Artificial Intelligence and Machine Learning

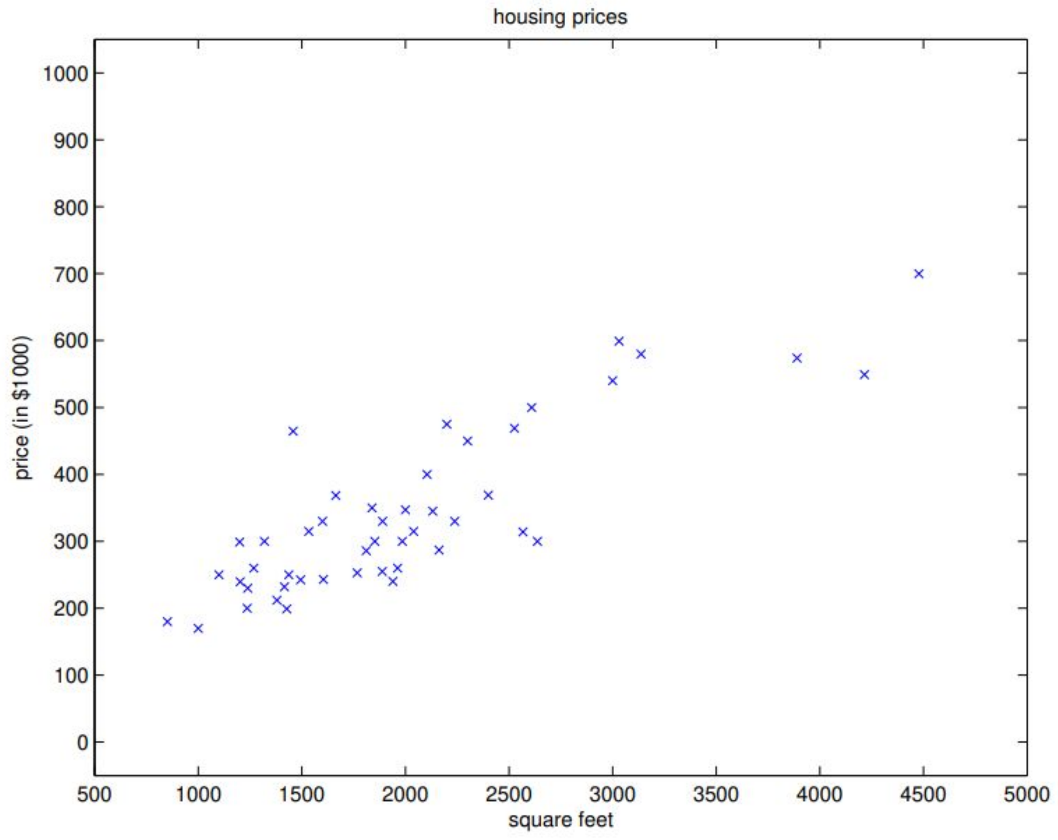
Linear Regression
Part 1

Motivation

- Linear Regression is “still” one of the most widely used ML/DL Algorithms
- Easy to understand and implement
- Efficient to Solve
- We will use Linear Regression to Understand the concepts of:
  - Data
  - Models
  - Loss
  - Optimization

