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

LỤC TẤN VƯƠNG - 520H0198

BÀI TẬP 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

LỤC TẤN VƯƠNG - 520H0198

BÀI TẬP CUỐI KỲ
NHẬP MÔN HỌC MÁY

Người hướng dẫn
TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Thầy Lê Anh Cường, người đã hướng dẫn và hỗ trợ tận tình cho nhóm chúng em trong quá trình thực hiện báo cáo này. Sự kiên nhẫn, kiến thức chuyên môn, và kinh nghiệm quý báu của Thầy đã giúp nhóm chúng em nắm bắt được những kiến thức cần thiết và áp dụng chúng một cách hiệu quả để hoàn thiện báo cáo. Sự giảng dạy và sự hỗ trợ của Thầy không chỉ giúp chúng em vượt qua những khó khăn trong quá trình làm báo cáo, mà còn là nguồn cảm hứng để chúng em phấn đấu và học hỏi nhiều hơn nữa. Chúng em xin chân thành cảm ơn và mong rằng sẽ tiếp tục nhận được sự hỗ trợ và hướng dẫn của Thầy trong tương lai.

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

Tác giả

(Ký tên và ghi rõ họ tên

CHƯƠNG 1. CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH

1.1 Định nghĩa Optimizer

Optimizer trong học máy, đôi khi được gọi là thuật toán tối ưu hóa, là một phần quan trọng trong quá trình huấn luyện mô hình học máy và học sâu. Mục đích cơ bản của một optimizer là điều chỉnh các tham số của mô hình (thường là trọng số) để giảm thiểu hàm mất mát. Hàm mất mát này là một hàm số đo lường sự chênh lệch giữa kết quả dự đoán của mô hình và giá trị thực tế của dữ liệu.

1.1.1 Tầm Quan Trọng

- Học từ Dữ liệu: Trong học máy, mô hình cần 'học' từ dữ liệu đầu vào. Optimizer giúp mô hình điều chỉnh các trọng số của mình dựa trên lỗi mà nó tạo ra, giúp mô hình học hỏi và cải thiện qua mỗi lần lặp.
- Cải thiện Hiệu suất: Việc lựa chọn và cấu hình đúng đắn optimizer có thể dẫn đến việc huấn luyện mô hình nhanh chóng và hiệu quả hơn, đồng thời cũng giúp tăng độ chính xác của mô hình.
- Đối phó với Vấn đề Quá Khớp (Overfitting): Một số optimizer có các cơ chế như điều chỉnh tỷ lệ học (learning rate) để giúp mô hình không bị quá khớp với dữ liệu huấn luyện.
- Khả năng Thích ứng: Các optimizer hiện đại như Adam hay RMSprop có khả năng thích ứng tỷ lệ học cho từng tham số, giúp mô hình học tốt hơn trong những điều kiện dữ liệu phức tạp.

1.2 Các loại optimizer phổ biến

- SGD (Stochastic Gradient Descent)
- Momentum
- Adagrad
- RMSprop
- Adam

1.2.1 Phương pháp SGD (Stochastic Gradient Descent)

1.2.1.1 Cơ Bản về Gradient Descent

- Gradient Descent (GD): Là một phương pháp tối ưu hóa tìm điểm cực tiểu của hàm mất mát bằng cách di chuyển ngược hướng của gradient (đạo hàm) của hàm mất mát.
- Cập Nhật Tham Số: Trong GD, tham số θ của mô hình được cập nhật theo công thức: $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$ Trong đó, η là tỷ lệ học (learning rate) và là gradient của hàm mất mát J theo tham số θ.

1.2.1.2 Stochastic Gradient Descent

- SGD làm thay đổi cách tính toán gradient:
- Chọn Mẫu Ngẫu Nhiên: Thay vì tính toán gradient dựa trên toàn bộ tập dữ liệu, SGD chọn một mẫu dữ liệu ngẫu nhiên hoặc một mini-batch (nhóm nhỏ các mẫu) để tính toán gradient.
- Cập Nhật Thường Xuyên: Mỗi khi xử lý một mẫu (hoặc một mini-batch), SGD cập nhật ngay tham số θ của mô hình.
- Công Thức Cập Nhật: Với mỗi mẫu dữ liệu x_i và nhãn y_i, gradient được tính và tham số được cập nhật theo công thức: $\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x_i, y_i)$

1.2.1.3 Quá Trình Học

- Vòng Lặp Học: SGD lặp qua tập dữ liệu, chọn mẫu ngẫu nhiên và cập nhật tham số sau mỗi lần lặp.
- Khám Phá và Tinh Chỉnh: Do tính ngẫu nhiên, SGD khám phá không gian tham số một cách rộng rãi, giúp tránh bị mắc kẹt tại các cực tiểu địa phương không tối ưu.

