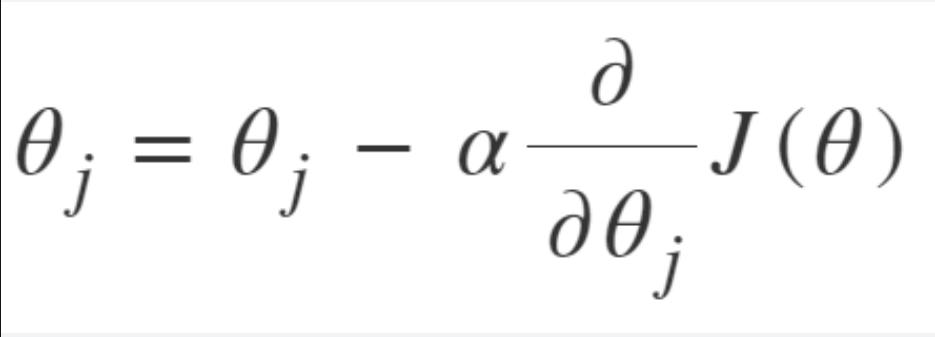
1. **Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy;**

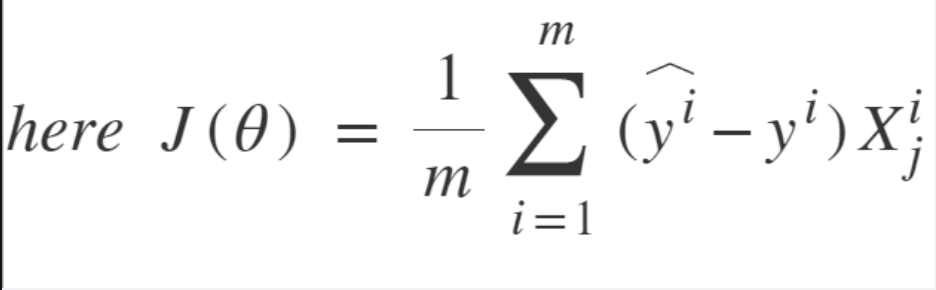
* Types of Optimizers
  + Batch Gradient Descent
  + Stochastic Gradient Descent
  + Mini-Batch Gradient Descent
  + RMSProp
  + Adam
  + AdaDelta
  + Adagra

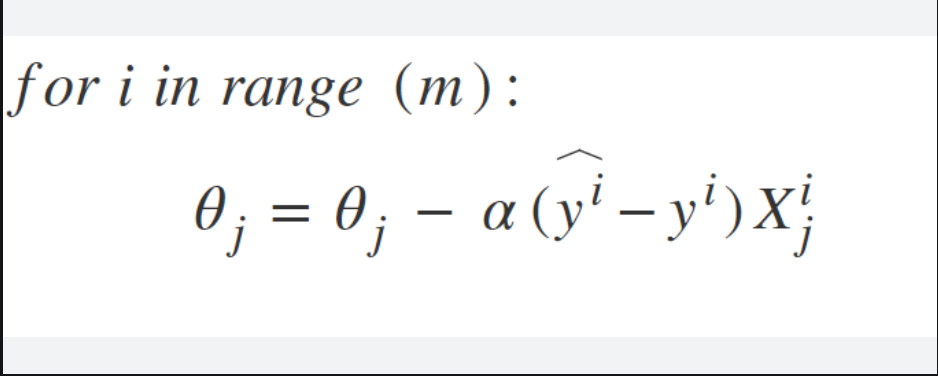
* Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) là một biến thể của Gradient Descent trong quá trình tối ưu hóa tham số trong mô hình máy học. Trong SGD, thay vì cập nhật trọng số (weights) một lần sau mỗi epoch, chúng ta cập nhật trọng số N lần trong mỗi epoch, với N là số lượng điểm dữ liệu trong tập huấn luyện.

Mặt khác, SGD có thể làm giảm tốc độ của một epoch do cần thực hiện nhiều cập nhật trọng số trong một epoch. Tuy nhiên, từ một góc độ khác, SGD có khả năng hội tụ nhanh chóng chỉ sau vài epoch. Công thức của SGD tương tự như Gradient Descent, nhưng được áp dụng trên từng điểm dữ liệu.







