Computing for Data Sciences
Lecture 6

Introduction to Regression Analysis

Suppose, a bunch of data points are given about pricing of houses (Classic Boston training dataset[1]) and the cost of house is required to be predicted.

X_1	X_2	Y
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2
. . .	. . .	. . .
x_{n1}	x_{n2}	y_n

Table 1: Training dataset

Here, X_1 is location, X_2 is size of house, Y is cost of the house (output or target variable) and n is the number of data points (also referred to as number of training examples). $X_1$and X_2 are known as input variables or features and together with corresponding output variable, they are called training example.

Now, the general approach to solve the problem would be to try and find out a function, Y = f(X_1,X_2) keeping in mind, that the data is prone to error. We can fit a model, maybe a straight line passing through this data such that for every $X_1$and X_2, we have a unique Y. This process is an example of a regression analysis, a type of supervised learning algorithm. Regression refers to the fact that we are predicting a real-valued output namely the price. We are taking location and size as input variables and trying to map the price (output) such as to get a continuous result function.

[1] The dataset of housing prices is called training dataset which can be used to predict prices of the houses.

Curve fitting vs ability to model

The high degree of fit of a curve on the points does not always imply a higher utility for understanding the underlying trend. Care should be taken while extrapolating the curve obtained from regression.

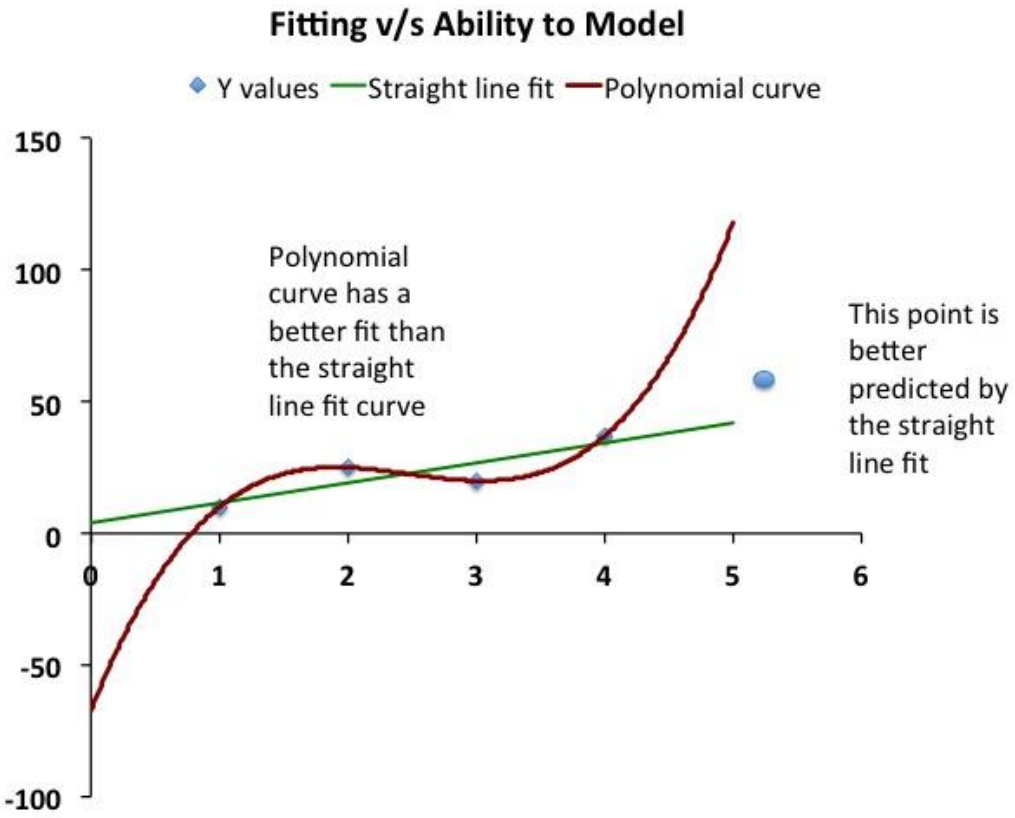


Fig 1: Fitting v/s Ability to Model

Idea of finding f & Hypothesis function

The function f maps the size and location of the house to the prediction about price of house.

$$f \rightarrow h_\theta(X_1, X_2) = some\ linear\ combination\ of\ parameter\ \theta = \theta_0 + \theta_1 X_1 + \theta_2 X_2$$

Here, h is known as hypothesis which maps x's to y's. θ_0 balances the y-intercept as we are not sure if data passes through 0 or not (mean of data may not be 0).

