Gradient Descent

1. Khái niệm

Gradient Descent là gì?

Giả sử bạn có một **hàm chi phí J(w, b)** và bạn muốn **tối thiểu hóa** nó.

- Với hồi quy tuyến tính, đây là hàm chi phí dựa trên lỗi bình phương.
- Nhưng **gradient descent** không chỉ dùng cho hồi quy tuyến tính — nó có thể dùng để tối thiểu **bất kỳ hàm nào**, kể cả với nhiều tham số như:

$$J(w_1, w_2, ..., w_n, b)$$

  Mục tiêu là điều chỉnh các giá trị $w_1, ..., w_n$,b sao cho J nhỏ nhất.

Cách gradient descent hoạt động:

- **Chọn giá trị khởi tạo** cho w và b. Với hồi quy tuyến tính, bạn có thể đơn giản bắt đầu từ:

$$w = 0, \quad b = 0$$

- **Thực hiện gradient descent**:
  - Ở mỗi bước, bạn sẽ **cập nhật một chút** giá trị của w và b
  - Mỗi lần như vậy sẽ **làm giảm giá trị của J(w, b)** cho đến khi J gần đạt giá trị nhỏ nhất.

Đặc điểm của gradient descent

- **Điểm bắt đầu ảnh hưởng lớn** đến nơi bạn sẽ kết thúc.
- Nếu bạn **bắt đầu từ một vị trí khác** (chỉ cách vài bước), gradient descent có thể **dẫn bạn đến một cực tiểu khác hoàn toàn**.
- Cả hai điểm đáy này đều là **cực tiểu cục bộ (local minima)**.
- Khi đã đi vào một cực tiểu, gradient descent **không thể tự thoát ra để tìm thung lũng khác**, vì nó chỉ "nhìn quanh gần mình" để đi xuống — không "nhìn xa".

2. Cài đặt

Đây là công thức gradient descent:

$$w := w - \alpha \cdot \frac{d}{dw} J(w, b)$$

Ý nghĩa của biểu thức trên là:

- Cập nhật tham số w bằng cách lấy giá trị hiện tại của nó,
- Trừ đi một lượng nhỏ: $\alpha \cdot \frac{d}{dw} J(w, b)$,
- Trong đó:
  - α là **learning rate** (tốc độ học),
  - $\frac{d}{dw} J(w, b)$ là **đạo hàm của hàm chi phí** theo w.
- α (alpha): là **learning rate**, thường là số nhỏ như **0.01**, quyết định kích thước bước nhảy mỗi lần đi xuống "dốc".
  - Alpha lớn → bước nhảy lớn → có thể đi nhanh nhưng dễ "nhảy qua" điểm tối ưu.
  - Alpha nhỏ → bước nhảy nhỏ → ổn định hơn nhưng chậm hơn.
- $\frac{d}{dw} J(w, b)$ là **đạo hàm của hàm chi phí theo w**. Đạo hàm này cho bạn biết **nên bước về hướng nào để giảm J nhanh nhất**.
- Tương tự, ta cũng có cập nhật cho bbb:

$$b := b - \alpha \cdot \frac{d}{db} J(w, b)$$

Cập nhật đồng thời (Simultaneous Update)

Vì mô hình có 2 tham số w và b, bạn cần cập nhật **cả hai đồng thời**. Điều này có nghĩa là:

1. **Tính giá trị mới của www** và **bbb** trước,
2. Lưu chúng vào biến tạm `temp_w` và `temp_b`,
3. Sau đó mới gán lại giá trị cho w và b.

```
temp_w = w - alpha * dJ_dw
temp_b = b - alpha * dJ_db
```

```
w = temp_w
b = temp_b
```

3. Learning rate

Việc chọn **learning rate (α)** đúng là **rất quan trọng** đối với hiệu quả của thuật toán Gradient Descent. Nếu chọn sai, thuật toán **có thể cực kỳ chậm** hoặc thậm chí **không hội tụ**.

Trường hợp 1: **Learning rate quá nhỏ**

Giả sử ta khởi tạo gradient descent tại một điểm nào đó trên đồ thị của J(w).

- Nếu α rất nhỏ, ví dụ $\alpha = 0.0000001$
- Mỗi bước cập nhật sẽ rất bé – chỉ là **những bước đi tí hon**
- Cập nhật sau mỗi bước:

$$w := w - \alpha \cdot \frac{d}{dw} J(w)$$

- Mặc dù hướng đi đúng, nhưng tiến rất chậm.
- Kết quả là cần **rất nhiều bước** mới tới được điểm cực tiểu.**Tóm lại**: Gradient descent **sẽ hội tụ**, nhưng **rất chậm** → không hiệu quả.

Trường hợp 2: **Learning rate quá lớn**

Giả sử bạn khởi tạo từ một điểm gần cực tiểu.

- Nếu α quá lớn, bước cập nhật sẽ là **bước nhảy khổng lồ**
- Điều này có thể khiến thuật toán **nhảy qua bên kia của cực tiểu**, và **tăng** giá trị hàm chi phí thay vì giảm.
- Sau mỗi bước, nếu vẫn giữ α lớn, bạn tiếp tục **nhảy qua nhảy lại**, và **càng lúc càng xa cực tiểu**.

Tóm lại: Với α quá lớn → **thuật toán dao động** hoặc thậm chí **diverge (phân kỳ)**.

Trường hợp đặc biệt: **W ở ngay cực tiểu**

Giả sử sau một số bước, w đã ở ngay **cực tiểu cục bộ** của hàm chi phí.

- Tại cực tiểu: đạo hàm $\frac{d}{dw}J(w) = 0$
- Do đó:

$$w := w - \alpha \cdot 0 = w$$

- Tức là: **w không đổi**, Gradient Descent giữ nguyên tham số.
- Đây là hành vi **mong muốn**, vì bạn đã tìm được nghiệm.

4. Gradient Descent trong Linear Regression

Tính chất hàm cost trong Linear Regression

Một điều đặc biệt: **hàm mất mát bình phương sai số là một hàm lồi (convex)**.

- Nó chỉ có **một điểm cực tiểu duy nhất** – không có cực tiểu cục bộ
- Vì vậy, Gradient Descent **luôn tìm ra nghiệm toàn cục** (global minimum), miễn là bạn chọn learning rate hợp lý.

5. Batch Gradient Descent

Batch Gradient Descent là một biến thể của gradient descent, trong đó:

- Mỗi bước cập nhật đều sử dụng **toàn bộ tập dữ liệu huấn luyện**
- Khi tính đạo hàm, ta dùng tổng từ i = 1 đến m, với m là số mẫu huấn luyện

Có những biến thể của Gradient Descent **không sử dụng toàn bộ dữ liệu** cho mỗi bước, mà dùng **một phần nhỏ** (ví dụ: Mini-Batch hoặc Stochastic Gradient Descent)