1.2.1.4 Tối Ưu Hóa và Hội Tụ

- Tốc Độ Học và Hội Tụ: Tỷ lệ học (learning rate) η quyết định tốc độ cập nhật và ảnh hưởng đến quá trình hội tụ. Một tỷ lệ học quá lớn có thể dẫn đến bỏ qua điểm cực tiểu, trong khi tỷ lệ học quá nhỏ có thể làm chậm quá trình học.
- Tinh Chỉnh: Trong thực tế, thường cần nhiều vòng lặp và tinh chỉnh tham số để đạt được sự hội tụ tối ưu.

1.2.2 Phương pháp Momentum

1.2.2.1 Tổng Quan về Gradient Descent

Momentum được xây dựng dựa trên cơ sở của Gradient Descent:

- Gradient Descent: Phương pháp này cập nhật tham số mô hình bằng cách di chuyển ngược hướng của gradient của hàm mất mát để giảm thiểu giá trị của hàm mất mát.

1.2.2.2 Cơ Chế Hoạt Động của Momentum

- Định nghĩa
    - Momentum thêm một thành phần "động lượng" vào cập nhật tham số, giúp tăng tốc độ hội tụ và giảm thiểu các vấn đề như mắc kẹt ở các cực tiểu địa phương.
- Phương Trình Cập Nhật
    - Tích Lũy Động Lượng: Động lượng v_t tại bước thời gian t được tính bằng cách kết hợp động lượng trước đó v_{t-1} với gradient hiện tại: $v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$
- Cập Nhật Tham Số: Tham số được cập nhật bằng cách trừ đi động lượng: $\theta = \theta - v_t$
- Trong đó, γ là hệ số động lượng (thường nằm trong khoảng 0.9 đến 0.99), là tỷ lệ học, và $\nabla_\theta J(\theta)$ là gradient của hàm mất mát.

1.2.2.3 Ưu Điểm của Momentum

- Tăng Tốc Hội Tụ: Bằng cách tích lũy thông tin từ các gradient trước đó, Momentum giúp tăng tốc độ hội tụ, đặc biệt là trên các bề mặt mất mát có dạng bằng phẳng hoặc có địa hình phức tạp.
- Giảm Rủi Ro Mắc Kẹt: Giảm thiểu nguy cơ mắc kẹt tại các cực tiểu địa phương, đồng thời vượt qua các vùng mất mát cao hơn.

1.2.2.4 Tinh Chỉnh và Ứng Dụng

- Tinh Chỉnh Tham SốViệc lựa chọn hệ số động lượng γ và tỷ lệ học η cần được tinh chỉnh cẩn thận để đạt hiệu quả tối ưu.
- Ứng Dụng: Phương pháp này rất hiệu quả trong huấn luyện các mạng nơ-ron sâu, đặc biệt trong các tình huống mà bề mặt mất mát có nhiều địa hình phức tạp.

1.2.2.5 Kết Luận

Momentum là một phương pháp tối ưu hóa mạnh mẽ, cung cấp một cách tiếp cận nâng cao so với SGD truyền thống. Nó cải thiện đáng kể tốc độ hội tụ và khả năng vượt qua các điểm không tối ưu trên bề mặt mất mát, làm cho nó trở thành công cụ lựa chọn trong nhiều tình huống học sâu.

Trình bày trên cung cấp một cái nhìn chi tiết về cách hoạt động và lợi ích của phương pháp Momentum, một cải tiến quan trọng của SGD trong lĩnh vực học máy và học sâu.

1.2.3 Phương pháp Adagrad

1.2.3.1 Giới thiệu

Adagrad, viết tắt của "Adaptive Gradient Algorithm", là một phương pháp tối ưu hóa được đề xuất để cải thiện hiệu quả của Gradient Descent. Adagrad điều chỉnh tỷ lệ học dựa trên độ lớn của gradient, giúp tăng cường hiệu quả học tập trong các mô hình học máy.

