TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Ngọc Lan Như - Hoàng Minh Quân

HUẤN LUYỆN MẠNG NƠ-RON NHIỀU TẦNG ẨN BẰNG THUẬT TOÁN ADAM

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 07/2021

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Ngọc Lan Như - 1712644
Hoàng Minh Quân - 1712688

HUẤN LUYỆN MẠNG NƠ-RON NHIỀU TẦNG ẨN BẰNG THUẬT TOÁN ADAM

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN
ThS. Trần Trung Kiên

Tp. Hồ Chí Minh, tháng 07/2021

Lời cảm ơn

Tôi xin chân thành cảm ơn ...

Mục lục

Danh mục hình ảnh

Danh mục bảng

Chương 1

Giới thiệu

Việc sử dụng mạng nơ-ron nhiều tầng ẩn trong các ứng dụng trí tuệ nhân tạo ngày càng chiếm một vị trí quan trọng khi mà các bài toán khó của học máy truyền thống trong việc xử lý ảnh và video, xử lý ngôn ngữ tự nhiên như: dịch máy tự động, nhận diện mặt người, phát hiện đồ vật,... đã phần nào được giải quyết và ngày càng được cải tiến.

Mạng nơ-ron nhiều tầng ẩn là một mạng nơ-ron nhân tạo gồm ba loại tầng chính: tầng nhập, các tầng ẩn, và tầng xuất. Việc có nhiều tầng ẩn cũng là một trong những lý do giúp cho mạng nơ-ron nhiều tầng ẩn có thể giải quyết các bài toán khó mà các thuật toán học máy truyền thống không giải quyết được. @Untitled lý giải rằng với nhiều tầng ẩn, mạng nơ-ron có thể "học" được từ đặc trưng đơn giản (low-level features) thành các đặc trưng phức tạp hơn (high-level features). Nhờ có các đặc trưng trung gian trên mà mạng nơ-ron nhiều tầng ẩn có thể tổng quát hoá tốt trên các bài toán phức tạp trong lĩnh vực trí tuệ nhân tạo.

Một mạng nơ-ron nhiều tầng ẩn được xem là tổng quát hoá tốt trên một bài toán khi sự sai khác giữa giá trị dự đoán của mạng nơ-ron và giá trị nhãn của dữ liệu ngoài tập huấn luyện là đủ nhỏ (trong bài toán huấn luyện có giám sát). Để đạt được kết quả đó, mạng nơ-ron nhiều tầng ẩn cần đi tìm một bộ trọng số phù hợp cho từng bài toán cụ thể. Việc đi tìm bộ trọng số này được thực hiện thông qua quá trình tối ưu hoá mạng

nơ-ron nhiều tầng ẩn. Vì thế việc tối ưu hoá mạng nơ-ron là cần thiết.

Bài toán tối ưu hoá mạng nơ-ron nhiều tầng ẩn nhận dữ liệu nhập là hàm chi phí nhận bộ trọng số của mạng nơ-ron nhiều tầng ẩn làm tham số. Hàm chi phí nói ở đây là hàm cho biết sự sai lệch giữa kết quả dự đoán của mạng nơ-ron so với giá trị đúng. Giá trị sai lệch này còn được gọi là độ lỗi. Sau quá trình tối ưu ta mong muốn có được một bộ trọng số của mạng nơ-ron nhiều tầng ẩn cho độ lỗi trong cả hai tập dữ liệu huấn luyện và tập dữ liệu kiểm tra là đủ nhỏ.

Việc tối ưu hoá mạng nơ-ron có thể hiểu là quá trình đi tìm cực tiểu của hàm chi phí bằng cách thay đổi giá trị của bộ trọng số. Từ đó, ta cũng có thể hiểu quá trình tối ưu hoá mạng nơ-ron là quá trình di chuyển trong mặt phẳng lỗi để tìm được cực tiểu mang giá trị độ lỗi đủ nhỏ. Tuy nhiên, việc di chuyển trong mặt phẳng lỗi gặp nhiều khó khăn. @Untitled cho thấy rằng sự hỗn loạn của mặt phẳng lỗi ngày càng tăng khi số tầng ẩn trọng mạng nơ-ron ngày càng tăng. Sự hỗn loạn này được cấu thành từ nhiều cực tiểu địa phương, các điểm yên ngựa, vùng bằng phẳng, và vùng rãnh hẹp.

