Machine Learning Project 2

Project Report

House Price Prediction using Polynomial Regression

By
Qazi Umer Jamil
RIME 2019
NUST Registration No: 317920

NUST School of Mechanical and Manufacturing Engineering
Sector H-12, Islamabad, Pakistan

Dated: 09-11-2019

Contents

List of Figures

List of Tables

1. Introduction

The task is to implement and train a linear regression model on the given data for predicting housing prices.

1.1 Environment

MATLAB R2018a is used for model implementation and training.

1.2 Files

The attached zip file contains the following files:

- `main.m`
  Main file of the program
- `createHyp.m`
  To create models with higher order polynomials
- `createPoly.m`
  To create higher order polynomial features of a feature vector
- `learningCurve.m`
  To calculate training and validation errors for learning curve
- `validationCurve.m`
  To calculate training and testing errors with various lambda values and produce the validation curve
- `dataCleaning_featureEngg.m`
  Implementation of Data Cleaning and Feature Engineering process
- `featureNorm.m`
  Feature Normalization
- `GD.m`
  Implementation of Gradient Descent Algorithm
- `calCost.m`
  Implementation of Cost Function

- `Prediction.m`
  Implementation of Prediction Function
- `modelEval.m`
  Implementation of Model Evaluation Metrics (RMSE, R-Squared)
- `sumFunc.m`
  To calculate sum of columns of input matrices.
- `mulFunc.m`
  To multiply two given matrices.
- `meanFunc.m`
  Calculates the mean of of individual columns of input matrix. Returns a row matrix.
- `createMatrix.m`
  Create a Matrix of input rows, columns, element value.
- `data3.csv`
  Given Dataset

1.3 Data Set

The data is provided in a *.csv* format. To make the data processing task easier, the header of the file (which is the 1st row) and the first two columns containing Id and date features are removed manually (Further explained in section 2.2). The new data is saved in the file *data.csv* and is provided with the source code.

1.3.1 Data Headers

The data headers [2] in the given *.csv* file are described in Table 1.1

Table 1.1: Data Headers

Sr. No	**Variable**	**Description**
1	Id	Unique ID of each home sold
2	Date	Date of the home sale
3	Price	Price of each home sold
4	Bedrooms	Number of Bedrooms
5	Bathrooms	Number of Bathrooms
6	Sqft_living	Square footage of apartment interior living space
7	Sqft_lot	Square footage of land space
8	Floors	Number of floors
9	Waterfront	A dummy variable whether the apartment was overlooking the waterfront or not
10	View	An index from 0-4 of how good the view is
11	Condition	An index from 1-5 on condition of the apartment
12	Grade	An index regarding quality level of construction and design
13	Sqft_above	The square footage of the interior housing space that is above ground level
14	Sqft_basement	The square footage of the interior house space that is below ground level
15	Yr_built	The year the house was initially built
16	Yr_renovated	The year of the house's last renovation
17	Zip Code	What zip code area are the house is in
18	Lat	Latitude
19	Long	Longitude
20	Sqft_living15	The square footage of interior housing living space for the nearest 15 neighbours
21	Sqft_lot15	The square footage of the land lots of the nearest 15 neighbours

2. Data Processing

2.1 Data Cleaning

By just scrolling through the data, I found out that there are some training examples in which the number of bedrooms is 0 or 33, which is certainly not possible in reality. Therefore, it is pertinent to remove such training examples. I manually removed every training example which has bedrooms either 0 or greater than 11.

2.2 Feature Selection and Engineering

The Id feature is not relevant in training the linear regression model. Similarly, the date on which a house sold cannot merely tell us about the price as there can be a case when the seller only found a buyer after a long time and this did not have any effect on the price of the house whatsoever.

Also, the feature 'Sqft_above' is linearly dependent on 'Sqft_living' and 'Sqft_basement' features. The relationship is given in equation 2.1

$$Sqft\_above = Sqft\_living - Sqft\_basement \tag{2.1}$$

Having correlated features in the data can lead to the poor performance of the linear regression model. Hence, the feature vector 'Sqft_above' is removed from the data set.