1.2.3.2 Cơ Chế Hoạt Động

- Điều Chỉnh Tỷ Lệ Học:
  - Cơ Bản: Adagrad thích ứng tỷ lệ học cho mỗi tham số dựa trên lịch sử của gradient. Nó giúp giảm tỷ lệ học cho các tham số với gradient lớn và tăng tỷ lệ học cho các tham số với gradient nhỏ.
  - Lợi Ích: Điều này giúp tối ưu hóa hiệu quả khi đối mặt với dữ liệu thưa thớt hoặc khi các đặc trưng có tần suất xuất hiện không đồng đều.
- Phương Trình Cập Nhật:
  - Tích Lũy Gradient: Đối với mỗi tham số, Adagrad tích lũy bình phương của gradient trong quá khứ: $G_{t,ii} = G_{t-1,ii} + (\nabla_\theta J(\theta_i))^2$
  - Cập Nhật Tham Số: Tham số được cập nhật theo công thức:

    $\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}+\epsilon}} \nabla_\theta J(\theta_i)$

  - Trong đó, η là tỷ lệ học ban đầu, và ϵ là một số nhỏ để tránh chia cho không.
- Ưu Điểm và Hạn Chế
  - Ưu Điểm
    - Hiệu Quả với Dữ Liệu Thưa: Adagrad rất hữu ích khi làm việc với dữ liệu thưa thớt, như trong các bài toán xử lý ngôn ngữ tự nhiên và trích xuất đặc trưng.
    - Tự Động Điều Chỉnh Tỷ Lệ Học: Không cần phải tinh chỉnh tỷ lệ học một cách thủ công.
  - Hạn Chế
    - Giảm Tỷ Lệ Học Quá Nhanh: Trong quá trình tích lũy gradient, tỷ lệ học có thể giảm quá nhanh, dẫn đến việc thuật toán sớm dừng lại trước khi đạt tới điểm tối ưu.

1.2.3.3 Ứng Dụng và Tinh Chỉnh

- Ứng Dụng: Adagrad phù hợp cho các bài toán có dữ liệu thưa và các đặc trưng không đồng đều.

- Tinh Chỉnh: Cần cân nhắc khi sử dụng Adagrad cho các bài toán có quá trình học dài hạn hoặc với dữ liệu không thưa.

1.2.4 Loại bài toán và dữ liệu phù hợp

Loại bài toán: KNN phù hợp với bài toán phân loại và hồi quy.

Dữ liệu phù hợp: Dữ liệu có không gian đặc trưng ít và khi có sự tương tự không gian giữa các điểm dữ liệu đại diện cho sự tương tự về mặt ngữ nghĩa.

Ví dụ: KNN thường được sử dụng trong các bài toán như phân loại hình ảnh, dự đoán giá nhà dựa trên đặc điểm của ngôi nhà, hoặc đề xuất sản phẩm dựa trên sở thích của người dùng.

1.2.4.1 Kết Luận

Adagrad là một phương pháp tối ưu hóa mạnh mẽ, đặc biệt hữu ích trong các bối cảnh học máy cụ

1.2.5 Phương pháp RMSprop

1.2.5.1 Giới thiệu

RMSprop, viết tắt của "Root Mean Square Propagation", là một phương pháp tối ưu hóa được thiết kế để khắc phục một số vấn đề của Adagrad, đặc biệt là việc giảm tỷ lệ học quá nhanh.

1.2.5.2 Cơ Chế Hoạt Động

- Điều Chỉnh Tỷ Lệ Học
  - Cơ Bản: RMSprop thích ứng tỷ lệ học cho mỗi tham số dựa trên lịch sử gần đây của gradient, không tích lũy tất cả lịch sử như Adagrad.
  - Lợi Ích: Điều này giúp ngăn chặn việc giảm tỷ lệ học quá nhanh, đồng thời cho phép thuật toán tiếp tục tiến triển ngay cả sau nhiều bước cập nhật.
- Phương Trình Cập Nhật:

- **Tích Lũy Gradient**: Tích lũy bình phương của gradient, nhưng chỉ tập trung vào gradient gần đây:
  - $E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\nabla_\theta J(\theta))^2$
- **Cập Nhật Tham Số**: Tham số được cập nhật theo công thức:
  - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla_\theta J(\theta)$
- Trong đó, η là tỷ lệ học ban đầu, β là hệ số suy giảm, và ϵ là một số nhỏ để tránh chia cho không.

1.2.5.3 Ưu Điểm và Hạn Chế

- Ưu Điểm
  - Phù Hợp cho Dữ Liệu Không Thưa: Hiệu quả trong việc tối ưu hóa các bài toán với dữ liệu không thưa và trong các mô hình học sâu phức tạp.
  - Ổn Định Hơn: So với Adagrad, RMSprop cung cấp một quá trình học ổn định hơn và ít có nguy cơ giảm tỷ lệ học quá nhanh.
- Hạn Chế
  - Cần Tinh Chỉnh Tham Số: Tùy thuộc vào bài toán, việc tinh chỉnh các tham số như β và η có thể cần thiết để đạt hiệu suất tối ưu.