General Representation of Hypothesis function is $h_\theta(x) = \theta_0 + \theta_1(x)$, where θ_0 is the y-intercept and θ_1 is slope of line fitted.

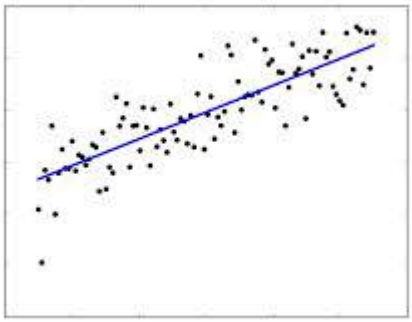


Figure 2: Best-fitted line

Hypothesis function tells that y is a function of x. In the above picture, we are predicting y (on vertical axis) which is some linear function of x (on horizontal axis). This is called fitting and is an example of linear regression with one variable. The idea is to choose parameters such that we get "right" y for the training example.

Least Square Errors

With different choices of parameters, we get different hypotheses functions. So, how do we determine the best fit line through our data?

We have to choose the values for the parameters such that, given the x's in the training set, we make reasonably accurate predictions for the y values, i.e., we have to minimize the difference between predicted value and actual value of house i.e. h(x) and y.

Least Squared Error function or Cost Function, $J(\theta_0 , \theta_1)= \frac{1}{2} * \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2$

Where, (x_i , y_i) represents the ith training example.

For Classic Boston Training Dataset,

$J(\theta_0 , \theta_1)= \frac{1}{2} * \sum_{i=1}^{m} (h_\theta(x_i^{(1)}, x_i^{(2)}) - y_i)^2$

Hypothesis function $h_\theta(x)$ is a function of x for fixed θ_0 , θ_1, whereas $J(\theta_0 , \theta_1)$ is a function of parameters θ_0 , θ_1.

Gradient Descent:

For minimizing the cost function J(θ), gradient descent provides an iterative method to find out the minimum point in J(θ) curve. On successive iterations, the algorithm converges to the minimum (which is not necessarily the global minimum of the J(θ)). It has been observed that gradient descent works quite well even for large number of features.

The algorithm can be stated as follows:

while (convergence criterion is not met) {

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

where

θ = Feature value,

α= Learning rate, $\alpha > 0$ (it determines the step size of change in value on each iteration)

Irrespective of the value of initial starting point (guess) chosen, the algorithm will certainly converge to minimum.

Eg. if J(θ) is expressed in two-Dimension :

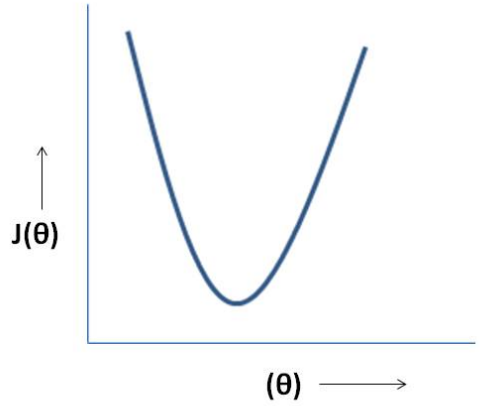


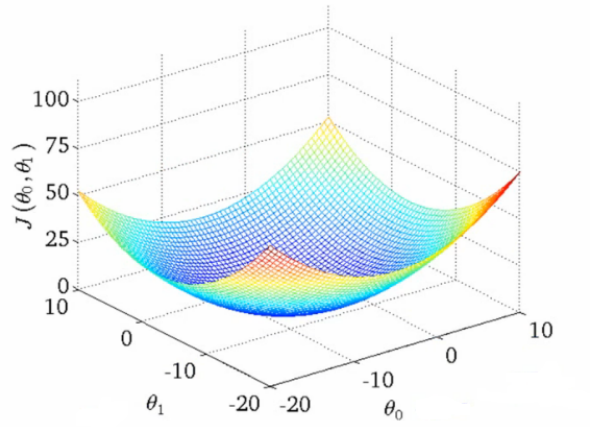
Figure 3: J(θ) V/S θ for single parameter

If initial guess is towards right of the minimum point, then the slope (partial derivative term of the algorithm) will be positive and since learning rate is always positive, so value of J(θ) will decrease (downward along the curve in this case).