House Price Prediction from 1 Feature

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

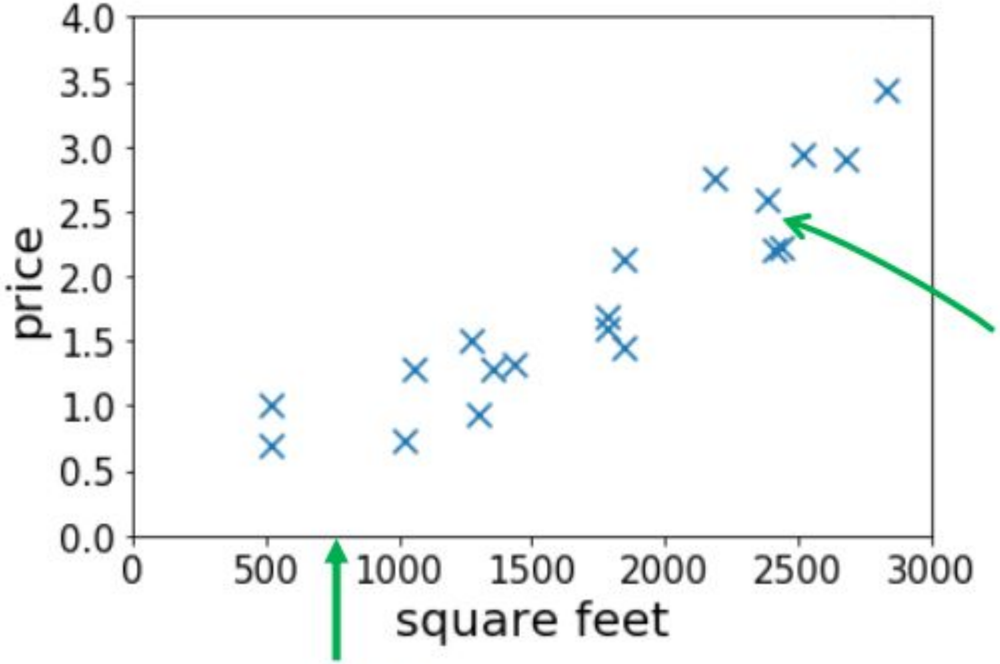


Housing Price Prediction

- Given: a dataset that contains n samples

$$\left(x^{(1)}, y^{(1)}\right), \dots \left(x^{(n)}, y^{(n)}\right)$$

- Task: if a residence has x square feet, predict its price?