1.2.5.4 Ứng Dụng và Tinh Chỉnh

- Ứng Dụng Rộng Rãi: RMSprop được sử dụng rộng rãi trong việc huấn luyện các mạng nơ-ron sâu, nhất là trong các tác vụ liên quan đến hình ảnh và xử lý ngôn ngữ tự nhiên.
- Tinh Chỉnh: Cần cân nhắc khi chọn hệ số suy giảm và tỷ lệ học để phù hợp với tính chất cụ thể của dữ liệu và mô hình.

1.2.5.5 Kết Luận

RMSprop là một phương pháp tối ưu hóa mạnh mẽ và linh hoạt, phù hợp với nhiều loại bài toán trong học máy và học sâu. Phương pháp này hiệu quả đặc biệt trong việc xử lý các vấn đề liên quan đến hội tụ chậm hoặc mắc kẹt ở các cực tiểu địa

phương không tối ưu, nhờ vào cách tiếp cận thích ứng tỷ lệ học cho từng tham số dựa trên lịch sử gần đây của gradient. Phương pháp Adam

1.2.6 Giới thiệu

Adam, viết tắt của "Adaptive Moment Estimation", là một phương pháp tối ưu hóa hiệu quả và phổ biến trong học máy và học sâu. Adam kết hợp các ý tưởng từ RMSprop và Momentum để cung cấp một cách tiếp cận tối ưu hóa mạnh mẽ và hiệu quả.

1.2.6.1 Cơ Chế Hoạt Động

- Kết Hợp RMSprop và Momentum
    - RMSprop: Adam sử dụng cách tính toán trung bình di động trượt của bình phương gradient như trong RMSprop.
    - Momentum: Đồng thời, nó cũng tính toán trung bình di động của gradient thực sự, tương tự như Momentum.
- Phương Trình Cập Nhật
    - **Tính toán Moment**: Adam tính toán hai moment - moment đầu tiên (trung bình của gradient) và moment thứ hai (trung bình của bình phương gradient):
        - $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta J(\theta)$
        - $v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta J(\theta))^2$
    - **Hiệu chỉnh Bias**: Các moment được hiệu chỉnh để khắc phục sự thiên vị về 0 ở các bước đầu tiên.
        - $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
        - $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
    - **Cập Nhật Tham Số**: Tham số được cập nhật sử dụng moment hiệu chỉnh:
        - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$

1.2.6.2 Ưu Điểm và Hạn Chế

- Ưu Điểm
    - Tự Điều Chỉnh Tỷ Lệ Học: Tự động điều chỉnh tỷ lệ học cho mỗi tham số, hiệu quả trong nhiều tình huống khác nhau.

- Phù Hợp Với Nhiều Loại Dữ Liệu và Mô Hình: Hiệu quả cao trong nhiều loại bài toán học sâu, từ mạng nơ-ron tích chập đến mạng nơ-ron tuần tự.

- Hạn Chế

  - **Cần Tinh Chỉnh Tham Số**: Tùy thuộc vào bài toán, việc tinh chỉnh các tham số như β_1, β_2 và η có thể cần thiết.

1.2.6.3 Ứng Dụng và Tinh Chỉnh

- Ứng Dụng Rộng Rãi: Adam được sử dụng rộng rãi trong cộng đồng học máy và học sâu, từ tác vụ xử lý ngôn ngữ tự nhiên đến nhận dạng hình ảnh.
- Tinh Chỉnh: Cần xem xét cẩn thận khi chọn các giá trị tham số để đạt hiệu suất tối ưu, đặc biệt trong các bài toán cụ thể.

1.2.6.4 Kết Luận

Adam được coi là một trong những phương pháp tối ưu hóa hiệu quả nhất hiện nay, cung cấp một cách tiếp cận linh hoạt và mạnh mẽ cho nhiều loại bài toán học sâu.

1.3 So sánh các phương pháp optimizer

1.3.1 SGD (Stochastic Gradient Descent)

- Ưu Điểm: Đơn giản và dễ hiểu. Hiệu quả với dữ liệu lớn và bộ nhớ hạn chế.
- Nhược Điểm: Hội tụ chậm; có thể mắc kẹt tại các điểm cực tiểu địa phương.
- Ứng Dụng: Tốt cho dữ liệu lớn và khi cần một giải pháp đơn giản.

