**1. Bản chất của Gradient Descent**

Gradient Descent được dùng để:

* **Tối ưu hóa**: Tìm cực tiểu của hàm mất mát (loss function), giúp mô hình học cách dự đoán tốt hơn.
* **Học máy**: Điều chỉnh các tham số (weights, biases) để mô hình khớp tốt nhất với dữ liệu.

Nguyên tắc cơ bản:

* Gradient Descent dựa trên **đạo hàm** để đo độ dốc của hàm mất mát tại một điểm.
* Nếu gradient tại điểm đó là dương, bạn cần giảm giá trị tham số. Nếu gradient âm, bạn cần tăng giá trị tham số.
* Lặp lại điều này nhiều lần để tiến gần đến điểm cực tiểu.

2. Cơ chế hoạt động

B1: Khởi tạo: Đặt giá trị ban đầu cho các tham số (ví dụ, trọng số của mô hình).

B2: Tính gradient: Tính Gradient của hàm mất mát theo tham số :

B3: Cập nhập tham số:

* : Learning rate, một hệ số nhỏ giúp kiểm soát tốc độ cập nhập

B4: Lặp lại: Tiếp tục cập nhật tham số đến khi hội tụ, tức là gradient gần bằng 0 hoặt đạt số vòng lặp tối đa.

**3. Gradient Descent giải quyết vấn đề gì?**

Gradient Descent giải quyết bài toán **tối ưu hóa phi tuyến** trong học máy:

* **Tối ưu hàm mất mát**: Giảm thiểu sai số giữa dự đoán của mô hình và dữ liệu thực tế.
* **Huấn luyện mô hình**: Tìm ra các tham số tối ưu để mô hình dự đoán chính xác nhất.
* **Xử lý bài toán kích thước lớn**: Với các mô hình có nhiều tham số (như deep learning), GD hiệu quả trong việc xử lý không gian tham số phức tạp.

**4. Phân tích cách xây dựng công thức**

Công thức cập nhật của Gradient Descent:

Các thành phần:

1. **Gradient :**
   * Là vector gradient của hàm mất mát.
   * Chỉ hướng tăng nhanh nhất của hàm mất mát.
2. **Learning rate ():**
   * Điều chỉnh bước nhảy trong mỗi lần cập nhật.
   * quá lớn: Có thể "nhảy qua" cực tiểu.
   * quá nhỏ: Tiến đến cực tiểu rất chậm.
3. **Cập nhật tham số:**
   * Thực hiện phép trừ gradient nhân với learning rate.

**Biến thể của Gradient Descent:**

* **Batch Gradient Descent**: Sử dụng toàn bộ dữ liệu để tính gradient.
* **Stochastic Gradient Descent (SGD)**: Sử dụng một mẫu ngẫu nhiên từ dữ liệu để tính gradient.
* **Mini-batch Gradient Descent**: Sử dụng một tập con nhỏ của dữ liệu.

**5. Đánh giá và tối ưu Gradient Descent**

**Đánh giá:**

1. **Tốc độ hội tụ**:
   * Phụ thuộc vào learning rate ().
   * Quá nhanh có thể dẫn đến dao động quanh cực tiểu.
2. **Hội tụ tại cực tiểu cục bộ**:
   * Gradient Descent có thể mắc kẹt ở cực tiểu cục bộ, đặc biệt với hàm mất mát phi lồi.
3. **Chọn learning rate**:
   * Learning rate cần được điều chỉnh phù hợp.
   * Sử dụng **Learning Rate Scheduler** (giảm theo thời gian) để tăng hiệu quả.

**Tối ưu hóa:**

1. **Gradient Descent với Momentum**:
   * Thêm "quán tính" vào quá trình cập nhật:
2. **Adaptive Gradient Methods (Adam, RMSprop)**:
   * Thay đổi learning rate tự động cho từng tham số:

**6. Giải thích dễ hiểu**

Hãy tưởng tượng Gradient Descent giống như việc bạn đang ở trên đỉnh một ngọn núi và muốn đi xuống đáy thung lũng:

* Bạn không biết đường chính xác, nhưng bạn có thể cảm nhận độ dốc dưới chân.
* Gradient chỉ hướng dốc nhất để bạn đi xuống.
* Bạn bước từng bước nhỏ (learning rate) để đảm bảo không bị trượt hoặc bỏ lỡ đáy.

Với mỗi bước, bạn đang tiến gần hơn đến đáy (cực tiểu của hàm mất mát). Các biến thể như SGD hoặc Adam là những cách tối ưu để bạn "đi xuống" hiệu quả hơn, tránh việc bị lạc ở những ngọn đồi nhỏ (cực tiểu cục bộ).

**Các câu hỏi liên quan đến Gradient Descent**

1. **Gradient Descent là gì**

Gradient Descent là thuật toán tối ưu hóa nhằm tìm cực tiểu của hàm mất mát. Khi cực tiểu được tìm thấy, mô hình học máy sẽ dự đoán tốt hơn vì các tham số của nó được tối ưu hóa để giảm lỗi dự đoán.

Gradient Descent hoạt động bằng cách tính gradient (đạo hàm) của hàm mất mát theo các tham số của mô hình và cập nhập các tham số đó theo hướng giảm dần giá trị của hàm mất mát.