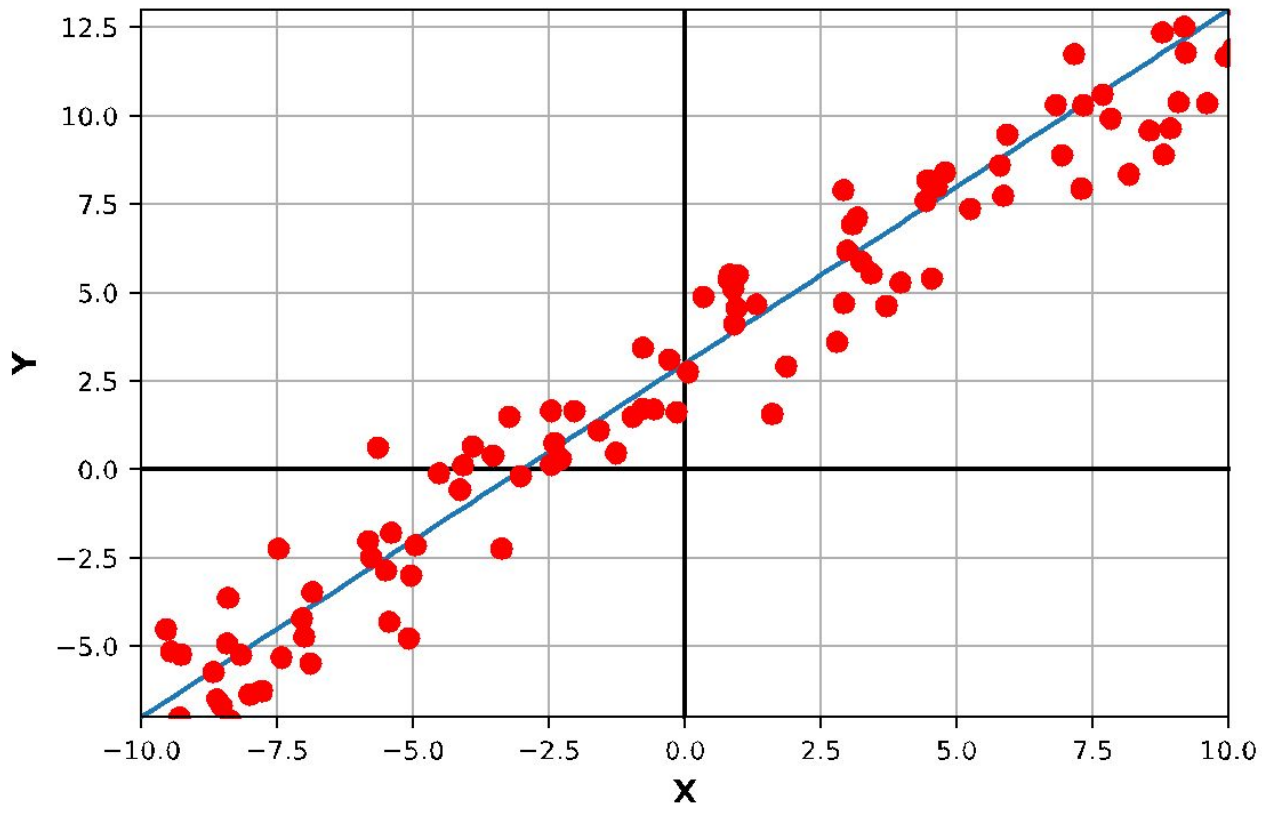


15th sample
$\left(x^{(15)}, y^{(15)}\right)$

$x = 800$
$y = ?$

from Machine Learning CS229/STATS229 by Moses Charikar and Chris Ré

Simple Linear Regression



Model (*Linear*)

