Mahindra École Centrale
COLLEGE OF ENGINEERING

Linear Regression

Venkateswara

Disclaimer: Some content and images are borrowed from online sources listed in the reference section

Marks Distribution: Updated

Mahindra École Centrale
COLLEGE OF ENGINEERING

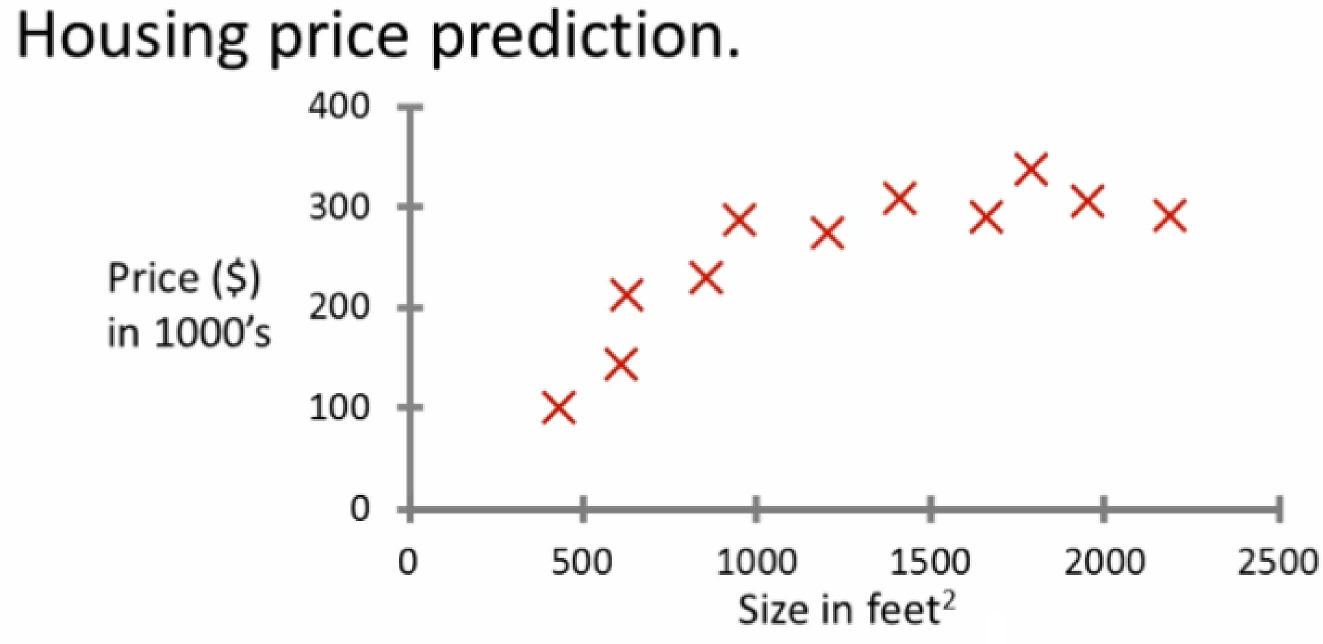
• Mid 1	15 %
• Mid 2	15 %
• Assignments/project	35 %
• Daily Assessment	05 %
• End-semester exam	30 %

Problem Formulation

- Given a the training set of pairs $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$, where $x^{(i)} \in R^d$ and $y^{(i)}$ is a **continuous** target variable, the task is to predict for $x^{(m+j)}, j \geq 1$.

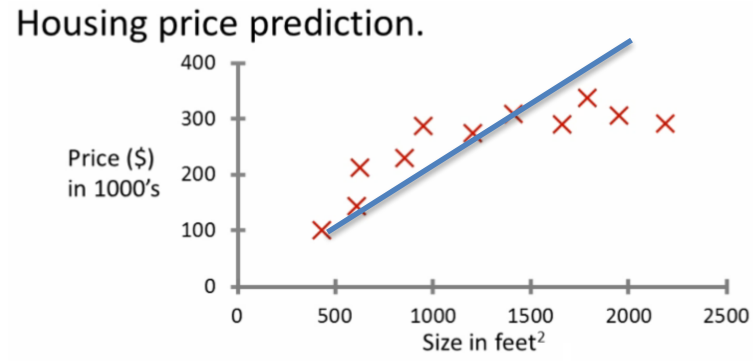
Let us start with an Example

- How do we predict housing prices
  - Collect data regarding housing prices and how they relate to size in feet.