1. **Sự khác biệt giữa Gradient Descent, SGD và Mini-Batch Gradient Descent?**

Batch Gradient Descent:

* Sử dụng toàn bộ dữ liệu để tính gradient trước mỗi lần cập nhập tham số.
* Ưu điểm: Ổn định, hội tụ tốt.
* Nhược điểm: Tốn thời gian và tài nguyên trên tập dữ liệu lớn

Stochastic Gradient Descent (SGD):

* Cập nhật tham số dựa trên gradient của một mẫu dữ liệu ngẫu nhiên tại mỗi bước.
* Ưu điểm: Nhanh hơn, dễ nhảy ra khỏi local minima.
* Nhược điểm: Không ổn định, dao động quanh điểm cực tiểu.

Mini-Batch Gradient Descent:

* Kết hợp hai phương pháp trên bằng cách sử dụng một tập con dữ liệu nhỏ (mini-batch) tại mỗi bước.
* Ưu điểm: Cân bằng giữa tốc độ và độ ổn định.

1. **Cách chọn Learning Rate?**

Sử dụng phương pháp điều chỉnh learing rate như learning rate scheduler hoặc adam optimizer để tự động điều chỉnh trong quá trình huấn luyện.

1. **Gradient Descent có đảm bảo hội tụ đến điểm tối ưu toàn cục không?**

Gradient Descent không đảm bảo hội tụ đến điểm tối ưu toàn cục.

* Trong các hàm mất mát lồi (convex), Gradient Descent sẽ hội tụ đến điểm cực tiểu toàn cục.
* Nhưng trong các hàm không lồi (non-convex), Gradient Descent có thể dừng ở local minima hoặc saddle point.

Giải pháp để tránh mắc kẹt ở local minima:

* Momentum: Thêm động lượng vào quá trình cập nhập để tránh bị kẹt.
* Adam Optimizer: Tự động điều chỉnh tốc độ học tập (learning rate).
* Khởi tạo tham số tốt: Sử dụng các kỹ thuật như He Initialization hoặc Xavier Initialization.
* Tăng cường regularization: Giúp kiểm soát overfitting và cải thiện khả năng hội tụ.

**5. Điều gì xảy ra nếu bạn không chuẩn hóa dữ liệu trước khi áp dụng Gradient Descent?**

* Nếu không chuẩn hóa dữ liệu (standardization hoặc normalization):
  1. Hàm mất mát hội tụ chậm: Khi các đặc trưng có đơn vị khác nhau hoặc giá trị khác biệt lớn, Gradient Descent cần nhiều bước hơn để hội tụ.
  2. Đường đi zig-zag: Gradient sẽ dao động qua lại dọc theo trục của đặc trưng có giá trị lớn, dẫn đến hội tụ kém hiệu quả.
  3. Kết quả không chính xác: Một số thuật toán có thể không hội tụ hoặc cho kết quả sai lệch.

**6. Làm thế nào để chọn learning rate tốt cho Gradient Descent?**

* Một số cách chọn learning rate phù hợp:
  1. Dùng giá trị mặc định nhỏ (ví dụ: 0.01 hoặc 0.001): Đây là lựa chọn an toàn.
  2. Learning rate decay: Giảm dần learning rate theo thời gian để hội tụ mượt mà hơn.
  3. Grid search: Thử nhiều giá trị khác nhau để tìm giá trị tối ưu.
  4. Dùng kỹ thuật adaptive learning rate: Sử dụng các thuật toán như Adam, RMSProp, hoặc Adagrad.

**7. Điều gì xảy ra nếu learning rate quá lớn hoặc quá nhỏ?**

* Learning rate quá lớn:
  1. Gradient Descent có thể nhảy qua cực tiểu và không hội tụ.
  2. Hàm mất mát dao động mạnh và không giảm.
* Learning rate quá nhỏ:
  1. Gradient Descent hội tụ rất chậm.
  2. Tốn nhiều tài nguyên tính toán.

**8. Các chiến lược để vượt qua local minima khi áp dụng Gradient Descent là gì?**

* Một số chiến lược vượt qua local minima:
  1. Thêm động lượng (Momentum): Giúp gradient tích lũy và vượt qua các local minima nông.
  2. Khởi tạo ngẫu nhiên nhiều lần: Thử nghiệm với nhiều điểm khởi tạo để tăng cơ hội tìm global minima.
  3. Sử dụng thuật toán nâng cao: Các thuật toán như Adam, RMSProp thường hiệu quả hơn với các hàm mất mát không lồi.
  4. Giới thiệu noise: Thêm noise nhỏ vào gradient để thoát khỏi local minima.

**9. Batch size ảnh hưởng đến Gradient Descent như thế nào?**

* Batch size lớn:
  1. Gradient tính toán chính xác hơn.
  2. Tốn nhiều bộ nhớ và thời gian hơn.
* Batch size nhỏ (SGD):
  1. Gradient không ổn định nhưng hội tụ nhanh hơn.
  2. Dễ vượt qua local minima nhưng có thể dao động xung quanh điểm tối ưu.
* Mini-batch size: (thường 32, 64, hoặc 128)
  1. Là sự cân bằng giữa tốc độ và độ chính xác, phổ biến nhất trong thực tế.

**10. Vai trò của regularization trong Gradient Descent là gì?**

* Regularization giúp tránh overfitting và cải thiện hội tụ bằng cách điều chỉnh hàm mất mát:
  1. L1 Regularization (Lasso): Thêm giá trị tuyệt đối của trọng số vào hàm mất mát → Khuyến khích trọng số nhỏ hơn hoặc bằng 0.
  2. L2 Regularization (Ridge): Thêm bình phương của trọng số vào hàm mất mát → Làm trọng số nhỏ hơn nhưng không về 0 hoàn toàn.
  3. Dropout: Ngẫu nhiên bỏ một số neuron trong mạng nơ-ron để tránh overfitting.