Observation: Linear relation between x and y

$$Y = mX + b$$

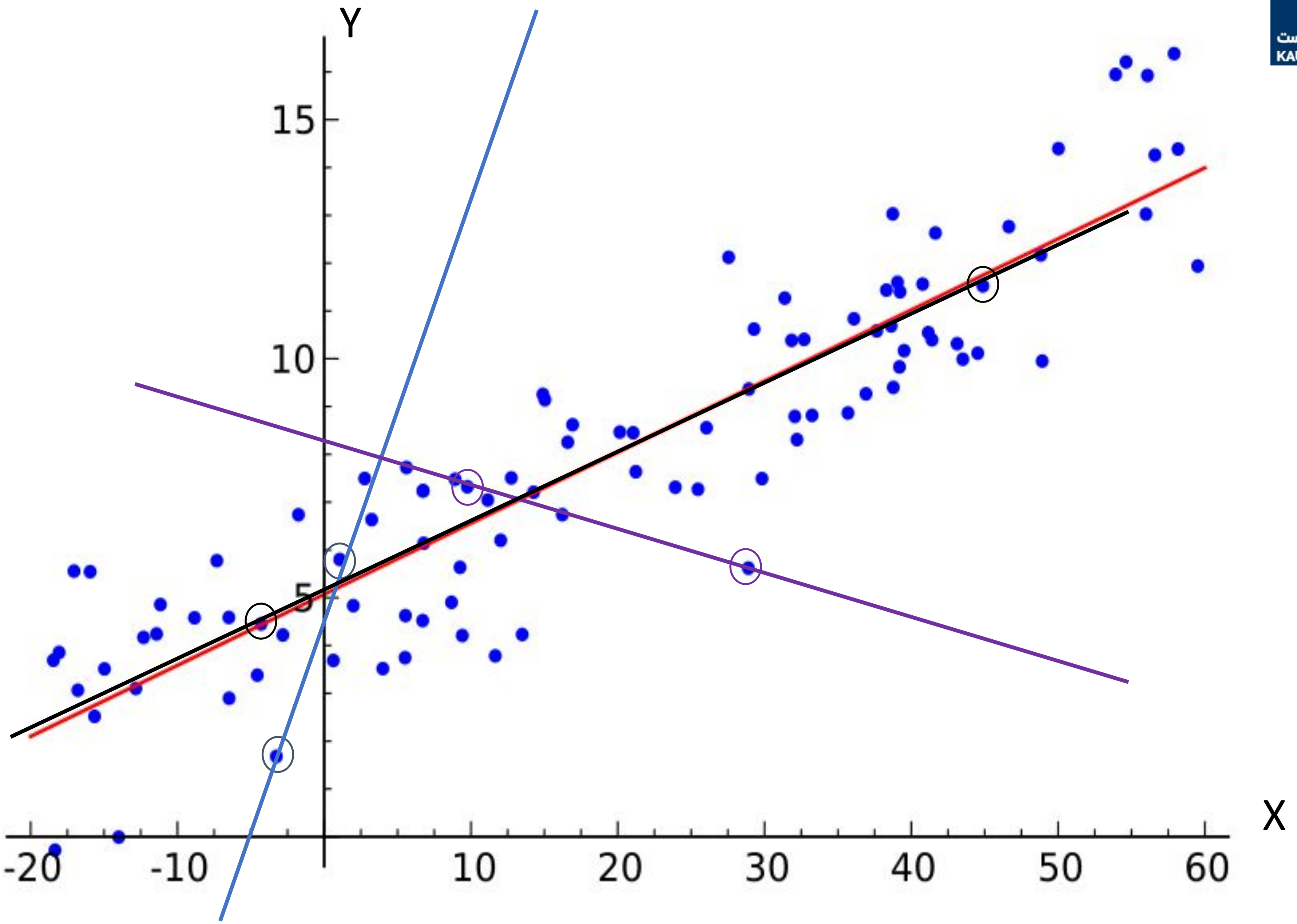
Objective: Find $\theta = \{m, b\}$

Y: Label, Response Variable
X: Features, Regressors
m: Slope
b: Bias

Simple Linear Regression

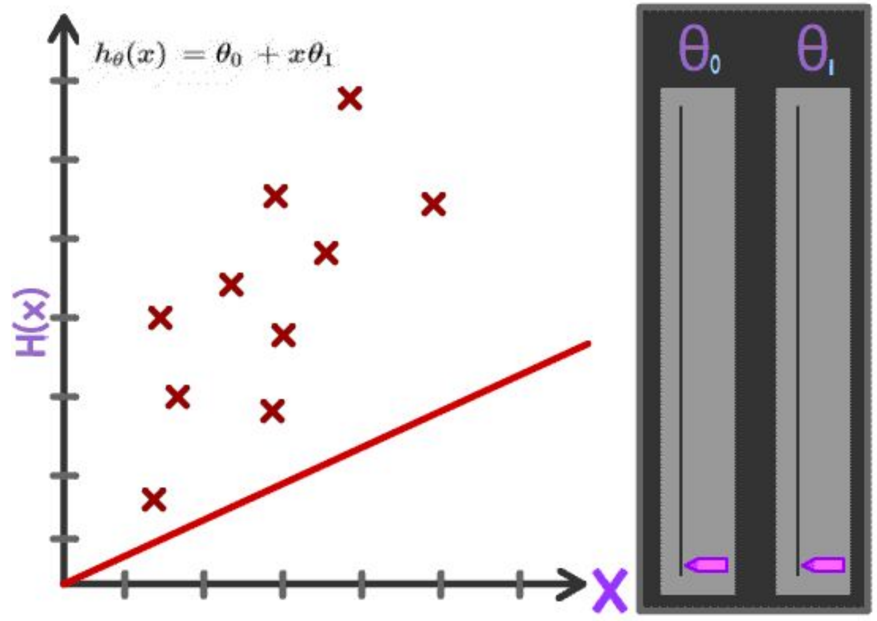
- Input: data (x_i, y_i), $i \in \{1,2,\dots,N\}$
- Goal: learn values of variable (m, b) $\quad Y = mX + b$

- Question: How many points in a plane do we need to fit a line through it?