1.3.2 Momentum

- Ưu Điểm: Giải quyết vấn đề hội tụ chậm của SGD bằng cách thêm động lượng, giúp "vượt qua" các điểm cực tiểu địa phương.
- Nhược Điểm: Cần tinh chỉnh thêm tham số (hệ số động lượng).
- Ứng Dụng: Hiệu quả trong việc huấn luyện các mạng nơ-ron sâu và vượt qua các điểm cực tiểu địa phương.

1.3.3 Adagrad

- Ưu Điểm: Tự điều chỉnh tỷ lệ học và hiệu quả với dữ liệu thưa.
- Nhược Điểm: Tỷ lệ học có thể giảm quá nhanh và quá sớm, dẫn đến việc học bị dừng lại sớm.
- Ứng Dụng: Phù hợp với các bài toán có đặc trưng thưa thớt như xử lý ngôn ngữ tự nhiên.

1.3.4 RMSprop

- Ưu Điểm: Khắc phục vấn đề giảm tỷ lệ học quá nhanh của Adagrad bằng cách sử dụng trung bình di động.
- Nhược Điểm: Cần tinh chỉnh thêm tham số (hệ số suy giảm).
- Ứng Dụng: Hiệu quả trong các tác vụ học sâu phức tạp, nhất là khi làm việc với dữ liệu không thưa.

1.3.5 Adam

- Ưu Điểm: Kết hợp ưu điểm của RMSprop và Momentum. Tự điều chỉnh tỷ lệ học và thích nghi tốt với nhiều bài toán khác nhau.
- Nhược Điểm: Cần tinh chỉnh nhiều tham số hơn (hệ số moment).
- Ứng Dụng: Rất phổ biến trong huấn luyện mạng nơ-ron sâu, phù hợp với hầu hết các loại bài toán.

CHƯƠNG 2. CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual Learning (Học Liên Tục)

2.1.1 Định Nghĩa

- Continual Learning, còn gọi là Lifelong Learning hoặc Incremental Learning, là quá trình mà một mô hình học máy tiếp tục học và phát triển sau khi được triển khai, bằng cách tích hợp kiến thức mới mà không quên những gì đã học trước đó.
- Mục tiêu là để mô hình có thể thích ứng với những thay đổi trong dữ liệu hoặc môi trường mà không cần phải huấn luyện lại từ đầu.

2.1.2 Vấn Đề Cần Giải Quyết

- Quên Kiến Thức (Catastrophic Forgetting): Đây là hiện tượng một mô hình quên những gì đã học trước đó khi học thông tin mới.
- Transfer Learning và Adaptation: Làm thế nào để mô hình có thể áp dụng kiến thức đã học vào các tình huống mới.

2.1.3 Ứng Dụng

Continual Learning rất quan trọng trong các ứng dụng đòi hỏi mô hình phải thích ứng liên tục với dữ liệu đang thay đổi, như trong nhận diện khuôn mặt, dự báo tài chính, hoặc hệ thống đề xuất sản phẩm.

2.2 Test Production (Kiểm Thử Sản Phẩm)

2.2.1 Định Nghĩa

- Test Production là quá trình kiểm thử mô hình học máy trong một môi trường "sản xuất" thực tế, nơi mô hình sẽ được sử dụng.
- Điều này bao gồm việc kiểm tra hiệu suất, độ chính xác, độ tin cậy và tính ổn định của mô hình khi đối mặt với dữ liệu thực tế.

2.2.2 Các Bước Thực Hiện

- A/B Testing: So sánh hiệu suất của mô hình mới với mô hình hiện tại hoặc với các biến thể khác nhau của mô hình.
- Monitoring và Logging: Theo dõi hoạt động của mô hình để nắm bắt các vấn đề như drift dữ liệu hoặc lỗi dự đoán.
- Feedback Loop: Tạo một vòng lặp phản hồi để liên tục cập nhật và cải thiện mô hình dựa trên phản hồi từ người dùng thực tế.

2.2.3 Tầm Quan Trọng

Đảm bảo rằng mô hình không chỉ hoạt động tốt trên dữ liệu kiểm thử, mà còn phải hiệu quả và đáng tin cậy trong môi trường thực tế.

2.3 Kết Luận

Cả Continual Learning và Test Production đều là những khía cạnh quan trọng trong việc phát triển và triển khai các giải pháp học máy. Trong khi Continual Learning tập trung vào khả năng phát triển và thích nghi liên tục của mô hình, Test Production tập trung vào việc đảm bảo mô hình hoạt động hiệu quả trong môi trường thực tế. Kết hợp cả hai sẽ giúp tạo ra các giải pháp học máy mạnh mẽ, linh hoạt và đáng tin cậy.