* Ưu điểm :

- Dễ triển khai, khả năng thực thi nhanh, hiệu quả về mặt bộ nhớ, tránh các cực tiểu cục bộ

* Nhược điểm :

- Noisy update: Cập nhật nhiễu làm cho quá trình cập nhật thiếu ổn định dẫn đến lao động quanh mức tối thiểu

- Do cập nhật nhiễu, SGD có thể không hội tụ về mức cực tiểu toàn cục chính xác và có thể dẫn đến giải pháp dưới mức tối ưu

|  |  |
| --- | --- |
| Batch Gradient Descent | Stochastic Gradient Descent |
| Tính gradient trên toàn bộ các mẫu dữ liệu training | Tính gradient trên 1 mẫu dữ liệu training |
| Chậm | Nhanh |
| Không được đề xuất để xử lý các mẫu training lớn | Đặc biệt hiệu quả khi có dữ liệu lớn vì chỉ cần tính toán trên một mẫu thay vì toàn bộ tập dữ liệu. |
| Có tính chất xác định trước | Có tính chất ngẫu nhiên |
| Không cần phải xáo trộn | Mẫu thứ tự ngẫu nhiên bị xáo trộn sau mỗi epoch |
| Không dễ tránh các cực tiểu cục bộ | Dễ tránh các cực tiểu cục bộ hơn |
| Hội tụ chậm | Hội tụ nhanh |
| Chỉ cập nhật tham số khi xử lý toàn bộ training set | Cập nhật ở mỗi điểm dữ liệu riêng lẻ |
| Learning rate cố định và không thể thay đổi trong quá trình training | Learning Rate có thể thay đổi linh hoạt trong quá trình training |
| Thường thì hội tụ đến global minimum đối với loss function | Nó có thể hội tụ đến điểm cực tiểu cục bộ hoặc điểm yên ngựa. |
| Có thể bị Overfitting nếu mô hình quá phức tạp so với bộ dữ liệu | Giúp giảm Overfitting bằng cách cập nhật tham số mô hình thường xuyên hơn |

* RMSProp (Root Mean Square Propagation):

RMSProp là một phương pháp tối ưu hóa được thiết kế để thích ứng tỷ lệ học tập cho từng tham số trong quá trình huấn luyện mô hình học máy. Nó giúp giảm hiện tượng vanishing gradients và cho phép mô hình học tập hiệu quả hơn trên dữ liệu thưa (sparse).

1. **Bình Phương Trung Bình Của Bình Phương Độ Dốc:**

RMSProp tính bình phương trung bình của bình phương độ dốc để theo dõi biến động của độ dốc theo thời gian. Điều này giúp ổn định quá trình học.

Ở đây, ) là độ dốc của hàm mất mát tại bước thời gian t, là bình phương trung bình của bình phương độ dốc.

1. **Cập Nhật Trọng Số:**

Trọng số được cập nhật bằng cách sử dụng tỷ lệ học tập được điều chỉnh theo căn bậc hai của bình phương trung bình của bình phương độ dốc:

= -

η là tỷ lệ học tập, và ϵ là một số nhỏ để tránh chia cho 0.

1. **Điều Chỉnh Tự Nhiên Tỷ Lệ Học Tập:**

RMSProp tự động điều chỉnh tỷ lệ học tập dựa trên biến động của độ dốc. Nếu độ dốc lớn, tỷ lệ học tập sẽ giảm, giúp mô hình hội tụ ổn định hơn.

1. **Ưu Điểm và Lợi Ích:**

- Hiệu quả cho dữ liệu thưa (sparse).

- Giảm nguy cơ quá mức biến động của tỷ lệ học tập.

- Thích ứng tỷ lệ học tập cho từng tham số.

* **Adam (Adaptive Moment Estimation)**

Adam kết hợp hai ý tưởng chính từ các phương pháp tối ưu hóa trước đó là Momentum và RMSProp. Adam được thiết kế để cải thiện việc điều chỉnh tỷ lệ học tập cho từng tham số và giảm hiện tượng mất học (vanishing gradients) trong quá trình huấn luyện mô hình sâu.

1. **Gradient Descent với Momentum:**

Adam tích hợp ý tưởng của Momentum để giảm động độc trong quá trình cập nhật trọng số. Công thức cập nhật Momentum của Adam:

Ở đây, là độ dốc của hàm mất mát tại bước thời gian t, và là độ độc tại thời điểm t.

1. **RMSProp:**

Adam cũng sử dụng ý tưởng của RMSProp để thích ứng tỷ lệ học tập cho từng tham số dựa trên lịch sử của độ dốc. Công thức cập nhật RMSProp của Adam:

Ở đây, là bình phương trung bình của độ dốc.

1. **Bias-Correction:**

Trong Adam, có một bước được thêm vào để điều chỉnh độ độc và trung bình xuống 0 ở các bước đầu tiên của quá trình huấn luyện.

1. **Cập Nhật Trọng Số:**

Cuối cùng, trọng số được cập nhật bằng cách sử dụng các ước lượng được điều chỉnh của và :

= -

η là tỷ lệ học tập, và ϵ là một số nhỏ để tránh chia cho 0.

1. **Đặc Điểm và Lợi Ích:**

Adam kết hợp tính hiệu quả của Momentum và RMSProp.

Thích ứng tỷ lệ học tập cho từng tham số.

Hiệu quả trên nhiều loại mô hình và bài toán.