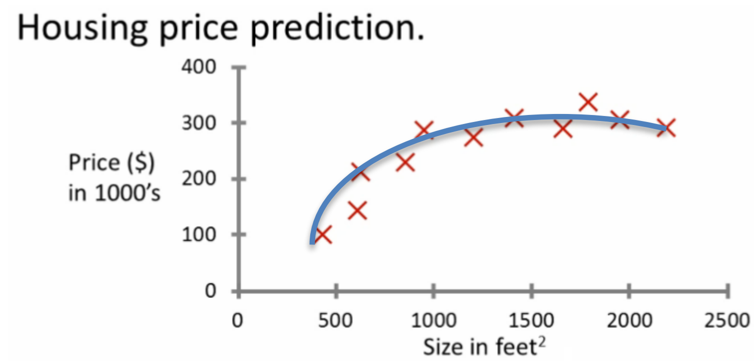


Example problem

- "Given this data, a friend has a house 750 square feet - how much can they be expected to get?“



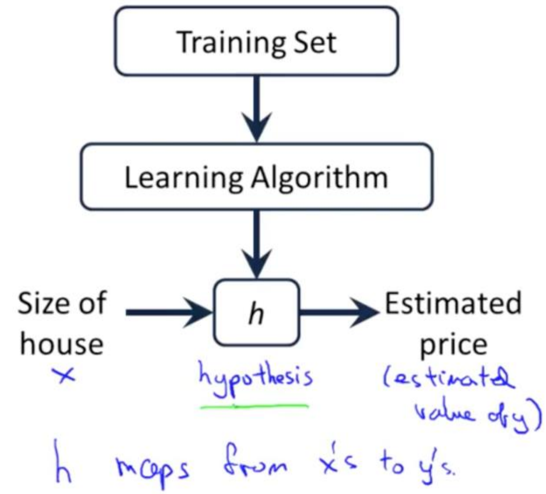
- Straight line through data
  - Maybe $150 000



- Second order polynomial
  - Maybe $200 000

Linear Regression

- With our training set defined - how do we use it?
  - Take training set
  - Pass into a learning algorithm
  - Algorithm outputs a function (denoted *h*)
  - This function takes an input (e.g. size of new house)
    - Tries to output the estimated value of Y
- How do we represent hypothesis *h* ?
  - Going to present *h* as $h_\theta(x) = \theta_0 + \theta_1 x$
  - Means Y is a linear function of x
  - θ_i are **parameters**

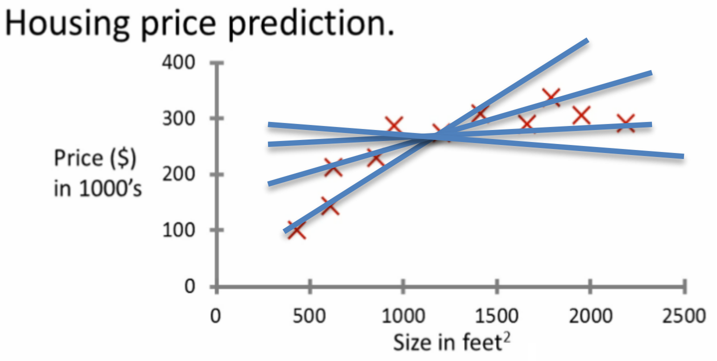


- A linear regression with one variable is also called **univariate linear regression**

- So in summary
  - A hypothesis takes in some variable
  - Uses parameters determined by a learning system
  - Outputs a prediction based on that input

Which line is best ?

- Many lines are possible !!
  Which is the best?
- A cost function lets us figure out how to fit the best straight line to our data.
- What makes a line different ?
  - Parameters θ_0, θ_1
- Which is the best line ?
  - The line that minimizes the difference between the actual and estimated prices.
- What is our objective ?
  - Choose these parameters θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples, i.e. minimize the difference between $h(x)$ and y for each/any/every example.



