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

BÁO CÁO CUỐI KÌ
MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **GV LÊ ANH CƯỜNG**

Người thực hiện: **VÕ THỊ QUẾ CHI – 52000741**

Lớp **: 20050301**

Khoá **: 24**

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

BÁO CÁO CUỐI KỲ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS LÊ ANH CƯỜNG**
Người thực hiện: **VÕ THỊ QUẾ CHI**
Lớp **: 20050301**
Khoá **: 24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

MỤC LỤC

MỤC LỤC 1
CHƯƠNG 1 – ĐỀ BÀI 2
1.1 Câu 1: 2
CHƯƠNG 2 – LÝ THUYẾT 3
2.1 Câu 1: 3
2.1.1 Phương pháp Optimizer: 3
2.1.2 Một số phương pháp Optimizer: 3
2.2 Câu 2: 6
2.2.1 Tìm hiểu về Continual Learning: 6
2.2.2 Tìm hiểu về Test Production: 7

CHƯƠNG 1 – ĐỀ BÀI

1.1 Câu 1:

Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

1) Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy;
2) Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

CHƯƠNG 2 – LÝ THUYẾT

2.1 Câu 1:

2.1.1 Phương pháp Optimizer:

Phương pháp optimizer thường được sử dụng trong ngữ cảnh của machine learning và deep learning để tối ưu hóa các mô hình máy học. Trong quá trình huấn luyện mô hình, mục tiêu là điều chỉnh các trọng số của mô hình để giảm thiểu hoặc tối đa hóa một hàm mất mát (loss function) nào đó, dựa trên dữ liệu huấn luyện.

Các thuật toán optimizer chịu trách nhiệm điều chỉnh các trọng số này theo cách thông minh để đạt được giảm thiểu tối ưu của hàm mất mát.

2.1.2 Một số phương pháp Optimizer:

- Gradient Descent (GD):

- Là phương pháp cơ bản, điều chỉnh trọng số dựa trên độ dốc của hàm mất mát. Có các biến thể như Stochastic Gradient Descent (SGD) và Mini-batch Gradient Descent.
- Nguyên lý hoạt động: GD là phương pháp cơ bản nhất trong tối ưu hóa, cập nhật trọng số bằng cách di chuyển ngược chiều của đạo hàm của hàm mất mát. Cụ thể, trọng số mới được tính theo công thức: $\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$, trong đó η là tốc độ học.
- Ưu điểm: Đơn giản và dễ hiểu.
- Nhược điểm: Chậm khi áp dụng cho tập dữ liệu lớn vì phải tính đạo hàm cho toàn bộ dữ liệu.
- Ví dụ: Bài toán Hồi quy Tuyến tính: Giả sử bạn có tập dữ liệu về diện tích nhà và giá bán tương ứng. Mục tiêu là tìm một đường tuyến tính (hàm dự đoán) sao cho sai số giữa giá thực tế và giá dự đoán là nhỏ nhất. Trong trường hợp này, GD sẽ được sử dụng để điều chỉnh trọng số của đường tuyến tính để làm giảm thiểu hàm mất mát, chẳng hạn như bình phương sai số.

- Stochastic Gradient Descent (SGD):

- Là một thuật toán tối ưu hóa thường được sử dụng trong quá trình đào tạo mô hình máy học và học máy. Nó là một biến thể của thuật toán Gradient Descent (GD). Trong GD, mỗi lần cập nhật tham số của mô hình, toàn bộ tập dữ liệu đào tạo được sử dụng để tính toán gradient của hàm mất mát, và sau đó, các tham số được cập nhật theo hướng ngược với gradient để giảm mất mát.
- Nguyên lý hoạt động: SGD là phiên bản ngẫu nhiên của GD, cập nhật trọng số dựa trên từng mẫu dữ liệu ngẫu nhiên thay vì toàn bộ tập dữ liệu. Điều này giúp nhanh chóng hội tụ và phù hợp với dữ liệu lớn.
- Ưu điểm: Tốc độ học nhanh, phù hợp với dữ liệu lớn.
- Nhược điểm: Dao động nhiều, không ổn định.
- Ví dụ: Phân loại Ảnh: Trong một mô hình học máy dành cho phân loại hình ảnh, SGD có thể được sử dụng để cập nhật trọng số dựa trên từng ảnh một. Mỗi ảnh đóng vai trò như một mẫu đào tạo độc lập, và trọng số được cập nhật ngay sau khi mỗi ảnh được xử lý.

