1. Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

Opimizer là một phần quan trọng trong việc huấn luyện mô hình học máy. Các phương pháp Optimizer là các thuật toán được dùng để điều chỉnh các tham số của model trong quá trình huấn luyện mô hình, nhằm giảm thiểu hàm mất mát (loss function) của mô hình đó. Những phương pháp Optimizer khác nhau đều có những quy luật cập nhật tham số, learning rate riêng để có thể dưa ra mô hình được tối ưu nhất. Vì vậy việc chọn phương pháp Optimizer hợp lý cho mô hình là cần thiết. Các phương pháp đó bao gồm:

* Gradient Descent Optimizer:

Đây là một trong những phương pháp optimizer cơ bản và phổ biến trong huấn luyện mô hình học máy. Ý tưởng là sử dụng gradient của hàm mất mát để điểu chỉnh các tham số mô hình, từ đó tối ưu hóa giá trị hàm mất mát. Về cơ bản, thuật toán này hoạt động như sau:

* + Thuật toán tính gradient của hàm mất mát đối với các tham số hiện tại.
  + Bắt đầu tại một trọng số trên hàm gradient, và tiến hành di chuyển các tham số theo hướng ngược gradient. Công thức để cập nhật trọng số được biểu diễn như sau:

Trong đó:  
*θ* là tham số cần cập nhật.  
 là learning rate, quyết định độ lớn của bước di chuyển.  
 là gradient của hàm mất mát *F* theo tham số θ.

* + Lặp lại quá trình trên cho đến khi đạt được độ chính xác mong muốn hoặc đạt số epoch tối đa.

Thuật toán này hoạt động tốt cho hầu như mọi trường hợp, tuy nhiên việc tính gradient đối với các dữ liệu lớn tốn tương đối nhiều thời gian và tài nguyên. Và đối với các hàm mất mát không lồi, thuật toán này không biết quãng đường phải đi trên hàm gradient.

* Stochastic Gradient Descent Optimizer:

Đây là một biến thể của Gradient Descent phổ biến trong huấn luyện mô hình học máy, đặc biệt đối với dữ liệu có kích thước lớn. Khác với Gradient Descent, thuật toán này chọn ra một mẫu dữ liệu ngẫu nhiên từ tập huấn luyện, sau đó tiến hành Gradient Descent lên mẫu dữ liệu đó, sau đó chọn điểm khác và lặp lại cho đến khi thuật toán kết thúc.

Bởi vì thuật toán này không sử dụng hết tập dữ liệu nên kết quả đưa ra nhiễu hơn so với kết quả có được từ Gradient Descent. Vì vậy Stochastic Gradient Descent sử dụng nhiều vòng lặp tại nhiều điểm dữ liệu khác nhau, nhưng khi so với Gradient Descent thời gian tính toán vẫn được tiết kiệm hơn nhiều. Thế nên nếu dữ liệu có kích thước đủ lớn và thời gian tính toán là yếu tố quan trọng thì Stochastic Gradient Descent vẫn được ưu tiên hơn so với Gradient Descent.

* Adaptive Gradient Optimizer (Adagrad):

Đây là một trong những thuật toán Optimizer điều chỉnh learning rate dựa trên tham số của mô hình. Trong quá trình huấn luyện, learning rate được thay đổi dựa trên sự khác nhau giữa các tham số. Tham số thay đổi càng lớn, learning rate thay đổi càng nhỏ. Thuật toán sử dụng công thức sau để cập nhật các trọng số:

Trong đó:

là giá trị tham số được cập nhật ở bước tiếp theo.

là learning rate.

là tổng bình phương của gradient cho tham số tính từ bước đầu tiên đến bước hiện tại.

là gradient hiện tại của tham số.

là một giá trị nhỏ được thêm vào mẫu số để tránh việc chia cho 0.

Phương pháp này cho phép ta thực hiện thuật toán mà không cần phải điểu chỉnh learning rate bằng tay, cũng như đáng tin cậy hơn so với các thuật toán khác. Điểm trừ của thuật toán này là learning rate có thể bị giảm một cách nhanh chóng và có thể đạt đến ngưỡng cực kì nhỏ, và điều này sẽ dẫn đến mô hình không thể học thêm và độ chính xác bị sai hoàn toàn.

* RMS Prop Optimization (Root Mean Squared Propagation):

Đây là một thuật toán Optimizer được sử dụng để điều chỉnh learning rate của mỗi tham số. Thuật toán này được coi là cải thiện từ phương pháp Adagra, bằng cách hạn chế sự giảm đột ngột của learning rate. Ý tưởng chính của RMSprop là điều chỉnh learning rate riêng cho từng tham số của mô hình bằng cách sử dụng thông tin về sự biến thiên của gradient theo thời gian. Công thức cập nhật trong RMS Prop cho mỗi tham số θ trong gradient:

Trong đó:

là giá trị tham số được cập nhật ở bước tiếp theo.

là learning rate.

là moving average của bình phương gradient.

là gradient hiện tại của tham số.

là một giá trị nhỏ được thêm vào mẫu số để tránh việc chia cho 0.