Objective

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Loss function: $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2$

- $\underset{\theta_0, \theta_1}{Minimize}\ \ J(\theta)$

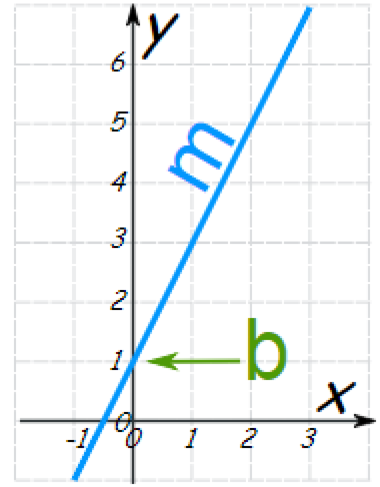
- How do we achieve this ?
  - That is where gradient descent helps us !

Gradient

$$y = mx + b$$

Slope or Gradient

y when x=0
(see Y Intercept)

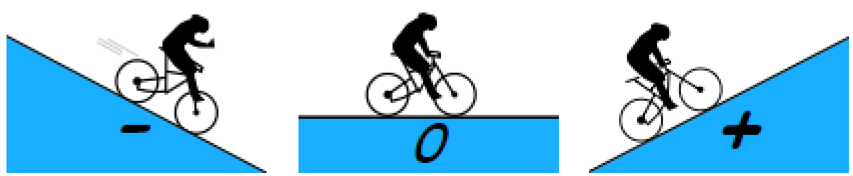


y = how far up

x = how far along

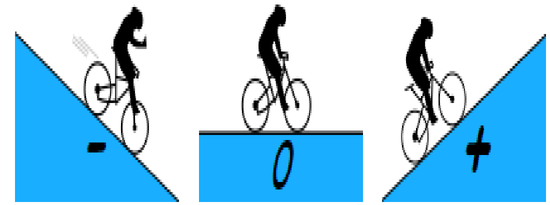
m = Slope or Gradient (how steep the line is)

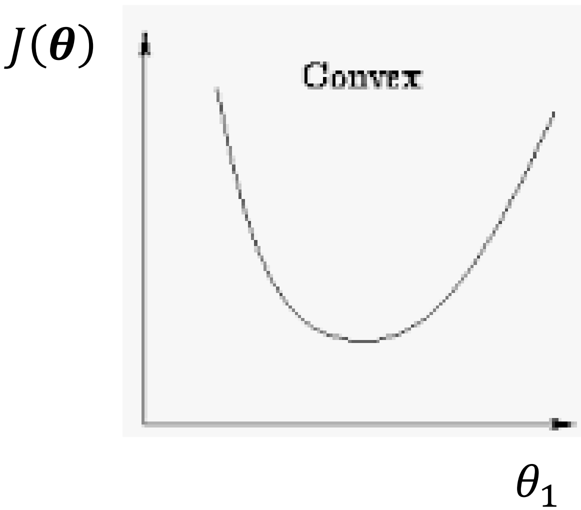
b = value of **y** when **x=0**

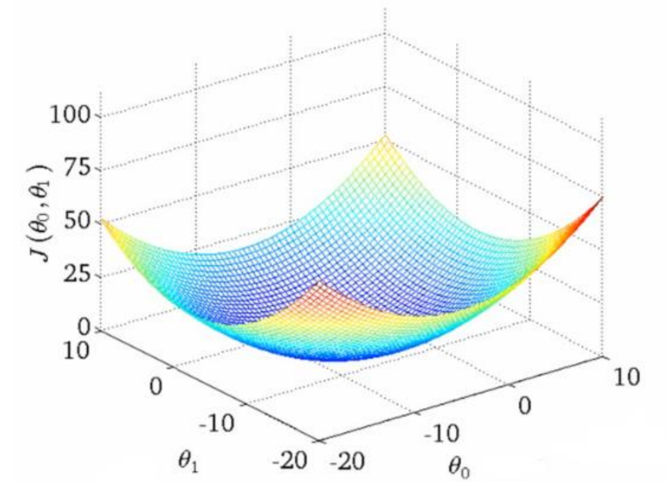


- The gradient is just the vector of partial derivatives

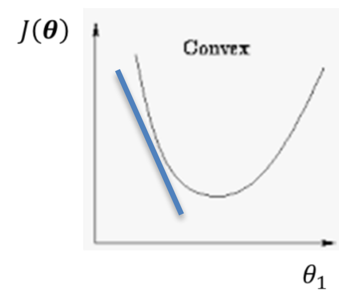
Graphical representation of a convex cost function