The lat, long coordinates represent a point in 3-dimensional space. Thus, these two features can not be treated independently in training the regression model. One way to address this problem is by using the long lat coordinates of the house to calculate the distance from the lat, long coordinates of a reference location. One such reference location can be the economic hub/center of the city. As this was not given, I took the reference location of the house, which has the maximum price. If the price of a house is higher, chances are the house located in its vicinity will also tend to have higher prices. A house that is farther from the location of the house of the highest price will tend to have a lesser price.

The lat, long coordinates of a house which has a maximum price in the given data can be easily obtained and are implemented in *dataCleaning_featureEngg.m* file. Once the reference lat, long coordinates are known, we can use the Haversine formula to calculate the distance between the two given sets of lat-long coordinates. The Haversine formula is given in equation 2.2

$$dist = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{lat2 - lat1}{2}\right) + \cos(lat1)\cos(lat2)\sin^2\left(\frac{long2 - long1}{2}\right)}\right) \tag{2.2}$$

Here r is the radius of sphere i.e., Earth. (lat1, long1) And (lat2, long2) are the latitudes and longitudes of two locations. A new feature vector is created (called dist), which is the distance of a house from the house of the highest price. This new feature vector is appended to the given data, and the feature vectors containing lat, long coordinates were removed as there is no need to include them in the data.

The 'Zipcode' feature represent categorical data, and this type of data is more useful in models such as Decision Trees and K-Means Clustering. Since we already have encoded the geographic information of the house using its lat, long coordinates, I decided to remove this feature vector too. Furthermore, Yr_built and Yr_renovated feature vectors also represent categorical data, and hence these were removed from the data set. Table 2.1 shows the selected features for the model.

Table 2.1: Selected Features

Sr. No	Feature	Description
1	Bedrooms	Number of Bedrooms
2	Bathrooms	Number of Bathrooms
3	Sqft_living	Square footage of apartment interior living space
4	Sqft_lot	Square footage of land space
5	Floors	Number of floors
6	Waterfront	A dummy variable whether the apartment was overlooking the waterfront or not
7	View	An index from 0-4 of how good the view is
8	Condition	An index from 1-5 on condition of the apartment
9	Grade	An index regarding quality level of construction and design
10	Sqft_basement	The square footage of the interior house space that is below ground level
11	Sqft_living15	The square footage of interior housing living space for the nearest 15 neighbours
12	Sqft_lot15	The square footage of the land lots of the nearest 15 neighbours
13	dist	Distance of house from the house of highest price

2.3 Testing and Training Dataset

Now, the data is split in a random manner into the training and testing set. The training set contains 80% of the given data, while the testing set contains the remaining 20% of the provided data. After data splitting, corresponding feature and label vectors are created.

2.4 Feature Normalization

As the given data set contains features that vary highly in their ranges, it is necessary to normalize these. If features are not normalized and have different ranges, the result is in the slow convergence of the Gradient Descent algorithm. To address this, all the features are normalized using Standardization. Feature normalization is done for all the training examples (both testing and training). Standardization basically replaces the values by their corresponding Z scores given by equation 2.3

$$x^{'} = \frac{x - \bar{x}}{\sigma} \tag{2.3}$$

The new features created have their mean $\mu = 0$ and standard deviation $\sigma = 1$.

3. Regularized Polynomial Regression

In polynomial regression, the relationship between the independent variable x and the dependent variable y is modeled as nth degree polynomial in x. The general form of p-th order polynomial regression for a feature vector x is given in equation 3.1

$$h_\theta(x) = \theta_0\, x + \theta_1\, x + \theta_2\, x^2 + \theta_3\, x^3 + \theta_4\, x^4 + \cdots + \theta_n\, x^p \tag{3.1}$$

Where $x0 = 1$. $h_\theta(x)$ is the dependent variable, and x, x^2, x^3 . . . , x^p are the independent variables. θ_0, θ_1, θ_2, θ_3, . . . , θ_n, are regression coefficients/parameters. We need to find the values of regression parameters that best fit our training set. This can be done using the Gradient Descent algorithm

3.1 Generating higher-order polynomial features

The generation of higher-order polynomial features is implemented in a file createPoly.m. This file takes a matrix of size m by n, and a variable poly(which represent the degree of polynomial), and returns a matrix that contains features vectors up-to their poly-th order. Further explanation of the working of code is in the code file.