Y
15
10
5
-20
-10
10
20
30
40
50
60
X

Solution Strategy for Solving the Problem

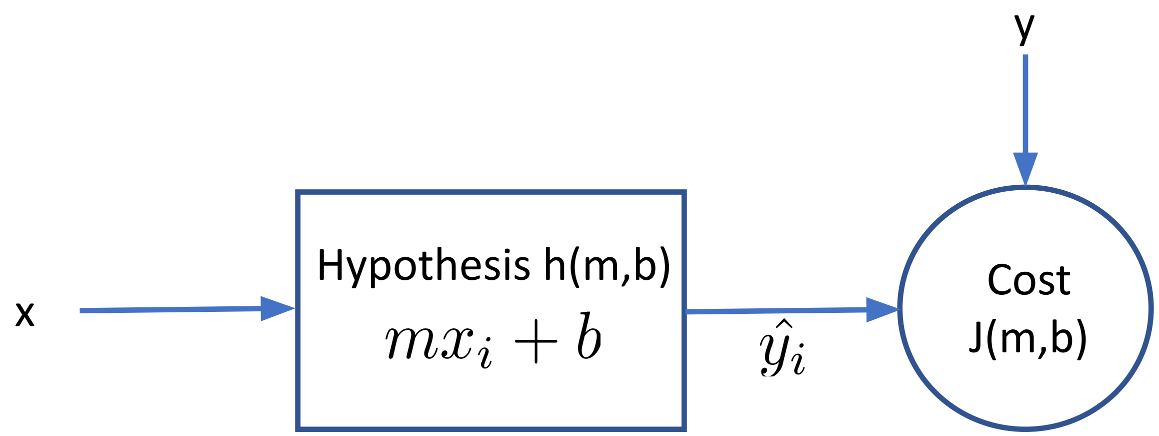
- We want a line which is in some sense the “average line” that represents the data.
- Any ideas as to how we can do it?