If initial guess is towards left of the minimum point, then the slope (partial derivative term of the algorithm) will be negative and since learning rate is always positive, so value of J(θ) will go towards minimum (downward along the curve in this case).

For 2 parameter case,

Cost function $J(\theta_0, \theta_1)$ = (1/2)* $\Sigma(\theta_0+\theta_1.x_i-y_i)^2$



Plotting the $J(\theta_0, \theta_1)$ function in 3 dimensional space, we get a surface which is bowl shaped.

In this figure, the value of cost function is given by the height on varying (θ_0 and θ_1).

Figure 4: Contour for 2 parameter case (Reference: Source)

Batch Gradient Algorithm:

On plotting the contour plot of the above surface, we get the concentric ellipse-like view of the surface wherein the value of J(θ) is decreasing on moving across these ellipses.

The objective is to reach the centre of the innermost ellipse on varying both θ_0 and θ_1.

In gradient descent, we simultaneously update both the parameters θ_0 and θ_1.

In the below code, on varying θ_0, we store the value of updated value in temporary variable so that the update in value of θ_0 does not affect the updation of θ_1. Finally, the changes in both parameters are reflected later using temporary variables.

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

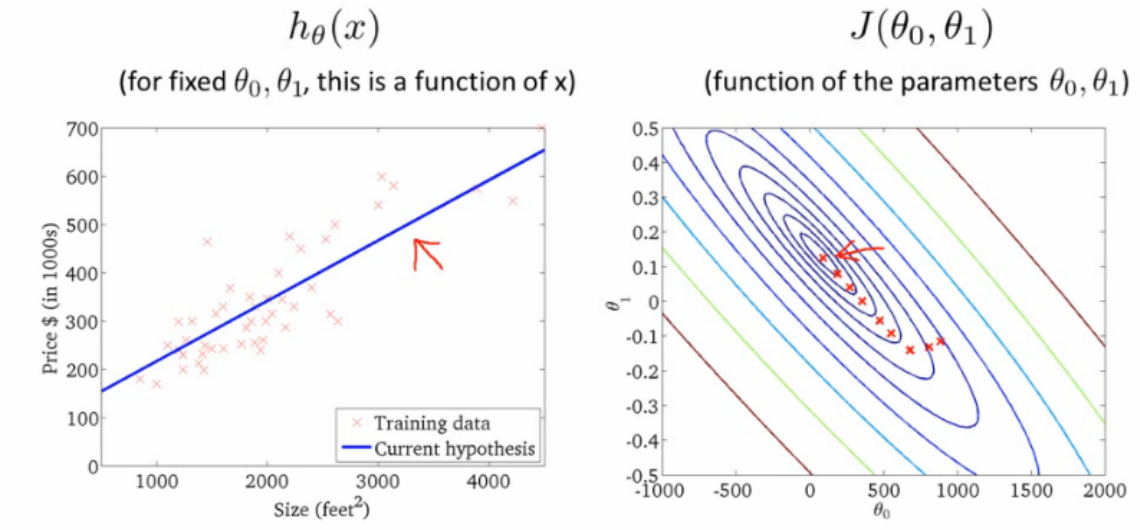


Figure 5: Batch Gradient (Reference: Source)

Converging criteria for Gradient Descent:

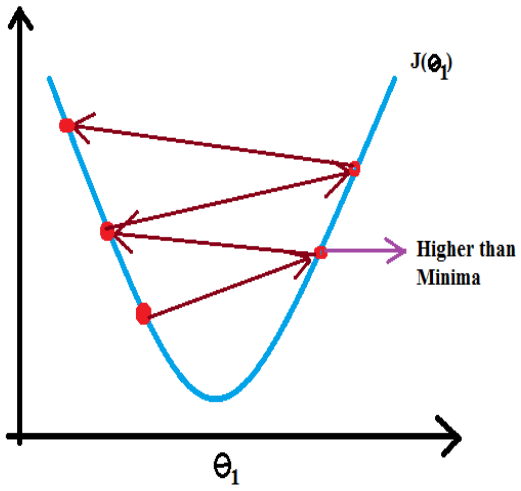
Convergence of Gradient Descent can be done by

- *Plotting a graph between J(θ) and number of iterations(k)*: On increasing the number of iterations for which Gradient Descent runs, we expect the J(θ) to decrease with it. If we go on increasing number of iterations, then we'll reach a point where J(θ) does not go down much more. Curve will look flattened after certain number of iterations. Then it means the algorithm has converged to a minimum.

  Correctness of Gradient Descent: If graph between J(θ) and number of iterations(k) is observed to be increasing, then the implementation of algorithm is certainly wrong.
- *Automatic Convergence Test*: Decide a threshold value of permissible error. E.g., Gradient Descent can be stopped when difference between value of J(θ) in previous and current iteration is $\leq$ 0.0001.

Effect of Learning Rate:

- If learning rate is too high, it may overshoot the minimum and the algorithm will not converge to minimum.

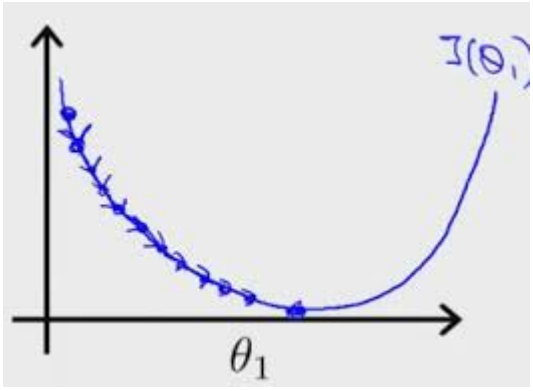


In the figure alongside, the larger value of learning rate results in divergence of algorithm from the minimum.

Solution: Decrease the value of learning rate.

Figure 6: (Reference: Source)

- If learning rate is too low, then convergence of algorithm will be slow.



In the figure alongside, too small value of learning rate is resulting in very slow convergence towards the minimum.

Figure 7: (Reference: Source)

Note: problem of optimum value of learning rate will not occur in successive iterations of Gradient Descent since the algorithm takes care of smaller step size.

Concerns with the use of Gradient Descent Algorithm

- The algorithm may not always converge
- The minima obtained may be a local minima and not the global minima

- The final solution (final point which we get from gradient descent algorithm) depends on the choice of the initial point and step size