3.2 Regularized linear regression cost function

The cost function for regularized linear regression is given as follows:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{3.2}$$

Here λ is a regularization parameter, and it puts a penalty on the cost J. It basically helps prevent overfitting our model to the training data.

3.3 Gradient Descent Algorithm

Gradient Descent is an iterative optimization algorithm. The objective of linear regression is to minimize the cost function given in equation 3.2. In gradient descent, we simultaneously update $\theta_0, \ldots, \theta_n$ for a specific number of iterations for every $j = 0, \ldots, n$. In case of regularized linear regression, update rule for θ is as follows:

for j = 0;

$$\theta_j \;= \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 x_j^{(i)} \tag{3.3}$$

for $j \geq 1$

$$\theta_j \ = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 x_j^{(i)} + \frac{\lambda}{m} \theta_j \tag{3.4}$$

Definition of some related terms is as follows:

- Learning rate (α): It determines how fast and slow the gradient descent algorithm converges. I tested multiple learning rates (0.1, 0.01, 0.001) and settled on using 0.02.
- m = number of examples in the training set.
- λ: Regularization parameter.

The gradient descent algorithm is implemented in *GD.m*, and the computation of cost function is being implemented in *calCost.m*

3.4 Model Selection

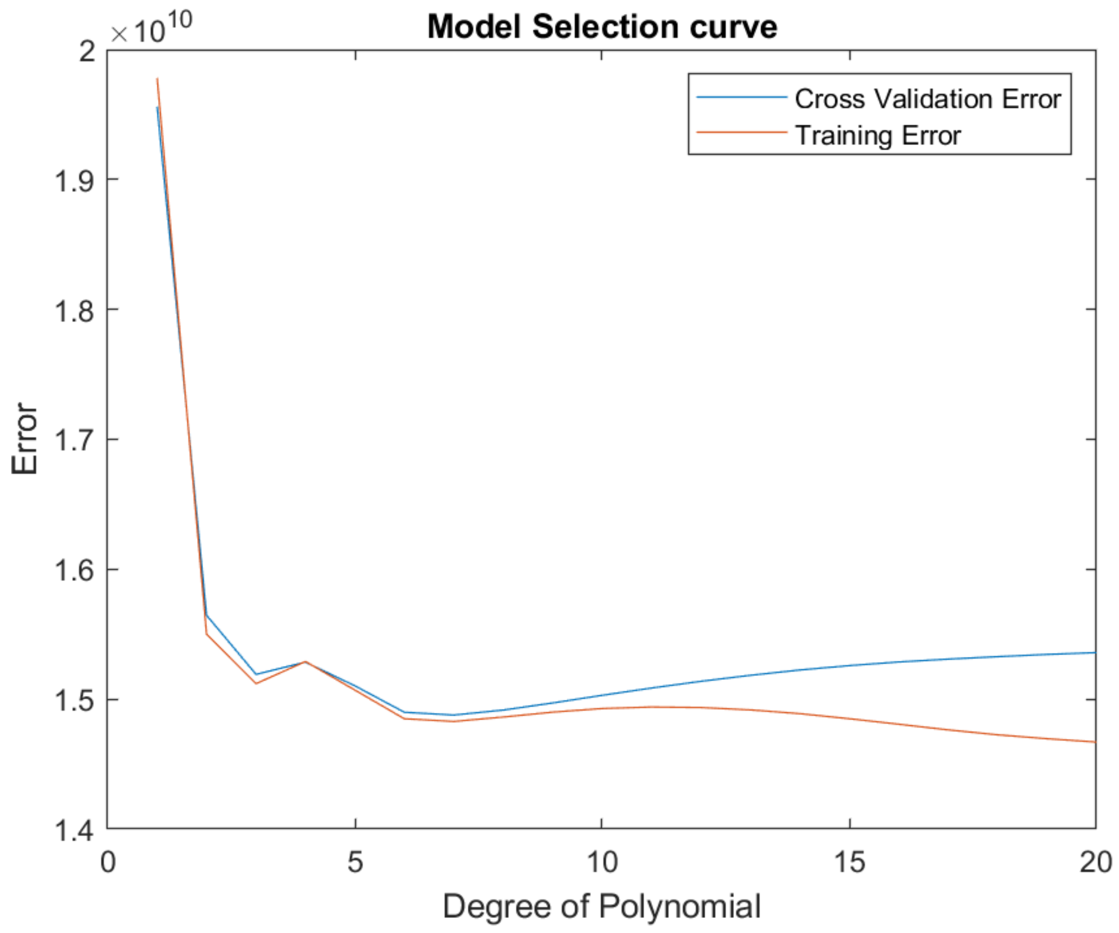
Here, the model selection means choosing the degree of a polynomial of our model/hypothesis. Model selection is done using the graph of validation and training error vs. degree of a polynomial. Various models up to pth order are evaluated, and their respective validation and training errors are calculated.