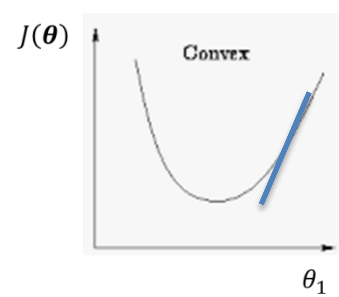




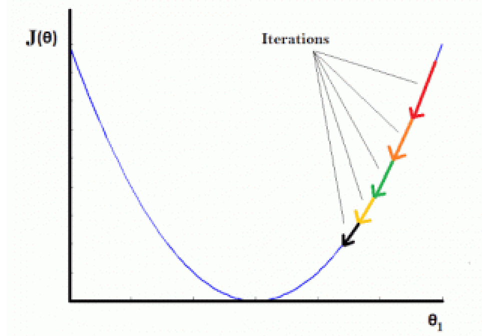
- If the slope is negative we have to increment θ_1 to reach to the minimum value of $J(\boldsymbol{\theta})$.



- If the slope is positive we have to decrement θ_1 to reach to the minimum value of $J(\boldsymbol{\theta})$.



Is this okay ?



Initialize θ_0, θ_1

- Repeat the following until convergence

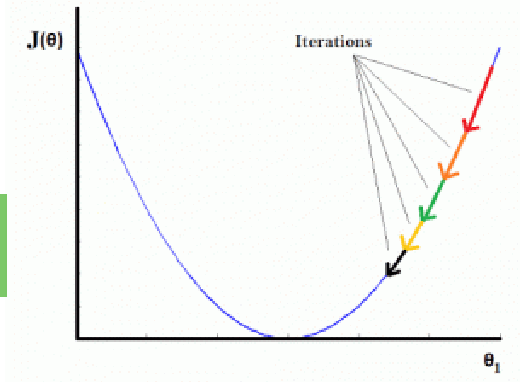
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

Gradient Descent Algorithm

- That is when the gradient is negative we are incrementing θ_j and when the gradient is positive we are decrementing θ_j.

- j=0: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)$
- j=1: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) . x^i$

Gradient Descent



- So, Gradient descent is an optimization algorithm used to find the values of parameters of a function that minimizes a cost function (cost).

- Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

Multivariate Linear Regression

- Linear Regression with multiple input variables/features.

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

- More notations
  - n: number of features (n = 4)
  - m : number of examples (i.e. number of rows in a table)
  - x^i : vector of the input for an example (so a vector of the four parameters for the i^{th} input example)
    - x^3 is, for example, the 3rd house, and contains the four features associated with that house
  - x_j^i The value of feature j in the i^{th} training example
    - x_2^3 is, for example, the number of bedrooms in the third house

What is the form of our hypothesis?

- Previously our hypothesis took the following form.
  - $h_\theta(x) = \theta_0 + \theta_1 x$
    - Here we have two parameters (theta 1 and theta 2) determined by our cost function
    - One variable x
- Now we have multiple features
  - $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
- Let us take $x_0 = 1$ for convenience of notation
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
  - $h_\theta(x) = \theta^T X$

Gradient descent for multiple variables

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Our cost function is

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} (simultaneously update for every $j = 0, \ldots, n$)

Gradient decent for multiple variables

- When n = 1

Repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

(simultaneously update θ_0, θ_1) }

- When n >=1

New algorithm $(n \geq 1)$:

Repeat {

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

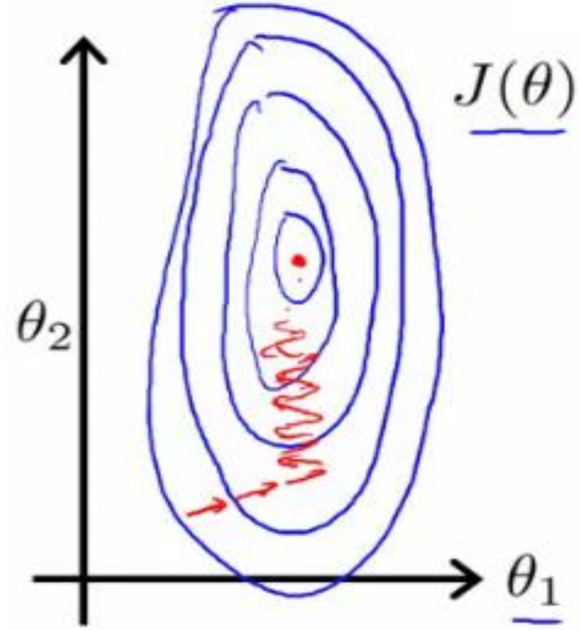
$$\theta_j := \theta_j - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}$$

