điều này liên quan đến việc đào tạo lại mô hình theo định kỳ trên dữ liệu cũ để giúp nó giữ lại kiến thức trước đó.
- Chắt lọc: Điều này liên quan đến việc chuyển kiến thức từ một mô hình lớn hơn, phức tạp hơn sang một mô hình nhỏ hơn, nhanh hơn, sau đó có thể được sử dụng để học tập liên tục.

Học tập liên tục có nhiều ứng dụng tiềm năng, bao gồm:

- Robotics: Robot có thể học các kỹ năng mới và thích nghi với môi trường mới.
- Xử lý ngôn ngữ tự nhiên: Các mô hình ngôn ngữ có thể học ngôn ngữ mới và cải thiện khả năng hiểu văn bản theo thời gian.
- Thị giác máy tính: Hệ thống nhận dạng hình ảnh có thể học cách xác định các vật thể và cảnh mới.
- Chăm sóc sức khỏe: Hệ thống chẩn đoán y tế có thể học hỏi từ dữ liệu bệnh nhân mới và cải thiện độ chính xác của chúng theo thời gian.

Học tập liên tục là một lĩnh vực nghiên cứu đầy thách thức nhưng đầy hứa hẹn với tiềm năng cách mạng hóa cách chúng ta tương tác với máy móc. Khi nghiên cứu trong lĩnh vực này tiến triển, chúng ta có thể mong đợi được thấy những ứng dụng ấn tượng hơn nữa của việc học tập liên tục trong tương lai.

Khi xây dựng một giải pháp machine learning, việc học liên tục có thể mang lại những lợi thế khác biệt, đặc biệt khi xử lý các tình huống dữ liệu luôn biến động và đang phát triển. Đây là cách nó có thể có liên quan:

Sự phù hợp của vấn đề:

- Dữ liệu động: Nếu vấn đề của bạn liên quan đến dữ liệu liên tục thay đổi hoặc phát triển, như dữ liệu cảm biến từ máy móc hoặc dữ liệu thị trường tài chính, việc học hỏi liên tục có thể giúp mô hình của bạn thích ứng và duy trì độ chính xác theo thời gian.
- Dữ liệu hạn chế: Nếu bạn có dữ liệu ban đầu hạn chế nhưng mong muốn có nhiều dữ liệu hơn sau này, việc học hỏi liên tục cho phép bạn cải thiện dần mô hình của mình với mỗi lô dữ liệu mới, ngay cả khi nó thay đổi một chút.
- Nhiệm vụ mở: Đối với các nhiệm vụ mà tình huống hoặc kịch bản mới có thể phát sinh theo thời gian, việc học tập liên tục cho phép mô hình của bạn điều chỉnh và mở rộng nền tảng kiến thức mà không cần đào tạo lại hoàn toàn.

Những lợi ích:

- Độ chính xác được cải thiện: Các mô hình học tập liên tục có thể thích ứng với việc thay đổi mẫu dữ liệu và có khả năng đạt được độ chính xác cao hơn so với các mô hình tĩnh theo thời gian.
- Giảm nhu cầu đào tạo lại: Bạn không cần phải đào tạo lại hoàn toàn mô hình của mình mỗi khi có dữ liệu mới, tiết kiệm thời gian và tài nguyên.
- Tính linh hoạt và khả năng thích ứng: Mô hình của bạn có thể xử lý các tình huống và nhiệm vụ mới ngay cả khi chúng không được xem xét ban đầu, khiến mô hình trở nên linh hoạt và mạnh mẽ hơn.

Những thách thức và cân nhắc:

- Sự quên lãng nghiêm trọng: Như đã đề cập trước đó, mô hình có thể quên kiến thức đã học trước đó trong khi tiếp thu thông tin mới. Các kỹ thuật như chính quy hóa, phát lại dữ liệu cũ hoặc chắt lọc có thể giảm thiểu điều này.
- Chi phí tính toán: Các thuật toán học liên tục có thể phức tạp hơn và tốn kém hơn về mặt tính toán so với các phương pháp truyền thống.
- Độ phức tạp của việc đánh giá: Việc liên tục đánh giá và theo dõi hiệu suất của mô hình trên các luồng dữ liệu và nhiệm vụ khác nhau có thể là một thách thức.

