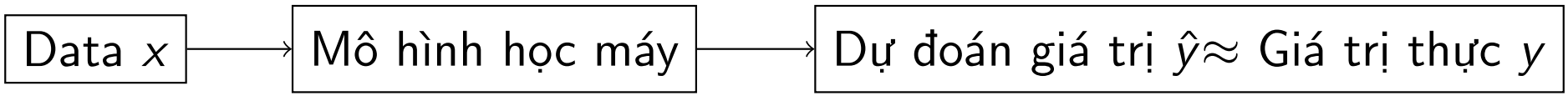
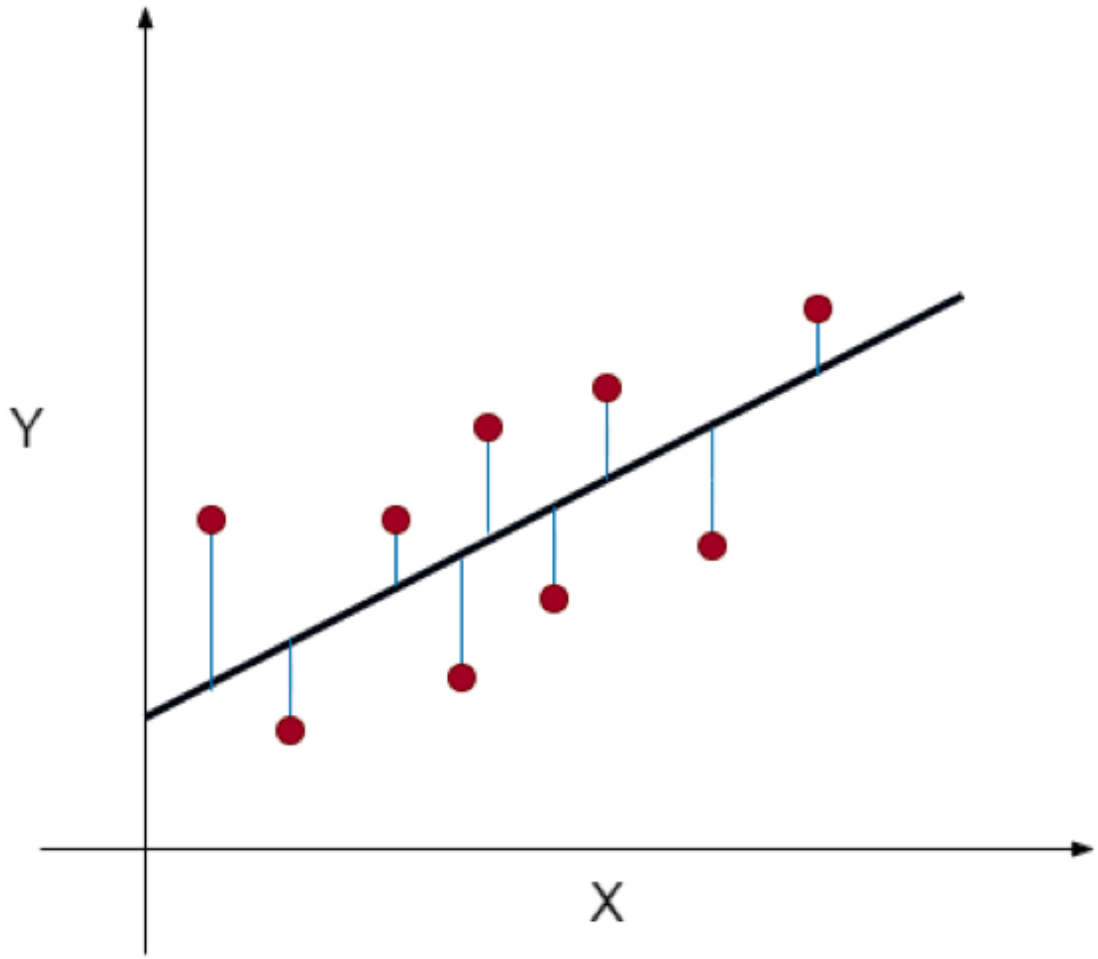
# Linear Regression

# Mục lục

# Mô hình Linear Regression



# Ví dụ




- X: diện tích nhà
- Y: giá nhà thực tế

# Mô hình hóa Linear Regression

- Chọn $f_\theta(x_i) = \theta_1 x_i + \theta_0 \Rightarrow \theta$ sẽ bao gồm $\theta_0$ và $\theta_1$
- Cần tìm:

$$\text{argmin}_\theta \sum_{i=1}^{n} (f_\theta(x_i) - y_i)^2$$

hay

$$\text{argmin}_{\theta_0,\theta_1} \quad \frac{1}{2n} \sum_{i=1}^{n} (\theta_1 x_i + \theta_0 - y_i)^2$$

- Tại sao không dùng hàm trị tuyệt đối mà dùng hàm bình phương?
- Tại sao lại chia $\frac{1}{2}$ và $\frac{1}{n}$?