- Momentum:

- Là một kỹ thuật được sử dụng trong quá trình tối ưu hóa khi đào tạo mô hình máy học. Nó giúp tăng tốc quá trình hội tụ bằng cách giảm độ dao động và giúp thuật toán vượt qua các điểm tối thiểu cục bộ.
- Trong thuật ngữ tối ưu hóa, "momentum" thường được mô tả bằng việc giữ lại một phần của hướng cập nhật từ các bước trước đó và thêm vào hướng cập nhật mới. Điều này giúp thách thức vượt qua các điểm thấp và không bị quá mức ảnh hưởng bởi độ dao động.
- Nguyên lý hoạt động: Momentum giúp giảm độ dao động của quá trình học bằng cách tích tụ một lượng động từ các bước trước đó. Công thức cập nhật trọng số là v_{t+1} = βv + (1 − β) ∇J(θ_t),θ_{t+1} = θ_t −ηv_{t+1} , trong đó β là hệ số đà (0 < β < 1).

- Ưu điểm: Giúp vượt qua các điểm yên ngựa, tăng tốc quá trình học.
- Nhược điểm: Cần điều chỉnh hệ số β để tránh overshooting.
- Ví dụ: Học Mô hình Ngôn ngữ Tự nhiên (NLP): Trong học máy dành cho xử lý ngôn ngữ tự nhiên, ví dụ như huấn luyện mô hình dự đoán từ tiếp theo, Momentum có thể giúp giảm dao động và giúp mô hình học nhanh chóng từ dữ liệu ngôn ngữ phức tạp.
  - Adagrad:
- Tự điều chỉnh tỷ lệ học cho từng trọng số dựa trên tần suất xuất hiện của nó trong quá trình huấn luyện.
- Nguyên lý hoạt động: Adagrad điều chỉnh tốc độ học của từng tham số dựa trên tần suất xuất hiện của chúng trong quá trình học. Công thức cập nhật là $G_{t+1} = G_t + \left(\nabla J(\theta_t)\right)^2, \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1}+\epsilon}}$, trong đó G_t là ma trận đường chéo chứa tổng bình phương đạo hàm theo từng tham số.
- Ưu điểm: Hiệu quả cho các tham số có độ dốc lớn.
- Nhược điểm: Tốc độ học giảm quá nhanh do tính chất tích tụ của G_t
- Ví dụ: Học Mô hình Dịch Ngôn ngữ (NMT): Trong bài toán dịch ngôn ngữ, Adagrad có thể được áp dụng để huấn luyện mô hình dịch từ tiếng Anh sang một ngôn ngữ khác. Tại đây, quá trình học sẽ tập trung vào các từ khó dựa trên tần suất xuất hiện của chúng trong tập dữ liệu.
  - RMSprop
- Giảm tỷ lệ học cho các trọng số có độ dốc lớn, giúp hội tụ nhanh hơn.
- Nguyên lý hoạt động: RMSprop là biến thể của Adagrad, giảm ảnh hưởng của tần suất xuất hiện của tham số bằng cách sử dụng trọng số động β. Công thức: $E[g^2]_t = \beta E[g^2]_{t-2} + (1-\beta)\left(\nabla J(\theta_t)\right)^2, \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t+\epsilon}}$
- Ưu điểm: Giảm vấn đề giảm tốc độ học quá nhanh của Adagrad.
- Nhược điểm: Cần điều chỉnh thêm siêu tham số β.