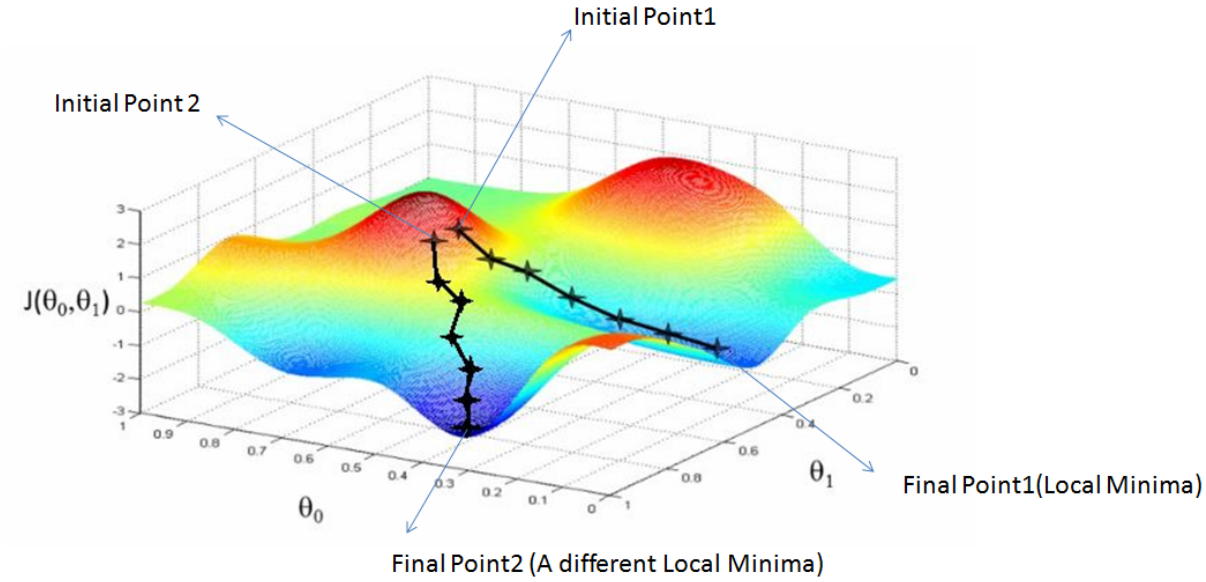


Fig 8: Different initial points leading to different solutions (Reference: Source)

Linear Algebra point of view:

Unlike gradient descent (which is an iterative approach), the normal equation method of linear regression is analytical approach, i.e., we can find the solution in single step.

$$h(\theta) = \theta^T X$$

$$J(\theta) = \frac{1}{K} * \sum_{i=1}^{K} (\theta^T X_i - y_i)^2$$

For example the θ and X matrices for a 2 dimension are:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & X_{11} \\ 1 & X_{12} \end{bmatrix}$$

$$error = 1/\sqrt{K} \begin{bmatrix} \theta^T X_1 - y_1 \\ \theta^T X_2 - y_2 \\ \vdots \\ \theta^T X_K - y_K \end{bmatrix}$$

$$J(\theta) = ||(error)||^2$$

$$X \quad \theta \quad = \quad h_\theta \quad \approx \quad Y$$

$$\begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{12} & a_{22} \\ \vdots & & \\ 1 & x_{1K} & x_{2K} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{K-1} \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_K \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

Advantage of the analytical approach over the iterative approach: Using the analytical approach we can hope to get to the best result as against iterative solution where depending on the initial value and step size, the algorithm can converge to a local minima and not the global minima.

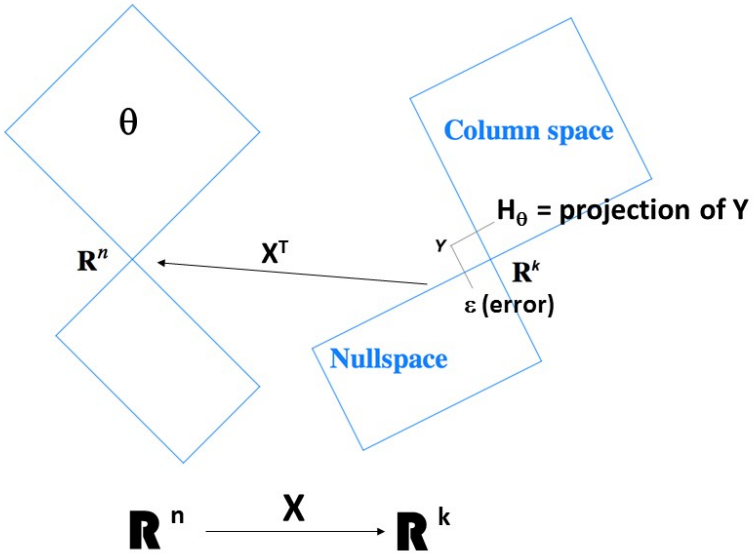


Figure 9: Projection of vector Y into Column Space of X and Null Space of X^T

Y may or may not fall in the column space. If Y falls inside the column space, then exact Y has been found and RMS error is zero. If Y falls outside the column space, then there is some error which lie in the null space of X^T.

$$error \; = \; \varepsilon \; = \; X_\theta \; - \; Y$$

Since the error lies in the null space of X^T

$$X^T(X\theta \; - \; Y) \; = \; 0$$

$$X^T X\theta \; - \; X^T Y \; = \; 0$$

$\theta \; = \; (X^T X)^{-1} X^T Y$ …….Normal Equations

To avoid the calculation of inverse the below equation can be fed in matrix solver

$$(X^T X)\theta \; = \; (X^T Y)$$