# Tìm tham số

**Bước 1: Tính đạo hàm riêng:** $\frac{\partial \mathcal{L}}{\partial \theta}$: $\mathcal{L}(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} (\theta_1 x_i + \theta_0 - y_i)^2$

- $\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} (\theta_1 x_i + \theta_0 - y_i)$
- $\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} (\theta_1 x_i + \theta_0 - y_i) x_i$

**Bước 2: Khởi tạo:** $\theta_0, \theta_1$ ngẫu nhiên, $\alpha, \epsilon > 0$ đủ nhỏ.

**Bước 3: Lặp:**

- $\theta_0 \leftarrow \theta_0 - \alpha \frac{\partial \mathcal{L}}{\partial \theta_0}$
- $\theta_1 \leftarrow \theta_1 - \alpha \frac{\partial \mathcal{L}}{\partial \theta_1}$
- Nếu $\left| \frac{\partial \mathcal{L}}{\partial \theta_0} \right| < \epsilon$ và $\left| \frac{\partial \mathcal{L}}{\partial \theta_1} \right| < \epsilon$: dừng lặp.

**Kết luận:** $\theta$ là tham số để $\mathcal{L}$ đạt cực tiểu.

# Vecto hóa công thức

$$\text{Đặt } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}$$

$$f_w(x) = \mathbf{w}^T \mathbf{X} = \begin{bmatrix} w_0 & w_1 \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{bmatrix}$$

$$= \begin{bmatrix} w_0 \cdot 1 + w_1 \cdot x_1 & w_0 \cdot 1 + w_1 \cdot x_2 & \cdots & w_0 \cdot 1 + w_1 \cdot x_n \end{bmatrix}$$

$$= \begin{bmatrix} w_0 + w_1 x_1 & w_0 + w_1 x_2 & \cdots & w_0 + w_1 x_n \end{bmatrix}$$

# Vector hóa công thức

Tính đạo hàm riêng theo $w_0$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n}(w_1 x_i + w_0 - y_i) = \frac{1}{n}\mathbf{X}_1(\mathbf{w}^T\mathbf{X} - \mathbf{Y})^T$$

Giải thích:

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\mathbf{w}^T\mathbf{X} - \mathbf{Y} = \begin{bmatrix} w_0 + w_1 x_1 - y_1 & w_0 + w_1 x_2 - y_2 & \cdots & w_0 + w_1 x_n - y_n \end{bmatrix}$$

$$\mathbf{X}_1(\mathbf{w}^T\mathbf{X} - \mathbf{Y})^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} w_0 + w_1 x_1 - y_1 \\ w_0 + w_1 x_2 - y_2 \\ \vdots \\ w_0 + w_1 x_n - y_n \end{bmatrix} = \sum_{i=1}^{n}(w_0 + w_1 x_i - y_i)$$

# Vector hóa công thức

Tính đạo hàm riêng theo $w_1$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n}(w_1 x_i + w_0 - y_i)x_i = \frac{1}{n}\mathbf{X}_2(\mathbf{w}^T\mathbf{X} - \mathbf{Y})^T$$

Giải thích:

$$\mathbf{X}_2 = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

$$\mathbf{w}^T\mathbf{X} - \mathbf{Y} = \begin{bmatrix} w_0 + w_1 x_1 - y_1 & w_0 + w_1 x_2 - y_2 & \cdots & w_0 + w_1 x_n - y_n \end{bmatrix}$$

$$\mathbf{X}_2(\mathbf{w}^T\mathbf{X} - \mathbf{Y})^T = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \cdot \begin{bmatrix} w_0 + w_1 x_1 - y_1 \\ w_0 + w_1 x_2 - y_2 \\ \vdots \\ w_0 + w_1 x_n - y_n \end{bmatrix} = \sum_{i=1}^{n}(w_0 + w_1 x_i - y_i)x_i$$

# Mô hình Linear Regression đa biến

Dữ liệu đầu vào được ký hiệu: $\mathbf{x}^{(1)} \in \mathbb{R}^d$, dữ liệu đầu ra $y^{(1)} \in \mathbb{R}$ khi đó:

$$\mathcal{L}(w_0, \mathbf{w}_1) \cong \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}_1^T \mathbf{x}^{(i)} + w_0 - y^{(i)})^2$$

trong đó: $\mathbf{w}_1 = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,d} \end{bmatrix}$

# Mô hình Linear Regression đa biến

Tính đạo hàm riêng ta có:

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{w}_1^T \mathbf{x}^{(i)} + w_0 - y^{(i)} \right)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{1,k}} = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{w}_1^T \mathbf{x}^{(i)} + w_0 - y^{(i)} \right) \mathbf{x}_k^{(i)}$$

# Vecto hóa công thức

Đặt $w = \begin{bmatrix} w_0 \\ \mathbf{w}_1 \end{bmatrix}$, $\mathbf{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(n)} \end{bmatrix}$, $Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(n)} \end{bmatrix}$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{w}_1^T \mathbf{x}^{(i)} + w_0 - y^{(i)} \right) = \frac{1}{n} X_1 \left( \mathbf{w}^T \mathbf{X} - Y \right)^T$$

$$\frac{\partial \mathcal{L}}{\partial w_{1,k}} = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{w}_1^T \mathbf{x}^{(i)} + w_0 - y^{(i)} \right) x_k^{(i)} = \frac{1}{n} X_{k+1} \left( \mathbf{w}^T \mathbf{X} - Y \right)^T$$

# Vecto hóa công thức

Tổng quát hóa công thức ta có:

$$\nabla \mathcal{L} = \frac{1}{n} \times \left( \mathbf{w}^T \mathbf{X} - Y \right)^T$$

# Cài đặt Linear Regression đơn biến bằng Numpy

# Các biến thể của Gradient Descent

Các biến thể của Gradient Descent

- Batch Gradient Descent
- Stochastic Gradient Descent (SGD)
- Mini-Batch Gradient Descent

# Batch Gradient Descent

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta J(\theta^{(k)})$$

- Cập nhật tham số $\theta$ bằng cách sử dụng **toàn bộ** các điểm dữ liệu $x_i$.
- Cách làm này có một vài hạn chế đối với cơ sở dữ liệu có vô cùng nhiều điểm.
- Thuật toán này được coi là không hiệu quả với online learning vì mỗi lần thêm vài điểm dữ liệu mới là mô hình phải tính lại đạo hàm của hàm mất mát tại tất cả các điểm dữ liệu.

# Stochastic Gradient Descent (SGD)

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta J(\theta^{(k)}; \mathbf{x}^{(i)}, y^{(i)})$$

- Thay vì sử dụng toàn bộ tập dữ liệu để cập nhập tham số thì ta có thể sử dụng **từng dữ liệu một** để cập nhập.
- Về cơ bản ở mỗi lần cập nhập tham số, ta duyệt toàn bộ các cặp mẫu $\mathbf{x}^{(i)}$ và $y^{(i)}$
- Vì sử dụng từng mẫu đơn một nên tốc độ tính toán đạo hàm sẽ nhanh hơn rất nhiều nhưng tốc độ hội tụ bị giảm đi.
- Sau mỗi epoch, cần xáo trộn thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên.

# Minibatch Gradient Descent:

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta J(\theta^{(k)}; \mathbf{X}^{(i)}, \mathbf{y}^{(i)})$$

- Kết hợp giữa BGD và SGD.
- Lấy một số lượng k dữ liệu $1 < k < n$. Chia mỗi minibatch có k dữ liệu, có trường hợp minibatch cuối cùng sẽ ít hơn k nếu số lượng dữ liệu không chia hết cho k.
- Mỗi epoch chúng ta sẽ xáo trộn dữ liệu ngẫu nhiên rồi chia vào mỗi mini-batch.

**Mini-batch GD thường được sử dụng hơn vì:**

- Tốc độ hội tụ nhanh hơn Stochastic Gradient Descent.
- Tốc độ tính đoạ hàm nhanh hơn Batch Gradient Descent.

# Điều kiện dừng:

- Giới hạn số vòng lặp: Phương pháp phổ biến đảm bảo chương trình không chạy quá lâu nhưng có thể thuật toán dừng lại trước khi đủ gần với nghiệm.
- So sánh gradient của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Phương pháp này cũng có một nhược điểm lớn là việc tính đạo hàm đôi khi trở nên quá phức tạp (ví dụ như khi có quá nhiều dữ liệu), nếu áp dụng phương pháp này thì coi như ta không được lợi khi sử dụng SGD và mini-batch GD.
- So sánh giá trị của hàm mất mát của nghiệm tại hai lần cập nhật liên tiếp, khi nào giá trị này đủ nhỏ thì dừng lại. Nhược điểm của phương pháp này là nếu tại một thời điểm, đồ thị hàm số có dạng bằng phẳng tại một khu vực nhưng khu vực đó không chứa điểm local minimum, thuật toán cũng dừng lại trước khi đạt giá trị mong muốn.
- Trong SGD và mini-batch GD, cách thường dùng là so sánh nghiệm sau một vài lần cập nhật.

# Vectorization for Speed

**Thank you!**