Các phương pháp thực hiện:

- Học tăng dần theo nhiệm vụ: Huấn luyện mô hình của bạn theo từng nhiệm vụ một, dần dần thêm các nhiệm vụ mới trong khi vẫn duy trì kiến thức trước đó.
- Học tăng dần dữ liệu: Huấn luyện mô hình của bạn theo các lô dữ liệu mới một cách tuần tự, điều chỉnh các tham số của nó để kết hợp thông tin mới.
- Học tập suốt đời: Coi quá trình học tập là liên tục, liên tục cập nhật mô hình với dữ liệu và nhiệm vụ mới khi chúng có sẵn.

Việc chọn phương pháp phù hợp phụ thuộc vào vấn đề cụ thể, đặc điểm dữ liệu và tài nguyên của bạn.

Cuối cùng, việc kết hợp học tập liên tục vào giải pháp học máy của bạn có thể là một cách tiếp cận mạnh mẽ để xử lý dữ liệu động và các vấn đề đang phát triển. Tuy nhiên, hãy cân nhắc kỹ những thách thức và lựa chọn cách thực hiện phù hợp để đạt được kết quả tối ưu.

2.2 Test Produciton

Test Production là quá trình đánh giá hiệu suất và hành vi của mô hình học máy trong môi trường thực tế với dữ liệu và lưu lượng truy cập của người dùng thực tế.

Test Production đề cập đến quá trình thử nghiệm và đánh giá các mô hình học máy trong môi trường sản xuất thực tế. Giai đoạn này rất quan trọng để đảm bảo mô hình hoạt động tốt và nhất quán trong điều kiện sử dụng thực tế. Dưới đây là các khía cạnh chính của Sản xuất thử nghiệm trong học máy:

Mục tiêu: Đánh giá hiệu suất và hành vi của mô hình học máy của bạn trong thế giới thực với dữ liệu và lưu lượng truy cập thực tế của người dùng.

Những lợi ích:

- Xác định các vấn đề không thể nhìn thấy trong môi trường thử nghiệm được kiểm soát.
- Xác thực tính hiệu quả của mô hình dưới sự tương tác thực tế của người dùng.
- Theo dõi sự trôi dạt của mô hình và kích hoạt đào tạo lại khi cần thiết.
- Cung cấp những hiểu biết có giá trị để cải thiện mô hình hơn nữa.

Những thách thức:

- Yêu cầu cân nhắc cẩn thận về quyền riêng tư và bảo mật dữ liệu.
- Sự mạnh mẽ để xử lý những đầu vào bất ngờ và những thất bại tiềm ẩn là rất quan trọng.
- Việc giám sát và ghi nhật ký liên tục các kết quả đầu ra và số liệu của mô hình là rất cần thiết.

Kỹ thuật:

- Triển khai Canary: Triển khai dần dần cho một nhóm người dùng nhỏ để thử nghiệm ban đầu.
- Cờ tính năng: Kiểm soát và thử nghiệm các tính năng của mô hình mới trong sản xuất.
- Kiểm tra bóng: Chạy mô hình mới cùng với mô hình hiện có để so sánh.
- Thử nghiệm A/B: So sánh các phiên bản hoặc cấu hình mô hình khác nhau với lưu lượng truy cập trực tiếp.

Sự kết hợp giữa Continual Learning và Test Production có thể mang lại nhiều lợi ích quan trọng cho các giải pháp học máy, bao gồm:

- Tăng độ chính xác và tính phù hợp:

  Mô hình có khả năng tiếp tục học và thích ứng với các mô hình và xu hướng dữ liệu mới, duy trì độ chính xác ngay cả khi dữ liệu đầu vào thay đổi.
- Giảm nhu cầu đào tạo lại:

  Không cần đào tạo lại mô hình trên các lô dữ liệu mới liên tục.
- Tăng tính linh hoạt và khả năng thích ứng:

  Mô hình có khả năng xử lý các tình huống và nhiệm vụ mới mà không cần được xem xét ban đầu.

Để triển khai hiệu quả sự kết hợp này, cần xem xét những điều sau:

- Chọn phương pháp Continual Learning phù hợp:

  Nhiều phương pháp Continual Learning khác nhau, phụ thuộc vào vấn đề và đặc điểm dữ liệu.
- Thiết kế chiến lược Test Production vững chắc:

  Đảm bảo bảo mật dữ liệu, giám sát hành vi mô hình, và kích hoạt cảnh báo cho các vấn đề tiềm ẩn.