(simultaneously update θ_j for $j = 0, \ldots, n$) }

Gradient Decent in practice: 1 Feature Scaling

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Range difference of features
  - x1 = size (0 - 2000 feet)
  - x2 = number of bedrooms (1-5)
  - Means the contours generated if we plot θ_1 vs. θ_2 give a very tall and thin shape due to the huge range difference



- Running gradient descent on this kind of cost function can take a long time to find the global minimum
- Feature scaling
  - If you have a problem with multiple features
  - You should make sure those features have a similar scale
    - Means gradient descent will converge more quickly

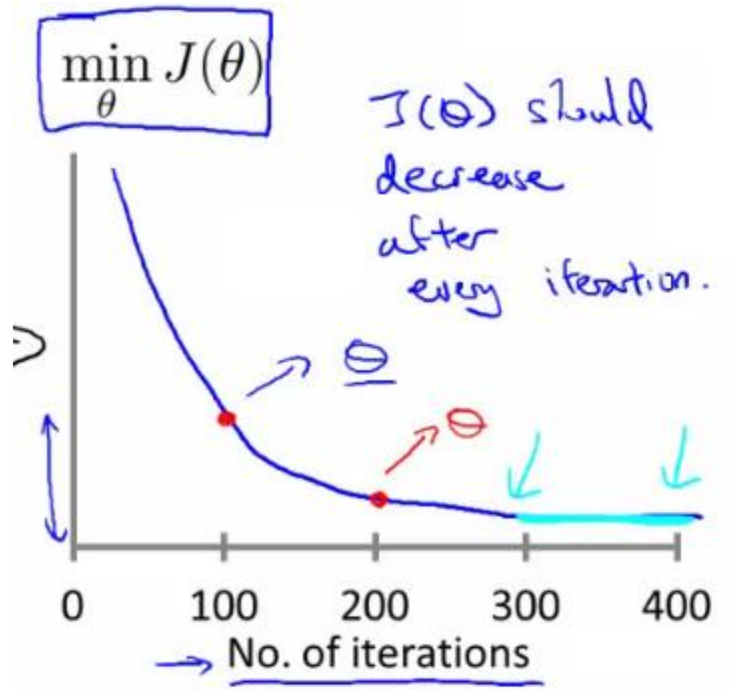
Some feature scaling methods

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Max/Min Scaling
- Mean Normalization
- Z-score Scaling