https://medium.com/analytics-vidhya/machine-learning-intuition-b4b49a671f65

Cost/Loss Function

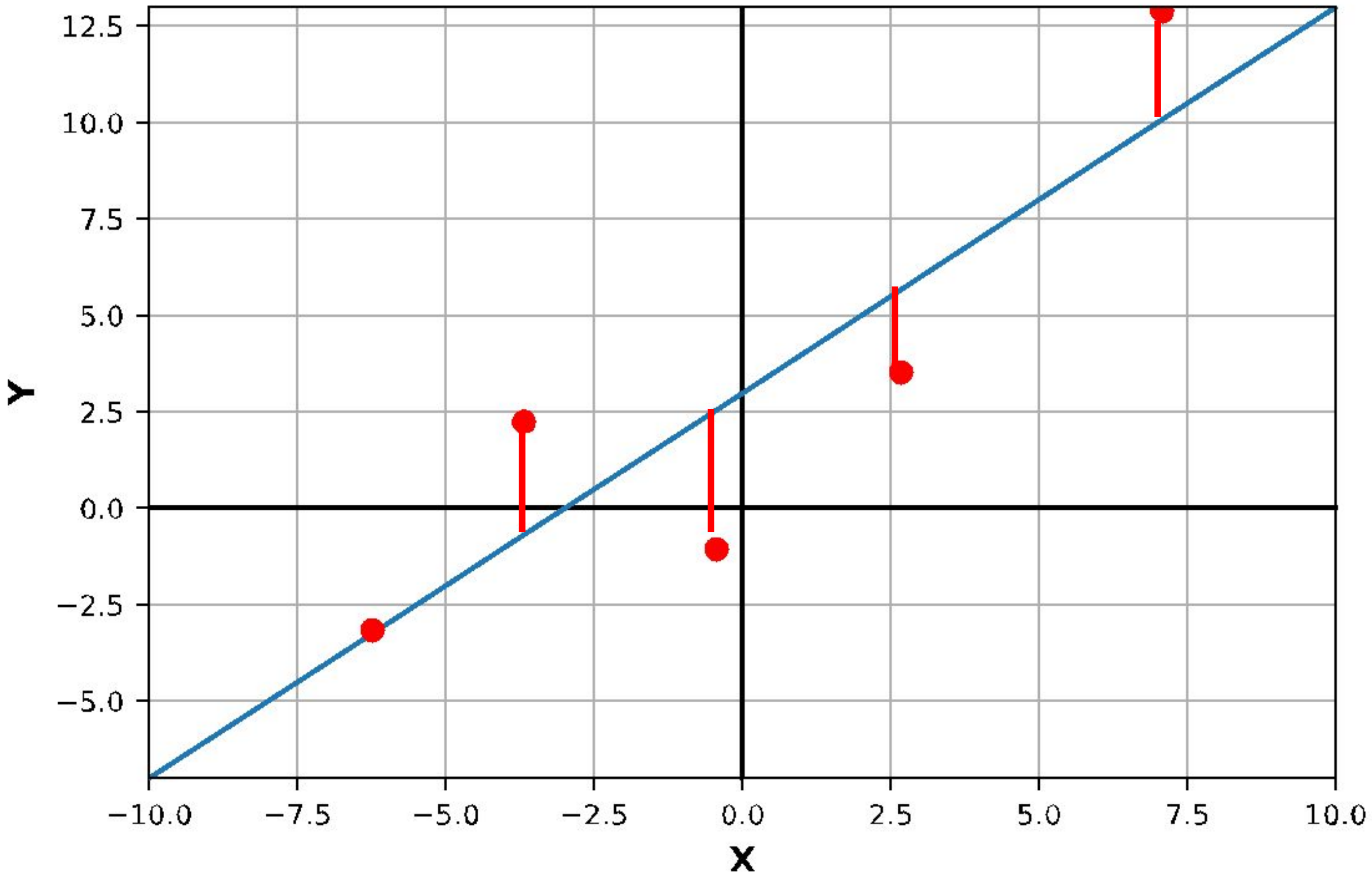
- We want to minimize the discrepancy between our model hypothesis (prediction) and the observed label (ground truth).