3.4.1 Model Selection Curve

The model selection curve in figure 3.1 shows the training error, and validation error as a function of the complexity(degree of polynomial) of the model considered. Up to 20th-degree polynomial based models/hypotheses are evaluated in our case. λ is assumed zero for the model selection process.

As shown in the figure, the training error is decreasing as we increase the complexity of our model. The cross-validation error, on the other hand, first decreases and then starts to increase, which means we are starting to suffer from overfitting.

Figure 3.1: Model Selection Curve

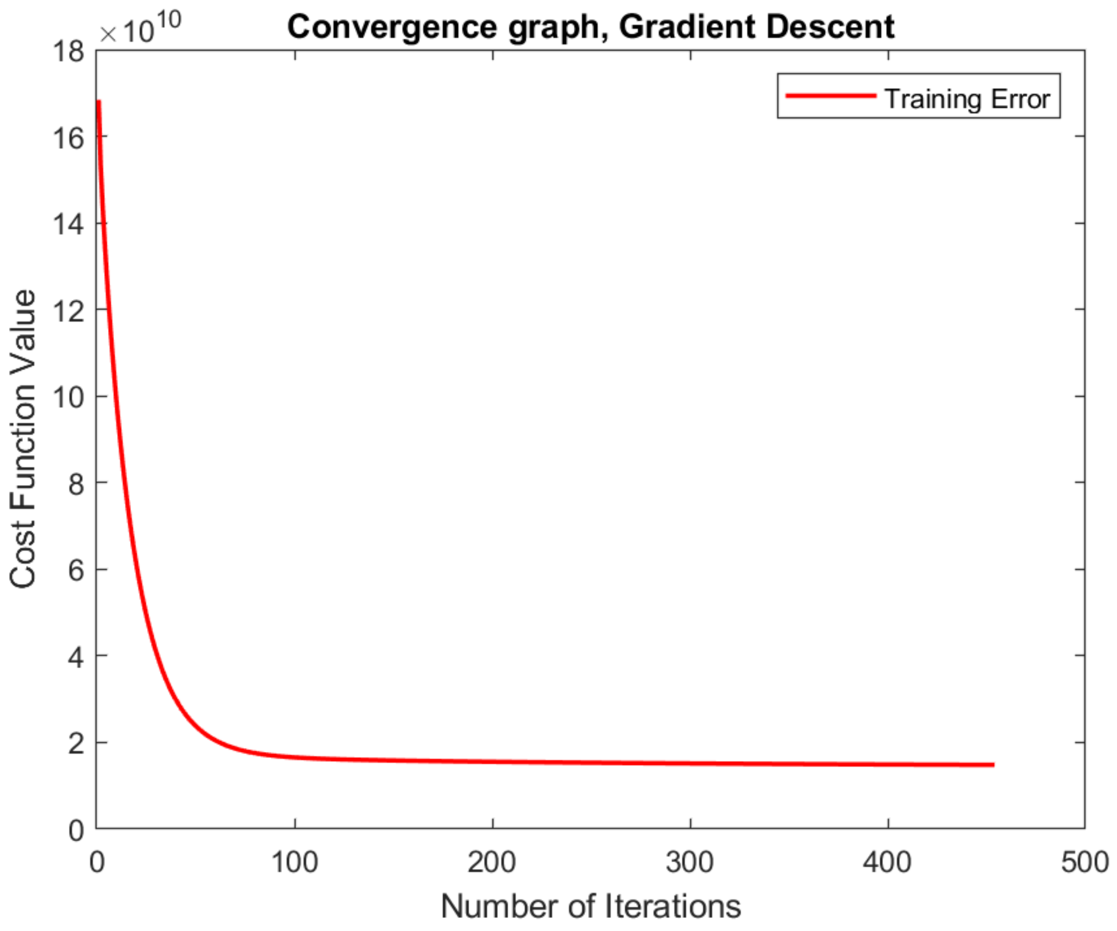


The Model Selection Curve shows that the cross-validation is minimum for a model of 7th-degree polynomial. Hence, this 7th-degree polynomial regression model is chosen for further process.

3.4.2 Convergence of Cost Function

Figure 3.2 shows the graph of Cost Function Values vs. Number of iterations for a 7th-degree polynomial based model. The graph also confirms that our cost function is decreasing with the number of iterations, which means that our algorithm is converging.