* **AdaDelta:**

AdaDelta là một phương pháp tối ưu hóa trong machine learning, một phần được thiết kế để giảm thiểu vấn đề về tỷ lệ học tập cần được thiết lập thủ công (hyperparameter tuning). Nó giống với RMSProp trong việc thích ứng tỷ lệ học tập cho từng tham số, nhưng AdaDelta sử dụng cách tiếp cận khác nhau để điều chỉnh tỷ lệ học tập.

1. **Hệ Số Chiết Khấu (Decay Factor):**

AdaDelta sử dụng một hệ số chiết khấu (γ) để giảm giá trị của bình phương trung bình của bình phương độ dốc (E[]) theo thời gian:

Ở đây, là độ dốc của hàm mất mát tại bước thời gian t, và là bình phương trung bình của bình phương độ dốc tại thời điểm t.

1. **Đặc Điểm Quan Trọng: Delta (Δ):**

Một điểm đặc biệt của AdaDelta là việc sử dụng Δ để giữ một bản sao của các thay đổi trọng số gần đây. Δ được cập nhật tương tự như E[]:

1. **Tỷ Lệ Học Tập (Learning Rate):**

AdaDelta không yêu cầu một tỷ lệ học tập cố định. Thay vào đó, nó sử dụng tỷ lệ học tập được tính toán dựa trên bình phương trung bình của bình phương thay đổi trọng số (Δθ) và bình phương trung bình của bình phương độ dốc (E[]):

Tỉ lệ học tập = ​

Ở đây, ϵ là một số nhỏ để tránh chia cho 0.

1. **Cập Nhật Trọng Số:**

Cuối cùng, trọng số được cập nhật bằng cách sử dụng tỷ lệ học tập tính được:

= - LR.

1. **Ưu Điểm và Lợi Ích**

Không yêu cầu điều chỉnh tỷ lệ học tập thủ công.

Hiệu quả cho dữ liệu thưa và quá trình học mô hình ổn định hơn.

Giảm vấn đề về lựa chọn tỷ lệ học tập, không cần phải thiết lập hyperparameter tỷ lệ học tập.

* **Adagrad:**

Adagrad (Adaptive Gradient Algorithm) là một phương pháp tối ưu hóa trong machine learning. Nó cũng thuộc dạng các phương pháp thích ứng tỷ lệ học tập, nhưng cách tiếp cận của nó khác so với RMSProp và AdaDelta.

Cách Adagrad Hoạt Động:

1. **Bình Phương Trung Bình Của Bình Phương Độ Dốc:**

Adagrad sử dụng một bình phương trung bình của bình phương độ dốc để thích ứng tỷ lệ học tập cho từng tham số:

Ở đây, là bình phương trung bình của bình phương độ dốc tại thời điểm t, và là độ dốc của hàm mất mát tại bước thời gian t.

1. **Tỷ Lệ Học Tập (Learning Rate):**

Tỷ lệ học tập được tính toán bằng cách chia tỷ lệ học tập cố định cho căn bậc hai của bình phương trung bình của bình phương độ dốc:

Tỉ lệ học tập =

Ở đây, η là tỷ lệ học tập cố định, và ϵ là một số nhỏ để tránh chia cho 0.

1. **Cập Nhật Trọng Số:**

Cuối cùng, trọng số được cập nhật bằng cách nhân độ dốc với tỷ lệ học tập tính được:

= - LR.

1. **Ưu Điểm và Lợi Ích:**

Không yêu cầu thủ công việc điều chỉnh tỷ lệ học tập.

Hiệu quả cho dữ liệu thưa và quá trình học mô hình ổn định hơn.

Tự động thích ứng với từng tham số dựa trên lịch sử của độ dốc.

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.**

* Continuous Learning trong Học Máy:

Continuous Learning trong ngữ cảnh Học Máy là quá trình cập nhật và cải tiến mô hình theo thời gian khi có dữ liệu mới. Điều này trở nên quan trọng trong môi trường động, nơi các mẫu dữ liệu cơ bản có thể thay đổi và mô hình cần thích ứng. Continuous Learning bao gồm các yếu tố quan trọng sau:

1. **Học Trực Tuyến (Online Learning)**: Trong Học Trực Tuyến, mô hình được cập nhật liên tục khi có dữ liệu mới xuất hiện. Điều này giúp mô hình thích ứng nhanh chóng và tận dụng tốt nguồn tài nguyên.
2. **Giám Sát Mô Hình**: Theo dõi hiệu suất của mô hình triển khai theo thời gian. Điều này bao gồm theo dõi các số liệu chất lượng, nhận biết sự suy giảm độ chính xác và đánh giá tác động của dự đoán mô hình.
3. **Đào Tạo Lại (Re-training)**: Định kỳ đào tạo lại mô hình với dữ liệu mới để duy trì độ chính xác. Tần suất này phụ thuộc vào bản chất của vấn đề và tốc độ phân phối dữ liệu.
4. **Vòng Lặp (Feedback Loop):** Thiết lập các vòng phản hồi để cải thiện mô hình dựa trên phản hồi từ người dùng, chuyên gia, và dữ liệu hiệu suất.
5. **Thuật Toán Thích Ứng:** Sử dụng các thuật toán có khả năng thích ứng với sự biến đổi trong phân phối dữ liệu.