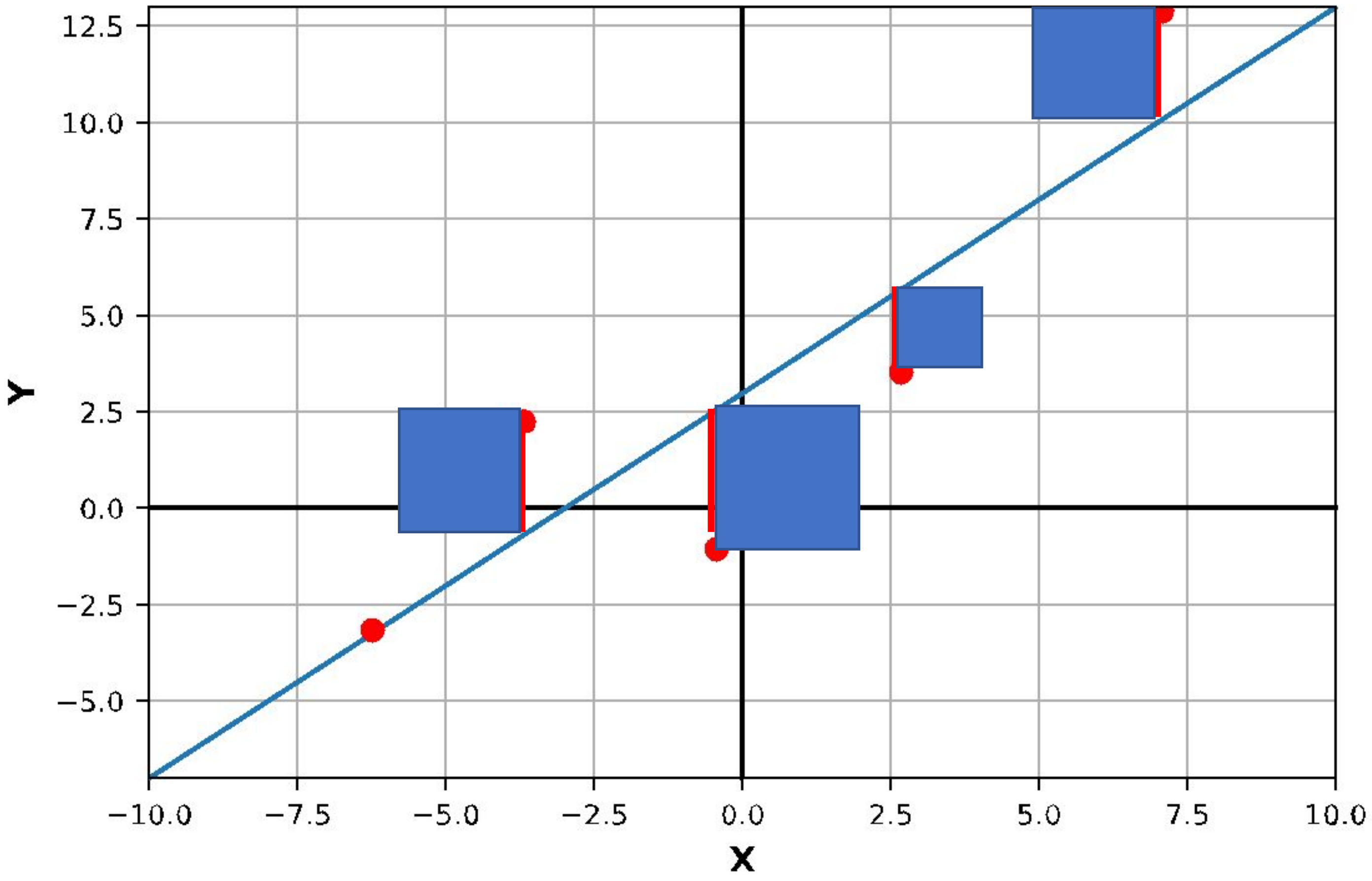
Mean Squared Error (MSE) Loss

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Error Visualization



Error Visualization



Hypothesis Function with 2 Variables

- Let's setup regression for linear function in two variables:
- The hypothesis function is:

$$\hat{y}_i = mx_i + b$$

- Similar to the previous problem our loss function is:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Let's calculate the partial derivatives of the loss function w.r.t. m, b

Gradient of the cost function

- We get the following expressions for the gradient of the cost function

$$\frac{\partial J}{\partial m} = \frac{1}{N} \sum_{i=1}^{N} -2(y_i - \hat{y}_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} -2(y_i - \hat{y}_i)$$

Gradient of the cost function

- Simplifying the above expressions, we get:

$$\frac{\partial J}{\partial m} = \frac{-2}{N}\sum_{i=1}^{N} y_i x_i + \frac{2m}{N}\sum_{i=1}^{N} x_i^2 + \frac{2b}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial J}{\partial b} = \frac{-2}{N}\sum_{i=1}^{N} y_i + \frac{2m}{N}\sum_{i=1}^{N} x_i + \frac{2b}{N}\sum_{i=1}^{N} 1$$

Gradient of the cost function

- Setting the Gradient equal to 0, and solving for m and b, we get

$$\begin{bmatrix} \frac{\sum_i x_i^2}{N} & \frac{\sum_i x_i}{N} \\ \frac{\sum_i x_i}{N} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \frac{\sum_i x_i y_i}{N} \\ \frac{\sum_i y_i}{N} \end{bmatrix}$$

Issues with the Approach

- Calculating gradients like this can quickly become tedious
- Each term on either side of the expression can be written a dot product of two vectors (maybe we can calculate it more efficiently)?