Figure 3.2: Cost Function Values vs. Number of iterations



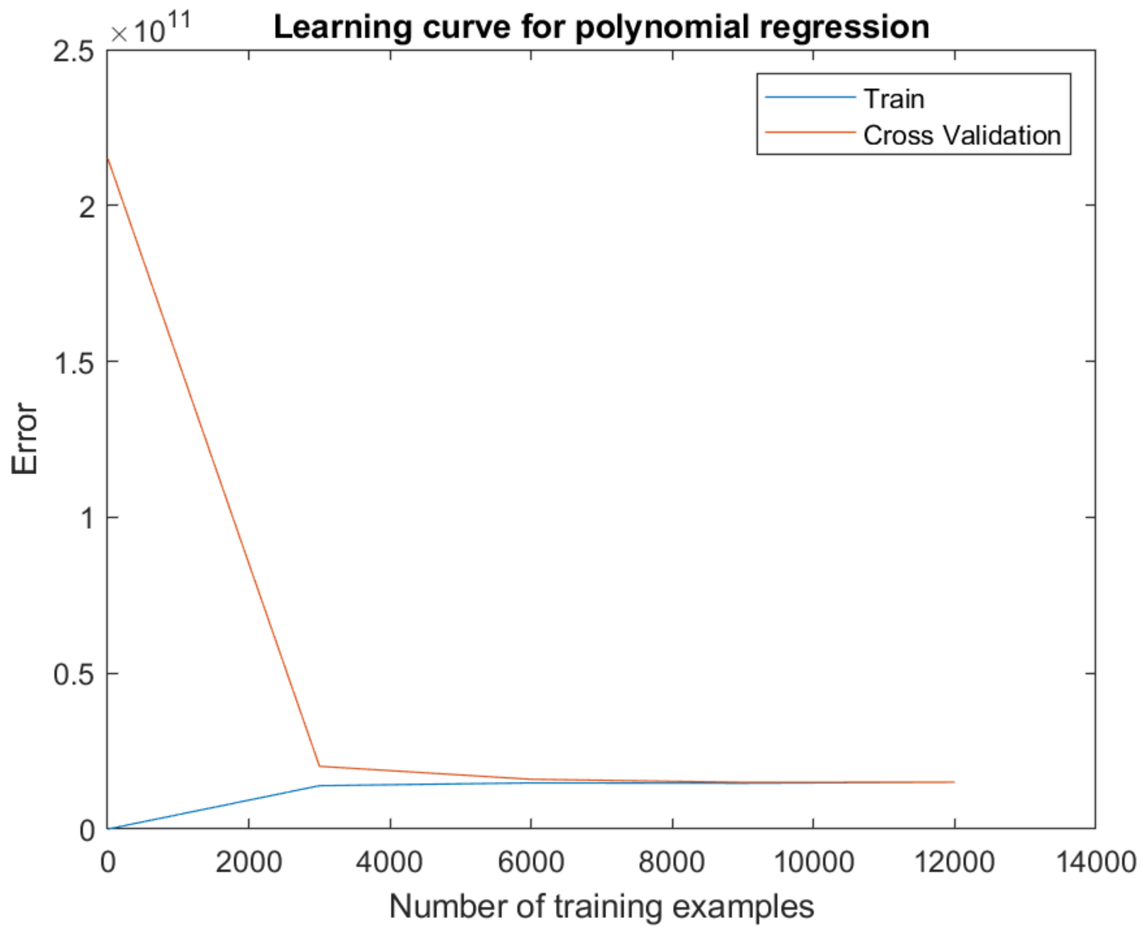
3.5 Bias/Variance

To find out whether our model is suffering from high bias(underfit) or high variance(overfit) problem, we will plot the learning curve for our model.

3.5.1 Learning Curve

The learning curve function is implemented in learningCruve.m file and it basically calculates regression parameters based on the first ith training examples, where i is an iterative variable (say 3000, 6000, 9000, 12000). The respective cross-validation error is also calculated for the obtained regression parameters for first ith training examples. The learning curve is shown in figure 3.3. For obtaining the learning curve, we increase the size of the training set while calculating the validation and training error using the obtained thetas.