* Ví dụ từ việc tạo một mô hình dự đoán giá nhà được đào tạo ban đầu với dữ liệu đã lưu trữ, và sau đó được cập nhật định kỳ khi có thông tin giá nhà mới. Mô hình này liên tục cập nhật kiến thức của mình để phản ánh thị trường bất động sản thay đổi.
* Test Production trong khi xây dựng 1 giải pháp học máy:

Test Production đề cập đến việc triển khai chiến lược thử nghiệm toàn diện cho mô hình Học Máy và đảm bảo rằng nó hoạt động đáng tin cậy trong môi trường thực tế. Các yếu tố chính của Test Production bao gồm:

1. **Kiểm Thử Đơn Vị (Unit Testing):** Kiểm tra từng thành phần của hệ thống Học Máy riêng lẻ để đảm bảo tính chính xác của chúng.
2. **Kiểm Thử Tích Hợp (Integration Testing)**: Đánh giá cách các thành phần khác nhau tương tác với nhau, từ đầu vào dữ liệu đến dự đoán mô hình
3. **Số Liệu Đánh Giá Mô Hình**: Xác định và theo dõi các số liệu đánh giá phù hợp để đảm bảo hiệu suất của mô hình, bao gồm các độ đo phổ biến như độ chính xác, độ chính xác, và các số liệu khác.
4. **Kiểm Thử Ổn Định(Stability Testing ):** Đánh giá độ ổn định của mô hình dưới các điều kiện khác nhau, như biến đổi trong phân phối dữ liệu.
5. **Kiểm Thử Hiệu Suất(Performance Testing):** Đánh giá tốc độ suy luận và sử dụng tài nguyên để đảm bảo mô hình đáp ứng yêu cầu sản xuất.
6. **Thử Nghiệm A/B**: Thực hiện thử nghiệm A/B để so sánh hiệu suất giữa các phiên bản mô hình khác nhau trong môi trường sản xuất.
7. **Giám Sát và Cảnh Báo(Monitoring and Alerting)**: Thực hiện giám sát liên tục để phát hiện các điểm bất thường và thiết lập cảnh báo khi có vấn đề phát sinh.

* Ví dụ thực tế em tìm hiểu về một mô hình dự đoán giá nhà dựa trên các đặc trưng như diện tích, số phòng, vị trí, và các yếu tố khác. Sau khi đào tạo mô hình và trước khi triển khai nó vào môi trường sản xuất, sau đó thực hiện các bước kiểm thử để đảm bảo rằng mô hình hoạt động hiệu quả và đáp ứng đúng đắn trong môi trường thực tế.

1. **Unit Testing (Kiểm thử đơn vị):**

Kiểm thử đầu vào: Đảm bảo rằng mô hình xử lý đúng các đặc trưng đầu vào và xử lý các tình huống đặc biệt như giá trị thiếu hoặc ngoại lệ.

Kiểm thử đầu ra: Xác minh rằng mô hình trả về dự đoán giá nhà có ý nghĩa và nằm trong khoảng dự kiến.

1. **Integration Testing (Kiểm thử tích hợp):**

Đảm bảo rằng mô hình tích hợp chính xác với hệ thống toàn cầu, bao gồm việc gửi và nhận dữ liệu từ ứng dụng hoặc hệ thống khác.

1. **Performance Testing (Kiểm thử hiệu suất):**

Đánh giá tốc độ suy luận: Xác minh rằng mô hình có thể đưa ra dự đoán một cách nhanh chóng và hiệu quả trong môi trường sản xuất.

Đánh giá tài nguyên: Kiểm tra mức sử dụng bộ nhớ và CPU của mô hình để đảm bảo nó không gây ra áp lực quá lớn cho hệ thống.

1. **Stability Testing (Kiểm thử ổn định):**

Chạy mô hình trong một khoảng thời gian dài để đảm bảo rằng nó không bị giảm hiệu suất theo thời gian.

1. **A/B Testing:**

So sánh hiệu suất của mô hình mới (đã được cập nhật) với mô hình cũ để đảm bảo rằng cải tiến nâng cao chất lượng dự đoán.

1. **Monitoring and Alerting (Giám sát và Cảnh báo):**

Triển khai hệ thống giám sát liên tục để theo dõi hiệu suất của mô hình trong môi trường sản xuất và cảnh báo khi có vấn đề.