- Ví dụ: Dự đoán Chuỗi Thời gian: Trong bài toán dự đoán chuỗi thời gian như dự báo nhiệt độ hàng ngày, RMSprop có thể được sử dụng để điều chỉnh trọng số mô hình. Điều này giúp kiểm soát tốc độ học và ngăn chặn ảnh hưởng lớn từ các biến đổi không đều trong dữ liệu thời gian.

2.2 Câu 2:

2.2.1 Tìm hiểu về Continual Learning:

- Continual Learning (CL), hay còn gọi là Lifelong Learning, là một lĩnh vực quan trọng trong học máy, nghiên cứu cách mà mô hình có thể liên tục học và cập nhật kiến thức khi có dữ liệu mới mà không quên đi kiến thức đã học trước đó.

- Lý thuyết:

- Plasticity và Stability: Đối với mô hình học máy, plasticity đại diện cho khả năng học mới, trong khi stability đại diện cho khả năng giữ lại kiến thức đã học. Một sự cân bằng giữa plasticity và stability là quan trọng để tránh việc quên đi kiến thức cũ khi học kiến thức mới.

- Catastrophic Forgetting: Hiện tượng này xảy ra khi mô hình quên đi kiến thức đã học khi học kiến thức mới. Continual Learning cố gắng giải quyết vấn đề này.

- Chiến lược Continual Learning:

- Regularization: Sử dụng các kỹ thuật regularization như Elastic Weight Consolidation (EWC) hoặc Learning Without Forgetting (LwF) để giảm thiểu sự thay đổi của trọng số quan trọng cho các nhiệm vụ trước đó.

- Memory Replay: Lưu trữ và tái sử dụng dữ liệu từ quá khứ để đào tạo lại mô hình trên dữ liệu cũ. Điều này giúp giảm thiểu việc quên kiến thức cũ.

- Dynamic Architectures: Sử dụng các kiến trúc mô hình linh hoạt có khả năng mở rộng để học kiến thức mới mà không ảnh hưởng đến kiến thức cũ.

- Đánh giá và đo lường:

- Retention: Đo lường khả năng giữ lại kiến thức cũ sau khi học kiến thức mới.

- Adaptability: Đo lường khả năng của mô hình thích ứng với dữ liệu mới và học các nhiệm vụ mới mà không ảnh hưởng đến hiệu suất trên các nhiệm vụ cũ.

- Ứng dụng:

- Robotics: Trong môi trường thực tế, robot cần học và thích ứng liên tục với môi trường xung quanh.

- Xử lý ngôn ngữ tự nhiên: Đối với các hệ thống xử lý ngôn ngữ tự nhiên, việc học và nâng cấp kiến thức liên tục là quan trọng.

- Xử lý ảnh: Trong ứng dụng nhận dạng hình ảnh, việc học từ dữ liệu mới để nhận ra đối tượng mới là một thách thức của Continual Learning.

2.2.2 Tìm hiểu về Test Production:

- Các bước khi áp dụng Test Production vào quy trình triển khai mô hình học máy:

- Chuẩn bị Dữ liệu:

  + Xác định và chuẩn bị dữ liệu đầu vào cho mô hình.

  + Đảm bảo tính chất và độ chất lượng của dữ liệu để đảm bảo hiệu suất tốt khi triển khai.

- Kiểm Thử Mô Hình:

  + Thực hiện kiểm thử kỹ thuật (unit testing) để đảm bảo rằng mô hình hoạt động đúng trên dữ liệu kiểm thử.

  + Sử dụng dữ liệu kiểm thử riêng để đánh giá hiệu suất và độ chính xác của mô hình.

- Triển Khai Mô Hình:

  + Lựa chọn một môi trường triển khai, có thể là trên cloud, on-premise hoặc trong môi trường edge.

  + Cài đặt mô hình vào môi trường triển khai đã chọn.