Figure 3.3: Learning Curve

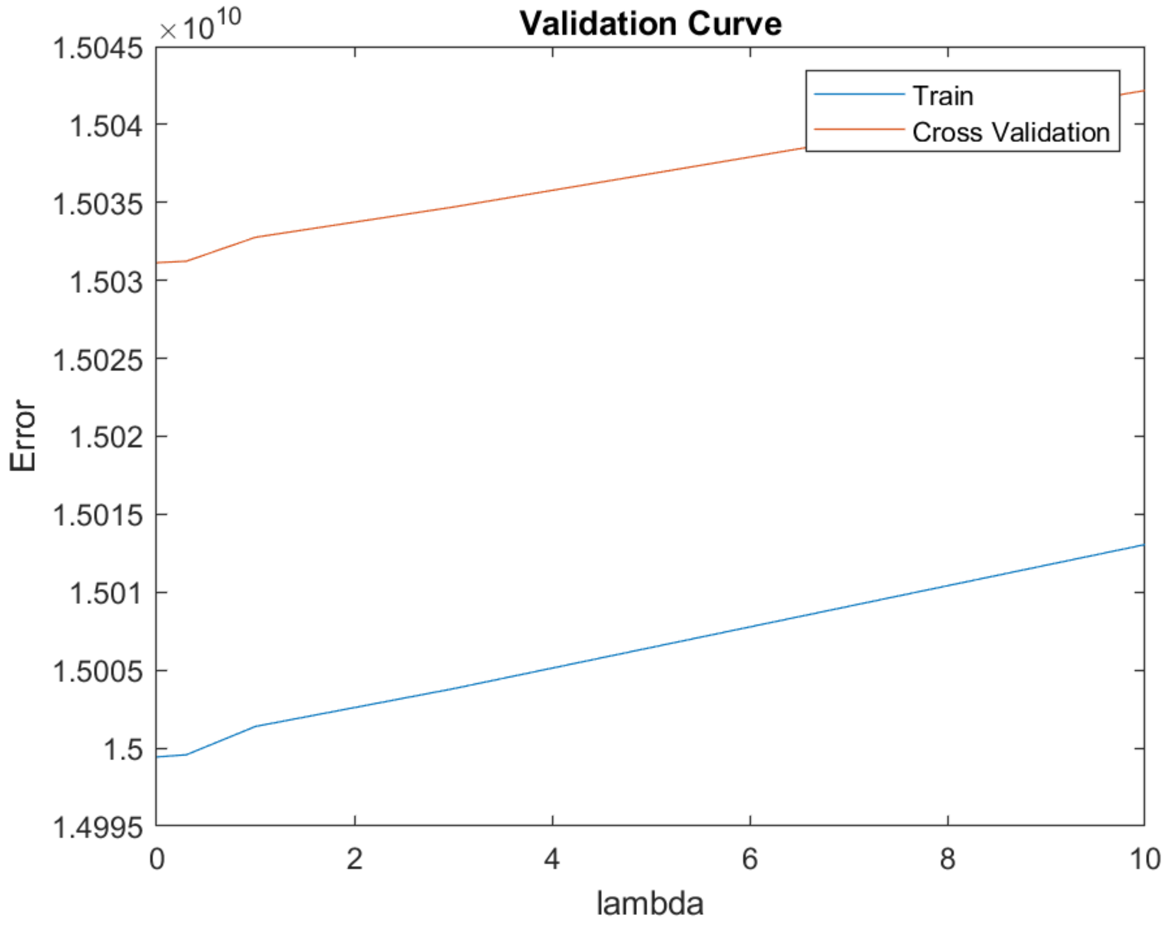


From figure 3.3, we can conclude that our model is suffering from high bias (underfitting) issue as the performance of the cross-validation and training set end up being similar. (Regularization would not help in case of high bias, but it is being implemented as the project requirement)

3.6 Validation Curve (Selecting λ using cross-validation set)

Figure 3.4 shows graph of cross-validation error and training error vs different values of λ. As we can see, increasing the λ reduces the complexity of our model, which further reduces the validation and training error. As mentioned before, the model is suffering from high bias, so regularization would not help here.

Figure 3.4: Validation Curve



3.7 Dealing with high bias

Here are a few ways to deal with the problem of underfitting or high bias:

- **Adding additional features:** Additional features such as the distance of each house from the house of highest price has already been incorporated. Domain knowledge might be required for further feature engineering.
- **Add polynomial terms:** As evident from figure 3.1, the model with 7th order polynomial gives the lowest validation error. Another solution is to increase the polynomial variables(x1,x2) in our model, but just for cubic terms, the feature grows as $O(n^3)$.
- **Decreasing λ:** As discussed in section 3.5, λ is already taken zero for our model.

3.8 Prediction

As required, the *prediction*() function is implemented in *prediction.m* file. The function takes two arguments as input (θ_n and features) and returns a vector containing predicted prices.