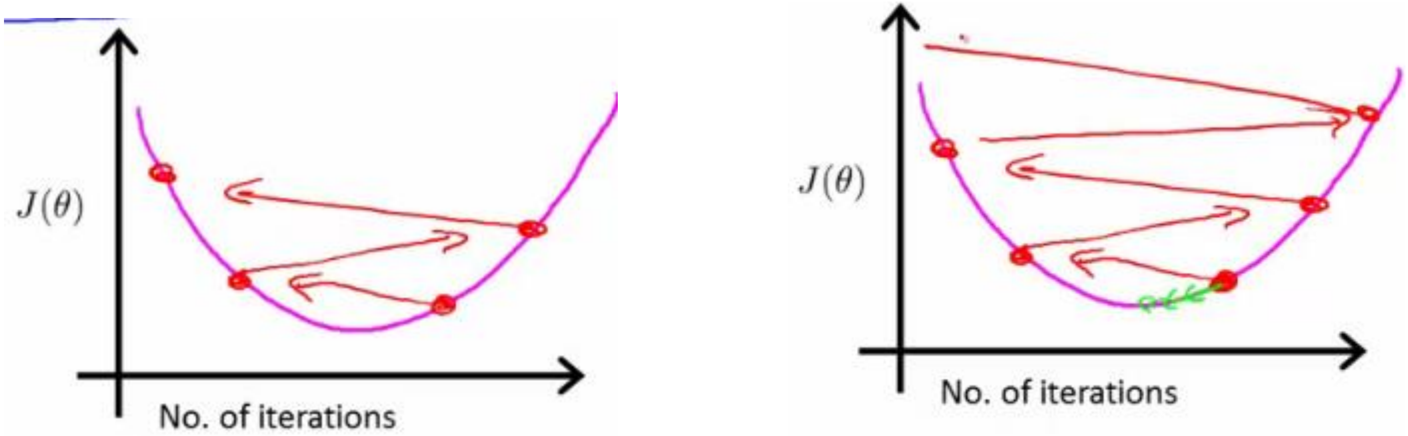
Learning Rate α

- Focus on the learning rate (α)
  - How to chose α ?
- Make sure gradient descent is working
- Plot min J(θ) vs. no of iterations
  - (i.e. plotting J(θ) over the course of gradient descent
- If gradient descent is working then J(θ) should decrease after every iteration



Learning Rate α

- Checking its working If you plot J(θ) vs. iterations and see the value is increasing - means you probably need a smaller α
  - Cause is because your minimizing a function which looks like this



  - But you overshoot, so reduce learning rate so you actually reach the minimum (green line)
-

Vector Notation

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

- $X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m\times(n+1)}$ $\quad Y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_{m\times 1}$

- $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{(n+1)\times 1}$ $\quad h_\theta(x) = x_0\theta_0 + \cdots + x_n\theta_n = X\theta$

- $J = \frac{1}{2m}(X\theta - Y)^T{}_{1\times m}\,(X\theta - Y)_{m\times 1}$

Vectorization

New algorithm $(n \geq 1)$:

Repeat $\{$

$\frac{\partial}{\partial \theta_j} J(\theta)$

$$\theta_j := \theta_j - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}$$

(simultaneously update θ_j for $j = 0, \ldots, n)$ $\}$

- $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right) . x_j^i$
- $= \frac{1}{m} \sum_{i=1}^{m} \left(\left(x^i \theta\right) - y^i\right) . x_j^i$
- $\Theta_{(n+1)\times 1} = \Theta_{(n+1)\times 1} - \frac{\alpha}{m} ({X^T}_{(n+1)\times m} (X\theta - Y)_{m\times 1})$

Closed form solution

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Given X and Y, our aim is to find θ so that $\mathrm{Y} = X\theta$.

- $\mathrm{Y} = X\theta$.
- $\mathrm{X}^{-1}\mathrm{Y} = \theta$
- $\mathrm{X}^{-1}\left(\mathrm{X}^{\mathrm{T}^{-1}}\mathrm{X}^{\mathrm{T}}\right)\mathrm{Y} = \theta$
- $(\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{Y} = \theta$

Why not closed form?

Mahindra École Centrale
COLLEGE OF ENGINEERING

- The problem with this operation is the time complexity of calculating the inverse of a nxn matrix which is O(n^3) and as *n* increases it can take a very long time to finish.
- When n is low (n < 1000 or n < 10000) you can think of normal equations as the better option for calculation theta, however for greater values **Gradient Descent** is much more faster, so the only reason is the time.

- Closed form works better when the input size is smaller. No need to choose α and no need to iterate.

Batch, Stochastic and Mini-Batch Gradient Descent

- In Gradient Descent or Batch Gradient Descent, we use the whole training data for updating theta (or, w).

$$w = w - \alpha \nabla_w J(w) \tag{6}$$

- In Stochastic Gradient Descent, we use only single training example for updating theta.

$$w = w - \alpha \nabla_w J(x^i, y^i; w) \tag{7}$$

- Mini-batch Gradient Descent lies in between of these two extremes, in which we can use a mini-batch(small portion) of training data for updating theta.

$$w = w - \alpha \nabla_w J(x^{\{i:i+b\}}, y^{\{i:i+b\}}; w) \tag{7}$$

References

1) Bishop, Christopher M. "Pattern recognition and machine learning, 2006." Spinger 60.1 (2012): 78-78.
2) http://www.holehouse.org/mlclass/
3) https://www.mathsisfun.com/equation_of_line.html
4) https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3