- Let’s explore if we can do something better through vectorization

House Price with more Features

High-dimensional Features

- $x \in \mathbb{R}^d$ for large d
- E.g.,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_d \end{bmatrix} \begin{array}{l} \text{--- living size} \\ \text{--- lot size} \\ \text{--- \# floors} \\ \text{--- condition} \\ \text{--- zip code} \\ \quad\quad\vdots \end{array} \longrightarrow y \text{ --- price}$$

from Machine Learning CS229/STATS229 by Moses Charikar and Chris Ré

Vectorization

- To truly appreciate the power of vectorization. Let’s make the problem a little more complex. The hypothesis function is now

$$\hat{y}_i = w_0 + w_1 x_i^1 + w_2 x_i^2 + \cdots + w_M x_i^M$$

- Where w_j are the unknow weightns of the data x^j features of the input
- Next, we denote the discrepency between y_i and $\hat{y}_i$ as ϵ_i

$$y_i = \hat{y}_i + \epsilon_i$$

Vectorization

- Now let's collect the above equation for all N datapoints

$$y_1 = \hat{y}_1 + \epsilon_1$$

$$y_2 = \hat{y}_2 + \epsilon_2$$

$$\vdots$$

$$y_N = \hat{y}_N + \epsilon_N$$

Vectorization

- Replacing the values of $\hat{y}$, we get:

$$y_1 = w_0 + w_1 x_1^1 + w_2 x_1^2 + \ldots + w_M x_1^M + \epsilon_1$$

$$y_2 = w_0 + w_1 x_2^1 + w_2 x_2^2 + \ldots + w_M x_2^M + \epsilon_2$$

$$\vdots$$

$$y_N = w_0 + w_1 x_N^1 + w_2 x_N^2 + \ldots + w_M x_N^M + \epsilon_N$$

Vectorization

- Collecting the equations in matrix form:

$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^M \\ 1 & x_2^1 & x_2^2 & \dots & x_2^M \\ 1 & x_3^1 & x_3^2 & \dots & x_3^M \\ & . & & & \\ & . & & & \\ & . & & & \\ 1 & x_N^1 & x_N^2 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}
$$

Vectorization

- Notice the rows of the matrix on the right are data samples:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} \dots & \mathbf{x_1} & \dots \\ \dots & \mathbf{x_2} & \dots \\ \dots & \mathbf{x_3} & \dots \\ & . & \\ & . & \\ & . & \\ \dots & \mathbf{x_N} & \dots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

Vectorization

$$\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{N}}$$

- Let’s formalize some notations:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \cdots & \mathbf{x_1} & \cdots \\ \cdots & \mathbf{x_2} & \cdots \\ \cdots & \mathbf{x_3} & \cdots \\ & . & \\ & . & \\ & . & \\ \cdots & \mathbf{x_N} & \cdots \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{X\theta} + \boldsymbol{\epsilon}$$

Cost function for the Vectorized form

- Notice that we are using the MSE cost function:

$$J = \frac{1}{N} \sum_{i} (y_i - \hat{y}_i)^2$$

- Using the definition of epsilon we can write the above as:

$$J = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \frac{1}{N} \sum_{i=1}^{N} (\epsilon_i)^2$$

- Using the definition of dot product the above can be written as:

$$J = \frac{1}{N} \sum_{i=1}^{N} (\epsilon_i)^2 = \frac{1}{N} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

Optimization

- The optimization problem is now:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = \min_{\boldsymbol{\theta}} (\boldsymbol{y} - (\boldsymbol{X\theta}))^T (\boldsymbol{y} - (\boldsymbol{X\theta}))$$

- We will use chain rule to calculate the gradient of the cost function:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \frac{dJ}{d\boldsymbol{\epsilon}} \nabla_{\boldsymbol{\theta}} \boldsymbol{\epsilon}$$

Linear Least Squares

- We get:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \boldsymbol{X}^T 2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})$$

- Setting it equal to zero we can solve for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Practice Time!