4. Model Evaluation Metrics

For a regression model, the following metrics can be used to determine the accuracy of the model [1]:

- **MSE (Mean Square Error):** It is basically the cost function given in equation 4.1 with $\lambda = 0$.

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{4.1}$$

- **RMSE (Root Mean Square Error):** RMSE measures the average error performed by the model in predicting the outcome for an observation. The lower the RMSE, the better the model.

$$RMSE = \sqrt{MSE} \tag{4.2}$$

- R^2 **value:** R^2 corresponds to the squared correlation between the actual outcome values and the predicted values by the model. The higher the R^2, the better the model. R^2 value range is [0, 1].

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(x_i - \hat{x_i})}{\sum_{i=1}^{m}(x_i - \bar{x})} \tag{4.3}$$

  Here, x_i is the actual(observed), $\hat{x_i}$ is predicted output and $\bar{x}$ is the mean of actual(observed) output of the model.

Both the above evaluation metrics are implemented in file *modelEval.m*. The RMSE and R^2 values for training and testing set are shown in Table 4.1

Table 4.1: Model Evaluation Metrics

Dataset	**MSE**	**RMSE**	**R-Squared**
Training	14828388231.48	121771.86	0.781840
Testing	17664794402.45	132908.97	0.728440
Validation	14877776739.51	121974.49	0.782433

Bibliography

[1] https://stats.stackexchange.com/questions/142873/how-to-determine-the-accuracy-of-regression-which-measure-should-be-used, last accessed on 18 oct, 2019.

[2] https://www.slideshare.net/pawanshivhare1/predicting-king-county-house-prices, last accessed on 15 oct, 2019.