x_j = j^{th} Feature $\quad$ $\vec{x}^{(i)}$ = Features of i^{th} training example

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b \quad : \quad f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$$

Vectorization in Python

np.dot(w,x) → f = np.dot(w,x) + b

Gradient Descent

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots, w_n, b) \qquad \frac{\partial}{\partial w_j} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots, w_n, b) \qquad \frac{\partial}{\partial b} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f(\vec{x}^{(i)}) - y^{(i)}\right)$$

Feature Scaling

Features with a range of big values tends to have their optimal weight small. In order to maintain that we can do scale down their range.

Mean Normalization

s_j = normalized x_j

μ_j = mean of x_j values

$$s_j = \frac{x_j - \mu_j}{\max(x_j) - \min(x_j)}$$

Z-Score Normalization

s_j = normalized x_j

μ_j = mean of x_j

σ_j = Standart deviation of x_j

$$s_1 = \frac{x_j - \mu_j}{\sigma_j}$$

* Aim for about $-1 \le x_j \le 1$ for each feature x_j