Linear Regression with Multiple Variables

Introduction

- Multiple variables = multiple features
- In the previous session we had
  - X = house size, use this to predict
  - y = house price
- If we consider more variables (such as number of bedrooms, number floors, age of the house) then x_1, x_2, x_3, x_4 are the four features
  - x_1 - size (feet squared)
  - x_2 - Number of bedrooms
  - x_3 - Number of floors
  - x_4 - Age of house (years)
  - y is the output variable (price)

More notations

- **n** - number of features (n = 4)
- **m** - number of examples (i.e. number of rows in a table)
- $\mathbf{x^i}$ - vector of the input for an example (so a vector of the four parameters for the i^{th} input example)
    - i is an index into the training set
    - So
        - x is an n-dimensional feature vector
        - x^3 is, for example, the 3rd house, and contains the four features associated with that house
- $\mathbf{x_j^i}$ - The value of feature j in the ith training example
    - So
        - x_2^3 is, for example, the number of bedrooms in the third house

With Multiple Features

- What is the form of our hypothesis?
- Previously our hypothesis took the 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
  - $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
- For example
  $h_\theta(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_3 - 2x_4$
  - An example of a hypothesis which is trying to predict the price of a house
  - Parameters are still determined through a cost function

- For convenience of notation, $x_0 = 1$ For every example i you have an additional 0th feature for each example
- So now your **feature vector** is n + 1 dimensional feature vector indexed from 0
  - This is a column vector called x
  - Each example has a column vector associated with it
  - So let's say we have a new example called "X"
- **Parameters** are also in a 0 indexed n+1 dimensional vector
  - This is also a column vector called θ
  - This vector is the same for each example

- Considering this, hypothesis can be written
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
- If we do
  - $h_\theta(x) = \theta^T X$
    - θ^T is an [1 x n+1] matrix
    - In other words, because θ is a column vector, the transposition operation transforms it into a row vector
    - So before
      - θ was a matrix [n + 1 x 1]
    - Now
      - θ^T is a matrix [1 x n+1]

- Which means the inner dimensions of θ^T and X match, so they can be multiplied together as
  - [1 x n+1] * [n+1 x 1]
  - $= h_\theta(x)$
  - So, in other words, the transpose of our parameter vector * an input example X gives you a predicted hypothesis which is [1 x 1] dimensions (i.e. a single value)
- This is an example of multivariate linear regression

Gradient Descent for Multiple Variables

- Fitting parameters for the hypothesis with gradient descent
  - Parameters are θ_o to θ_n
  - Instead of thinking about this as n separate values, think about the parameters as a single vector (θ)
    - Where θ is n+1 dimensional
- The cost function is

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

- Similarly, instead of thinking of J as a function of the n+1 numbers, J() is just a function of the parameter vector
  - $J(\theta)$
- **Gradient descent**

Repeat {

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

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

- Once again, this is
  - $\theta_j = \theta_j$ - learning rate (α) times the partial derivative of $J(\theta)$ with respect to $\theta_{J(...)}$
  - We do this through a **simultaneous update** of every θ_j value
- Implementing this algorithm
  - 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) $\}$

- Above, we have slightly different update rules for θ_0 and θ_1
  - Actually they're the same, except the end has a previously undefined $x_0^{(i)}$ as 1, so wasn't shown
- We now have an almost identical rule for multivariate gradient descent

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 $j = 0, \ldots, n$) $\}$

The interpretation

- We're doing this for each j (0 until n) as a simultaneous update (like when n = 1)
- So, we re-set θ_j to
  - θ_j minus the learning rate (α) times the partial derivative of the θ vector with respect to θ_j
  - In non-calculus words, this means that we do
    - Learning rate
    - Times 1/m (makes the maths easier)
    - Times the sum of
      - The hypothesis taking in the variable vector, minus the actual value, times the j-th value in that variable vector for EACH example

- It's important to remember that

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

- These algorithm are highly similar

Next ....

- Feature scaling (normalization)
- How to choose a learning rate
- How to check if GD is working?
- Features and polynomial regression