# Linear Regression with multiple features

$$X = \begin{bmatrix} \ldots x^1 \ldots \\ x^2 \\ \vdots \\ x^n \end{bmatrix} = \begin{bmatrix} x_0^1 & x_1^1 & \ldots & x_n^1 \\ & x_j^i & \\ & \vdots & \end{bmatrix}$$

$x_j^i$ = $i^{th}$ example $j^{th}$ feature

$$y = \begin{bmatrix} \\ m_i \\ \\ \end{bmatrix}$$

Training $X, y$ } given
Test $x$

Hypothesis

$$y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Parameters

weighted sum of all features

let Area $\rightarrow x_1$

rooms $\rightarrow x_2$

$$= \theta_0 + \sum_{i=1}^{n} \theta_i x_i$$

$$= \theta_0 x_0 + \sum_{i=1}^{n} \theta_i x_i$$

dummy feature $x_0 = 1$

$$= \sum_{i=0}^{n} \theta_i x_i$$

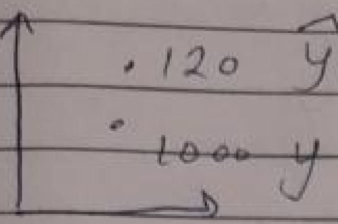
$$= \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$x$ = 1 house

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$ its features

# Loss Function




$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (y^i - \hat{y}^i)^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (y^i - \underbrace{h_\theta x^i}_{\theta^T x^i})^2$$

# Gradient Descent Update Rule

Get value of $\theta$ to minimize $J(\theta)$

$$\theta := \theta - \eta \nabla_\theta J(\theta) \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \text{ vector}$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} (\hat{y} - y)^2$$

for $X_j^i$ one example

$$= \frac{\partial}{\partial \theta_j} (h_\theta(x) - y)^2$$

$$= 2 \frac{\partial}{\partial \theta_j} (h_\theta(x))((h_\theta x) - y)$$

$$= (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} \left[\sum_{j=0}^{n} \theta_j x_j\right]$$

$(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_j x_j + \theta_n x_n$

$$= (h_\theta(x) - y) x_j^i$$

$$\boxed{\frac{\partial J(\theta)}{\partial \theta_j} = (\hat{y} - y) x_j^i} \text{ gradient wrt } \theta_j$$

for all example $\frac{\partial J \theta}{\partial \theta_j} = \sum_{i=1}^{m} (\hat{y}^i - y^i) x_j^i$

Finally gradient Update

$$\theta_j = \theta_j - \eta \sum_{i=1}^{m} (\hat{y}^i - y^i) x_j^i$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$ $\hat{y} = \theta^T x$ & given a prediction

axis = 0 rowwise    axis = 1 column wise

Implementation of Gradient Descent for Multiple Features

$X$ = Matrix ($m \times n$)    $x$ : Vector (single example with n features)

$$= \begin{bmatrix} 1 - x^1 \text{———} \\ 1 - x^2 \text{———} \\ 1 - x^3 \text{———} \\ \vdots - \vdots \text{———} \\ 1 - x^n \text{———} \end{bmatrix}$$

$$= \theta_0 \underset{\substack{\downarrow \\ 1}}{x_0^i} + \theta_1 x_1^i + \theta_{13} x_{13}^i$$

```
def hypothesis (x, theta):
    y_ = 0.0
    n = x.shape[0]
    for i in range(n):
        y_ += (theta[i] * x[i])
    return y_
```

calc:- $y_- = \sum \theta_i x_i$ predictions

```
def error(x, y, theta):
    e = 0.0
    m = x.shape[0]
    for i in range(m):
        y_ = hypothesis(x[i], theta)
        e += (y[i] - y_)**2
    return e/m
```

$$error = \frac{1}{m}\sum_{i=0}^{m}(\hat{y}-y)^2$$

```
def gradient(x, y, theta):
    m, n = x.shape
    grad = np.zeros((n,))
    for j in range(n):
        # for all values of j
        for i in range(m):
            # sum over all examples
            y_ = hypothesis(x[i], theta)
            grad[j] += (y_ - y[i]) *
                        x[i][j]
    return grad/m
```

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{m}(\hat{y}^i - y^i)x_j^i$$

1 epoch = 1 iteration over complete dataset

Page No.

Date :

```
def gradient_descent(X, y, lr = 0.1, max_epochs = 300)
    m,n = X.shape
    theta = np.zeros((n,))
    error_list = []
    for i in range(max_epochs):
        e = error(X, y, theta)
        error_list.append(e)
        # gradient descent
        grad = gradient(X, y, theta)
        for j in range(n):
            theta[j] = theta[j] - lr * grad[j]

    return theta, error_list
```