- Kiểm Thử Tích Hợp:

  + Kiểm tra tích hợp giữa mô hình và các thành phần khác của hệ thống.

+ Đảm bảo rằng mô hình tương tác chính xác và hiệu quả với các phần mềm và phần cứng xung quanh.

- Giám Sát và Theo Dõi:

  + Thêm các công cụ giám sát để theo dõi hiệu suất của mô hình trong thời gian thực.

  + Xác định ngưỡng cảnh báo để báo cáo vấn đề khi mô hình không hoạt động như mong đợi.

- Tối Ưu Hóa và Cập Nhật:

  + Liên tục tối ưu hóa mô hình dựa trên dữ liệu mới và phản hồi từ môi trường thực tế.

  + Cập nhật mô hình khi cần thiết để giữ cho nó phản ánh đúng với dữ liệu mới.

- Bảo mật:

  + Đảm bảo rằng mô hình và dữ liệu được bảo vệ chặt chẽ để ngăn chặn rủi ro bảo mật.

- Document và Đào Tạo:

  + Tạo tài liệu chi tiết về cách triển khai và vận hành mô hình.

  + Huấn luyện nhóm vận hành và bảo trì về cách sử dụng và duy trì mô hình.

- Ví dụ bài toán:

Bài toán: Dự đoán Churning (Từ chối sử dụng dịch vụ) trong Dịch vụ di động

Mục tiêu: Dự án nhằm xây dựng một mô hình học máy để dự đoán khả năng churning của khách hàng dịch vụ di động, tức là khả năng họ sẽ từ chối sử dụng dịch vụ trong tương lai.

- Bước 1: Chuẩn bị dữ liệu:

  + Sử dụng dữ liệu lịch sử về sử dụng dịch vụ, thông tin tài khoản, và các chỉ số liên quan.

+ Tiền xử lý dữ liệu để xử lý giữ liệu thiếu, loại bỏ nhiễu, và chuẩn hóa các đặc trưng.

- Bước 2: Kiểm thử mô hình:

  + Chia tập dữ liệu thành tập huấn luyện và tập kiểm thử.

  + Xây dựng một mô hình dự đoán khả năng churning, ví dụ như mô hình Logistic Regression hoặc Random Forest.

  + Đánh giá mô hình trên tập kiểm thử để đảm bảo rằng nó có hiệu suất tốt và không bị quá mức đào tạo (overfitting).

- Bước 3: Triển khai mô hình:

  + Lựa chọn một môi trường triển khai, chẳng hạn như một hệ thống dịch vụ web.

  + Triển khai mô hình vào môi trường thực tế và tích hợp nó với hệ thống tổng thể.

- Bước 4: Kiểm thử tích hợp:

  + Kiểm tra tích hợp giữa mô hình và các thành phần khác của hệ thống, như cơ sở dữ liệu khách hàng và hệ thống thông báo.

  + Đảm bảo rằng dữ liệu đầu vào từ hệ thống thực tế được xử lý đúng.

- Bước 5: Giám sát và theo dõi:

  + Thêm các công cụ giám sát để theo dõi hiệu suất của mô hình trong thời gian thực.

  + Xác định ngưỡng cảnh báo để báo cáo vấn đề khi mô hình không hoạt động như mong đợi.

- Bước 6: Tối ưu hóa và cập nhật:

  + Liên tục theo dõi và cập nhật mô hình dựa trên dữ liệu mới và phản hồi từ môi trường thực tế.

  + Tối ưu hóa mô hình để giữ cho nó phản ánh đúng với thay đổi trong hành vi của khách hàng.

- Bước 7: Bảo mật:

  + Đảm bảo rằng mô hình và dữ liệu được bảo vệ chặt chẽ để ngăn chặn rủi ro bảo mật.

- Bước 8: Document và đào tạo:

  + Tạo tài liệu chi tiết về cách triển khai và vận hành mô hình.

  + Huấn luyện nhóm vận hành và bảo trì về cách sử dụng và duy trì mô hình.