Các điểm cực tiểu địa phương từng được xem như là nguyên nhân chính gây ra sự khó khăn trong việc tối ưu hoá mạng nơ-ron có nhiều tầng ẩn, các bài báo như @Untitled đã chỉ ra rằng cực tiểu địa phương không phải là nguyên nhân chính dẫn đến sự hội tụ chậm trong quá trình huấn luyện mà là các điểm yên ngựa được bao bọc bởi vùng bằng phẳng. Vì thế, việc tránh các điểm này và tìm được cách thoát ra khỏi điểm yên ngựa là quan trọng của các thuật toán tối ưu.

Chương 2

Kiến thức nền tảng

2.1 Mạng nơ-ron nhiều tầng ẩn

Placeholder

2.2 Quá trình huấn huyện mạng nơ-ron nhiều tầng ẩn

Placeholder

2.3 "Gradient Descent"

Placeholder

2.3.1 "Batch" Gradient Descent

Placeholder

2.3.2 "Stochastic" Gradient Descent

Placeholder

Chương 3

Huấn luyện mạng nơ-ron nhiều tầng ẩn bằng thuật toán Adam

3.1 Thuật toán Gradient Descent with Momentum

Placeholder

3.2 Thuật toán Gradient Descent với tỉ lệ học thích ứng

Placeholder

3.3 Thuật toán Adam

Placeholder

Chương 4

Các kết quả thí nghiệm

Trong chương này, chúng tôi trình bày các kết quả thí nghiệm nhằm đánh giá những nội dung đã trình bày ở chương 3. Cho các thí nghiệm về nguyên lý, chúng tôi sử dụng dữ liệu được tạo ngẫu nhiên và hàm lỗi Mean Squared Error; với các thí nghiệm thực tế, chúng tôi sử dụng bộ dữ liệu MNIST và CIFAR10 và hàm lỗi Cross Entropy. Kết quả thí nghiệm cho thấy thuật toán mà chúng tôi cài đặt có thể xấp xỉ kết quả mà bài báo công bố, tuy nhiên chúng tôi nhận thấy rằng bộ siêu tham số được sử dụng để tái tạo kết quả có thể không cho kết quả tốt nhất cho tất cả thuật toán. Ngoài ra, các kết quả thí nghiệm cũng cho thấy các trường hợp mà các thuật toán khác gặp khó khăn, và cách mà Adam vượt qua các khó khăn đó. Cuối cùng, các thí nghiệm cho thấy tốc độ tối ưu hóa của các thuật toán trên các mô hình mạng nơ-ron thực tế với nhiều tầng ẩn gồm các cấu trúc khác nhau.

4.1 Các thiết lập thí nghiệm

Chúng tôi sử dụng ngôn ngữ lập trình Python và thư viện Numpy cho các thí nghiệm nguyên lý, thư viện Pytorch cho các thí nghiệm thực tế. Cả thư viện Numpy và Pytorch đều cung cấp khả năng tăng tốc thực thi bằng việc véc-tơ hóa tính toán trên nền C/C++. Thư viện Numpy tập

trung vào các chức năng tính toán đại số trên CPU, phù hợp với các thí nghiệm đơn giản; trong khi thư viện Pytorch là một thư viện máy học có thể xây dựng các mạng nơ-ron nhiều tầng ẩn phức tạp cùng với khả năng thực thi song song trên GPU. Loại GPU mà chúng tôi sử dụng là NVIDIA RTX 2080.

Chúng tôi sử dụng ngôn ngữ lập trình Python và thư viện Numpy cho các thí nghiệm nguyên lý, thư viện Pytorch cho các thí nghiệm thực tế. Cả thư viện Numpy và Pytorch đều cung cấp khả năng tăng tốc thực thi bằng việc véc-tơ hóa tính toán trên nền C/C++. Thư viện Numpy tập trung vào các chức năng tính toán đại số trên CPU, phù hợp với các thí nghiệm đơn giản; trong khi thư viện Pytorch là một thư viện máy học có thể xây dựng các mạng nơ-ron nhiều tầng ẩn phức tạp cùng với khả năng thực thi song song trên GPU. Loại GPU mà chúng tôi sử dụng là NVIDIA RTX 2080, ngoài ra chúng tôi cũng sử dụng NVIDIA Tesla T4 trên nền tảng Google Colaboratory.

4.2 Các kết quả thí nghiệm

4.2.1 Kết quả của thuật toán cài đặt so với bài báo

Placeholder

4.2.2 Phân tích trường hợp bề mặt lỗi align và không align với tham số

Placeholder

4.2.3 Vấn đề dữ liệu thưa và nhiều lỗi

Placeholder

4.2.4 Thử nghiệm trên mô hình VGG16

Placeholder

4.2.5 Thử nghiệm trên mô hình RNN

Placeholder

Chương 5

Kết luận và hướng phát triển

5.1 Kết luận

Placeholder

5.2 Hướng phát triển

Placeholder

Tài liệu tham khảo