* Gradient Descent with Momentum Optimizer:

Là một biến thể của thuật toán Gradient Descent, sử dụng momentum để cải thiện hiệu suất. Momentum là một kỹ thuật quan trọng trong các thuật toán tối ưu hóa trong học máy. Momentum được sử dụng để giảm bớt dao động và tăng tốc độ hội tụ của quá trình tối ưu hóa. Cơ chế của thuật toán hoạt động như sau:

* + Trong mỗi vòng lặp cập nhật tham số, momentum giữ lại một phần của giá trị trước đó và kết hợp nó với gradient hiện tại để di chuyển tham số sang hướng tối ưu. Công thức để cập nhật giá trị như sau:

Trong đó:  
 là vận tốc tại bước thời gian .

là hệ số momentum, thường được đặt trong khoảng từ 0.8 đến 0.999.

là gradient của hàm mất mát tại thời điểm (tham số hiện tại).

là learning rate.

là giá trị tham số được cập nhật.

* + Sau khi cập nhật, thuật toán giảm bớt giao động nhằm tránh hiện tượng dao động quanh điểm tối ưu và giúp mô hình đạt được điểm cực tiểu.

Gradient Descent with Momentum giúp tăng tốc độ và ổn định quá trình hội tụ của Gradient Descent. Thuật toán này phát huy trong trường hợp mô hình đối mặt với các hàm phức tạp và chứa các điểm cực tiểu cục bộ. Điều này làm cho việc tối ưu hóa trở nên hiệu quả hơn và giúp thuật toán tìm được điểm tối ưu một cách nhanh chóng hơn.

* AdaDelta Optimizer

Là một thuật toán Optimizer trong huấn luyện mô hình học máy, đặc biệt được sử dụng trong việc huấn luyện mạng neural sâu. Thuật toán này tập trung vào việc điều chỉnh learning rate một cách tự động và thích ứng dựa trên lịch sử của các bước cập nhật trước đó.

AdaDelta có thể được xem như là bản nâng cấp từ AdaGrad. Vấn đề chính của AdaGrad và RMS Prop là phải cung cấp giá trị learning rate ban đầu, vấn đề còn lại là learning rate có thể giảm xuống mức cực thấp, dẫn đến tình trạng không học thêm của mô hình.

Để giải quyết những vấn đề trên, AdaDelta sử dụng hai biến trạng thái để lưu lại sự biến đổi của gradient và tham số của mô hình.

* Adam Optimizer (Adaptive Moment Estimation):

Là một thuật toán Optimizer phổ biến được sử dụng trong huấn luyện mô hình học máy. Thuật toán này được coi là bản mở rộng từ thuật toán Stochastic Gradient Descent, khi nó có thể điều chỉnh được learning rate, và được thiết kế để cập nhật các trọng số của mạng neural trong quá trình huấn luyện.

Bằng cách kết hợp mean và uncentered variance của gradient, Adam đạt được một learning rate có thể thích nghi với các hàm phức tạp trong quá trình huấn luyện. Đặc tính này giúp quá trình hội tụ nhanh hơn và tăng cường hiệu suất của mạng neural.

Công thức cập nhật của Adam:

và lần lượt là các moving average của gradient (m) và bình phương gradient (v) tại bước thời gian. là gradient của hàm mất mát tại thời điểm (tham số hiện tại). và là hệ số momentum và hệ số trong RMS Prop, thường lần lượt là khoảng từ 0.8 đến 0.999.

Sau đó, Adam sử dụng các moving average đã được tính để cập nhật tham số:

Trong đó, và là giá trị được điều chỉnh để khắc phục việc ưu tiên tại các bước ban đầu trong quá trình tối ưu hóa. là learning rate, và ϵ là một giá trị nhỏ để tránh việc chia cho 0.

Khi tối ưu hóa mô hình, thuật toán Adam thường có hiệu suất tốt, đặc biệt là trong huấn luyện mạng neural, vì nó kết hợp các ưu điểm của cả momentum và RMS Prop. Tuy nhiên, việc điều chỉnh các siêu tham số , , cũng có thể ảnh hưởng đến hiệu suất của thuật toán.

1. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

**Continual Learning:**

Continual Learning, hay còn gọi là Lifelong Learning, là khái niệm khi mà một hệ thống có khả năng tiếp thu và học từ các chuỗi thông tin liên tục, với lượng thông tin được cung cấp theo thời gian và áp dụng kiến thức từ dữ liệu mới để có được các kĩ năng mà không được định trước. Và quan trọng hơn, quá trình tiếp thu những lượng thông tin mới không làm mất đi hay quên đi những kiến thức đã học trước đó.

Một mô hình Continual Learning nên có khả năng:

- Học và hoạt động trong môi trường mở, nơi không chỉ dùng mô hình hoặc kiến thức đã học để giải quyết vấn đề, mà còn khám pháp ra vấn đề mới, tức là tự tạo ra kiến thức mới để học.

* Học để cải thiện hiệu suất của mô hình, tức là giống như trường hợp khi mà ta hoàn thành khóa đào tạo nghề nghiệp, ta vẫn học để có thể cải thiện hơn trong công việc đó.

Nếu thiếu những khả năng trên, một mô hình Continual Learning sẽ không thể học kiến thức mới, hay hoạt động trong một môi trường mở, từ đó dẫn đến một hệ thống không đủ thông minh.

Những vấn đề gặp phải trong Continual Learning:

* Catastrophic Forgetting: Đây là vấn đề chính cần giải quyết trong continual learning. Các mô hình học máy truyền thống thường gặp phải hiện tượng này, khi mà kiến thức mới sau khi học làm mất đi kiến thức cũ có được từ các tập dữ liệu trước.
* Tăng trưởng dữ liệu: Khi các mô hình continual learning tiếp tục học hỏi, lượng dữ liệu mà chúng xử lý và tiếp thu tăng lên, dẫn đến các vấn đề về hiệu suất và khả năng mở rộng, cũng như kiểm soát tài nguyên cho hệ thống.
* Tính biến đổi của dữ liệu: Dữ liệu mới mà các mô hình continual learning được huấn luyện có thể khác với dữ liệu cũ mà chúng đã được đào tạo, từ đó có thể ảnh hưởng xấu đến độ chính xác và hiệu suất.

Để có thể giải quyết những vấn đề trên, có thể áp dụng một số phương pháp như sau lên mô hình Continual Learning:

* Phân cụm dữ liệu và huấn luyện đa nhiệm: Kỹ thuật này phân chia quá trình huấn luyện thành các giai đoạn khác nhau. Trong mỗi giai đoạn, mô hình chỉ được huấn luyện trên một nhiệm vụ cụ thể. Điều này có thể giúp giảm thiểu Catastrophic Forgetting bằng cách ngăn chặn mô hình học các nhiệm vụ mới quá nhanh.
* Bảo vệ bộ nhớ: Sử dụng bộ nhớ ngoài để lưu trữ thông tin về các nhiệm vụ cũ. Mô hình có thể sử dụng bộ nhớ này để truy cập thông tin về các nhiệm vụ cũ khi cần thiết.

**Test Production:**

Test Production là quá trình kiểm tra một mô hình học máy trong môi trường sản xuất. Quá trình này bao gồm kiểm thử quá trình thực thi của mô hình trong môi trường thực tế, cũng như đánh giá mức đáp ứng các yêu cầu về hiệu suất, độ chính xác và khả năng mở rộng hay không.

Quá trình này là một bước quan trọng trong quy trình phát triển một mô hình học máy có hiệu quả, bằng cách giúp đảm bảo rằng mô hình sẽ hoạt động tốt trong môi trường sản xuất và đáp ứng đủ nhu cầu của người dùng hệ thống.

Một số bước cụ thể để thực hiện quá trình Test Production được thực hiện như sau:

* Xác định các yêu cầu: Bước đầu tiên là xác định các yêu cầu về hiệu suất, độ chính xác và khả năng mở rộng của mô hình.
* Lựa chọn phương pháp Test Production: Bước tiếp theo là lựa chọn phương pháp Test Production phù hợp với nhu cầu người dùng. Hai phương pháp chính có thể sử dụng là sử dụng môi trường thử nghiệm sản xuất hoặc sử dụng môi trường sản xuất thực tế.
* Tạo các kịch bản kiểm tra: Bước tiếp theo là tạo các kịch bản kiểm tra để kiểm tra các yêu cầu đã xác định.
* Thực thi các kịch bản kiểm tra: Bước cuối cùng là thực thi các kịch bản kiểm tra và ghi lại kết quả.

GitHub Bài 1: [Cerberoseee/machine-learning-final-invi (github.com)](https://github.com/Cerberoseee/machine-learning-final-invi)

GitHub Bài 2: [Cerberoseee/machine-learning-final (github.com)](https://github.com/Cerberoseee/machine-learning-final)