Thuật toán tối ưu (optimizer) mô hình học máy là một bước quan trọng trong việc huấn luyện mô hình. Chúng là bước đầu để xây dựng nên một mô hình mạng neural dùng để học các feature hay pattern từ dữ liệu đầu vào, tìm weight và bias phù hợp và tối ưu model.

Có nhiều phương pháp optimizer, mỗi phương pháp sẽ ảnh hưởng đến tốc độ và hiệu suất khác nhau. Dưới đây là một số phương pháp optimizer dùng để huấn luyện mô hình học máy:

1. Gradient descent

Khi tìm giá trị nhỏ nhất của 1 hàm số trong bài toán tối ưu, ta thường tìm giá trị đạo hàm của nó bằng 0. Nhưng đối với các bài toán phức tạp hay nhiều biến, việc tìm đạo hàm rất khó, nhiều lúc bất khả thi, từ đó hình thành nên cách thức tìm tiệm cận đến điểm cực tiểu của hàm số trong một vòng lặp. Cách thức này gọi là Gradient descent (giảm dần độ dốc), sử dụng gradient của hàm mất mát

Công thức :

Trong đó: xt+1: các tham số của model tại thời gian t+1

xt: các tham số của model tại thời gian t

learnrate: trọng số tốc độ học, kiểm soát bước đếm giữa các tham số

gradient: gradient của hàm mất mát trong thời gian t

1. Stochastic gradient descent

Là biến thể của gradient descent. Thay vì sau mỗi vòng lặp ta cập nhật weight 1 lần thì nếu có N điểm dữ liệu thì ta sẽ cập nhật trọng số N lần. Biến thể này cập nhật các tham số bằng cách áp dụng gradient với tập con ngẫu nhiên của train data

Công thức: , với điểm dữ liệu ngẫu nhiên (xi, yi)

Trong đó: xi: điểm dữ liệu ngẫu nhiên

yi: giá trị tương ứng với điểm dữ liệu ngẫu nhiên

1. Momentum/Gradient descent with momentum

Phương pháp momentum đưa động lượng vào phương trình để tăng tốc độ hội tụ, nhất là trong các trường hợp có gradient thưa hoặc nhiều noise. Phương pháp này khắc phục 2 nhược điểm của gradient descent: learning rate và điểm dữ liệu đầu.

Công thức :

Trong đó: vt+1: động lượng tại thời gian t+1

momentum: hệ số động lượng, kiểm soát gradient

vt: động lượng tại thời điểm t

1. Adaptive gradient (Adagrad)

Các phương pháp đều có learning rate không đổi trong quá trình train data. Phương pháp adagrad biến learning rate thành 1 tham số và điều chỉnh learning rate cho từng tham số trong data dựa trên các gradient trước sau mỗi thời điểm t

Công thức :

Trong đó: Gt+1: gradient bình phương tích lũy tại thời gian t+1

Gt: ma trận chéo của tổng bình phương gradient

ϵ: hằng số nhỏ để tránh lỗi (trường hợp chia cho 0)

1. Root mean square propagation (RMSprop)

RMSprop giải quyết vấn đề learning rate giảm dần của phương pháp adagrad bằng cách chia learning rate cho căn bậc hai của bình phương các gradient trước

Công thức :

Trong đó: E[g2]t: Trung bình exponential move của bình phương gradient ở thời gian t

decay: Hệ số decay, thêm vào bình phương gradient hiện tại

1. Adaptive moment estimation (Adam)

Adam là sự kết hợp của momentum và RMSprop, duy trì trung bình các gradient trước và các gradient bình phương

Công thức:

Trong đó: mt+1: Ước tính thời điểm thứ 1 tại thời gian t+1

vt+1: Ước tính thời điểm thứ 2 tại thời gian t+1

m\*t+1: Ước tính thời điểm thứ 1 đã điều chỉnh độ sai lệch tại thời gian t+1

v\*t+1: Ước tính thời điểm thứ 1 đã điều chỉnh độ sai lệch tại thời gian t+1

Qua các phương pháp optimizer trên, ta có thể so sánh chúng với các tiêu chí như sau:

* Tốc độ: Adam > RMSprop > Momentum > SGD > Adagrad > Gradient descent (Adam và RMSprop nhanh nhất do có yếu tố động lượng)
* Xử lý dữ liệu noise: Adam > RMSprop > Momentum > SGD > Gradient Descent > Adagrad (Learning rate thích ứng của Adam và RMSprop khó thay đổi theo gradient noise hơn)
* Ứng dụng vào dữ liệu thưa: Adagrad > RMSprop > Momentum > Adam > SGD > Gradient Descent (Learning rate thích ứng của Adagrad có lợi thế hơn cho các trường hợp có dữ liệu thưa thớt)
* Độ tối ưu trong tính toán: Adam > RMSprop > Momentum > SGD > Adagrad > Gradient Descent (Adam và RMSprop sử dụng learning rate thích ứng nên thường sẽ có ít lần lặp hơn các phương pháp khác)

Dựa trên các tiêu chí đó, ta thấy rằng Adam là phương pháp hoạt động khá tốt, tính toán mức tối thiểu nhanh hơn so với các phương pháp khác.