Multiple Regression

ARYAN

Multiple Regression

In multiple regression, $f_{w_1, w_2 \cdots w_n, b}(x_1, x_2, x_3, \cdots x_n, b)$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_n \end{bmatrix}, \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_n \end{bmatrix}, \quad b$$

$$f_{w,b}(\vec{x}^{(i)}) = w_1 x_1 + w_2 x_2 \cdots w_n x_n + b$$

$$= \vec{w} \cdot \vec{x} + b$$

- Eg Price of a house depends on multiple variables like square feet, distance to school ....

  These are our $\vec{x}$ for a particular house

★ Each i^{th} house has n variables $\Rightarrow \vec{x}$

- Each variable in MR has weights associated with it.

Notation:

n = number of features

m = number of samples

$\vec{x}_j$ = Value for the j^{th} feature (Square foot)

$\vec{x}^{(i)}$ = Value of features for i^{th} sample

$\vec{x}_j^{(i)}$ = Value of j^{th} feature for i^{th} sample

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

Computation Concerns

- **Vectorization:**

  Numpy:

```
w = np.array([1.2.5, -3.3])
b = 4
x = np.array([10, 20, 30])
f = w[0] x x[0] ... w[2] x x[2] + b
```

Improvement? → `f = np.dot(w, x) + b`

Faster?

└ It uses parallel processing to compute each $w \times x$

Gradient descent

$\vec{w}, \vec{d}$ ← Partial derivatives

$w_1 = w_1 - 0.1 d_1$

$w_n = w_n - 0.1 d_n$

$w = w - 0.1 \times d$ } Vectorised Approach

$= \begin{bmatrix} \vdots \end{bmatrix} - 0.1 \begin{bmatrix} \vdots \end{bmatrix}$

Gradient descent

1. Initialize w, b randomly
2. Repeat till convergence

$$w = w - \alpha \, \partial J / \partial w$$
$$b = b - \alpha \, \partial J / \partial \alpha$$

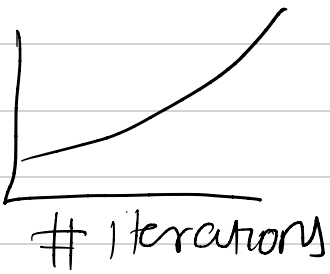
$$w = w - \alpha \left(\frac{1}{m} \sum_{i} (f(x_i) - y_i) x_i \right)$$
$$b = b - \alpha \left(\frac{1}{m} \sum (f(x_i) - y_i) \right)$$

- Feature size & parameter sizes must be chosen appropriately
- Thats why tend to scale our features
  eg normalization, mean normalization, z score normalisation

---

- We keep a threshold to check for convergence, and if the change in $J(w)$ is $<$ threshold we conclude that it has converged

if α is too large $\Rightarrow$ $J(w)$



Feature Engineering

$f_{w,b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$

x_1: frontage, x_2: depth

area $= x_1 \times x_2 = x_3$

$f_{w,b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$