- Xây dựng các vòng phản hồi rõ ràng:

  Test Production cung cấp phản hồi về hiệu suất mô hình, điều này có thể được sử dụng để điều chỉnh và cải thiện quy trình Continual Learning.

Ví dụ cụ thể về cách áp dụng Continual Learning và Test Production trong giải pháp học máy:

- Trong hệ thống phân loại email rác, Continual Learning có thể cập nhật mô hình khi xuất hiện thuật ngữ và mẫu mới trong thư rác. Test Production đảm bảo rằng mô hình mới vẫn chính xác và không nhầm lẫn các email hợp pháp là rác.
- Trong hệ thống phát hiện gian lận thẻ tín dụng, Continual Learning cập nhật mô hình với thủ thuật mới. Test Production xác định xem mô hình mới có phát hiện gian lận hiệu quả hơn mô hình cũ hay không.
- Trong hệ thống đề xuất sản phẩm, Continual Learning cập nhật mô hình khi sở thích của người dùng thay đổi. Test Production đánh giá liệu các đề xuất mới có được người dùng ưa thích hơn các đề xuất cũ hay không.

Tổng cộng, Continual Learning và Test Production là những công cụ mạnh mẽ giúp xây dựng các giải pháp học máy đáng tin cậy và hiệu quả hơn.

Continual learning cung cấp một khuôn khổ cho các mô hình để thích ứng với những thay đổi được quan sát thấy trong Test Production. Dữ liệu mới và thông tin chi tiết từ Test Production có thể được sử dụng để tinh chỉnh các mô hình và cải thiện hiệu suất của chúng theo thời gian.

Test Production cung cấp nơi thử nghiệm trong thế giới thực cho các mô hình học tập liên tục. Nó giúp xác nhận tính hiệu quả của họ, xác định các vấn đề tiềm ẩn và kích hoạt đào tạo lại khi cần thiết.

Kết hợp lại, chúng tạo ra một vòng phản hồi để cải tiến lặp đi lặp lại. Test Production cung cấp phản hồi về hiệu suất của mô hình, phản hồi này được sử dụng để điều chỉnh và tinh chỉnh quá trình học hỏi liên tục, từ đó tạo ra các mô hình tổng thể tốt hơn.

Việc thực hiện sức mạnh tổng hợp này đòi hỏi phải lập kế hoạch và cân nhắc cẩn thận:

- Chọn phương pháp học tập liên tục phù hợp dựa trên đặc điểm vấn đề và dữ liệu của bạn.
- Thiết kế các chiến lược Test Production mạnh mẽ để đảm bảo quyền riêng tư của dữ liệu, giám sát hành vi của mô hình và kích hoạt cảnh báo về các vấn đề tiềm ẩn.
- Thiết lập các vòng phản hồi và quy trình rõ ràng để kết hợp những hiểu biết sâu sắc về Test Production vào quy trình học tập liên tục.

Bằng cách tích hợp hai khái niệm mạnh mẽ này, bạn có thể xây dựng các giải pháp máy học không chỉ chính xác và hiệu quả mà còn không ngừng học hỏi và thích ứng với những thay đổi trong thế giới thực, cuối cùng dẫn đến một giải pháp năng động và mạnh mẽ hơn.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Q. Nguyen, “4. Gradient Descent,” 3 April 2021. [Trực tuyến]. Available: https://ndquy.github.io/posts/gradient-descent-2/. [Đã truy cập 15 December 2023].

[2] "Bài 8: Gradient Descent (phần 2/2)," Buy me a coffe, 16 January 2017. [Online]. Available: https://machinelearningcoban.com/2017/01/16/gradientdescent2/. [Accessed 15 December 2023].

[3] T. T. Trực, “Optimizer- Hiểu sâu về các thuật toán tối ưu (GD,SGD,Adam,..),” 17 October 2020. [Trực tuyến]. Available: https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8. [Đã truy cập 17 December 2020].

[4] P. Ngoc, “Thuật toán tối ưu adam,” 27 May 2018. [Trực tuyến]. Available: https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m. [Đã truy cập 18 December 2023].

Tiếng Anh

[5] D. Villarraga, "AdaGrad," 14 December 2021. [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=AdaGrad. [Accessed 15 December 2023].

[6] J. Huang, "RMSProp," 21 December 2020. [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=RMSProp. [Accessed 19 December 2023].