Predicting Home Sale Prices using Linear Regression Models

Garrett Ransom

September 2018

Abstract

We describe and analyze the use of regression models to predict home sale prices in King County, USA by utilizing one of Kaggle's open source dataset composed of 19 independent variables and roughly 21.6k observations. Using simple linear regression as a baseline, we explore regularization methods to see if changes to the cost function result in a boost to model performance.

1 Introduction

The objective of this project is to use different regression methods for the prediction task of pricing home sales in King County. Although other models could be utilized for the prediction of a continuous variable and potentially deliver stronger results, regression models were chosen as a means to disect the theoretical underpinnings of linear regression and use it against some data. Below, we detail the features collected for the dataset. During data cleaning, we dropped "id", "date", "yr_built", "yr_renovated", "zipcode", "waterfront", and "view" to avoid using features during model training with less predictive power.

Column Name	Description
id	identification number for a house
date	selling date
price	dependent variable
bedrooms	number of bedrooms in the house
bathrooms	number of bathrooms in the house
sqft_living	square footage of the house
sqft_lot	square footage of the lot
floors	total floors in house
waterfront	house which has a view to a waterfront
view	has been viewed
condition	how good the condition is overall
grade	numeric grade given to the unit based on KC grading system
sqft_above	square footage of house apart from basement
sqft_basement	square footage of the basement
yr_built	built year
yr_renovated	year the house was renovated
zipcode	zip
lat	latitude coordinates
long	longtitude coordinates
sqft_living15	living room area in 2015
sqft_lot15	lot size area in 2015

2 Linear Regression Theory and Intuition

2.1 Introduction to Basic Concepts of Linear Regression

As with all machine learning problems, we have an $(N \times p)$ input matrix $X^T = (X_1, X_2, X_3, ..., X_p)$ that is used to predict a real-valued ouput Y. In the case of linear regression problems, we use our input matrix to predict a continuous output value Y. The linear regression model has the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j.$$

In this model, we consider β_0 the bias unit or the intercept, whereas the remainder of β_j's represent the weights or coefficients that correspond to the respective input feature vectors of X. The features of X can range from quantitative inputs, numeric transformations of qualitative features, basis

expansions, or combinations of variables ($X_3 = \frac{X_1}{X_2}$).
Our objective is to find a function that best approximates the linear relationship between our input and output data. Least squares is often used for this purpose, where we pick coefficients that minimize the residual sum of squares

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^{N}(y_i - f(x_i))^2 && (1) \\ &= \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2. && (2) \end{aligned}$$

We approach minimizing the residual sum of squates equation by first changing it to its vectorized form

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta).$$

By taking the partial derivative with respect to β we get

$$\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta).$$

Assuming that X is not a singular matrix, we can set our partial derivative to zero and obtain the unique solution

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Therefore, we can predict values of our output vector by using the equation

$$\hat{y} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

2.2 Geometric Representation of Linear Regression

We can view $\mathbf{X}$ as a matrix composed of p n-dimensional column vectors in $\mathbb{R}^p$ such that it creates a subspace of $\mathbb{R}^p$ called $W = \text{span}\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_p}\}$. Given the output vector $\mathbf{y}$, we can project each element in $\mathbf{y}$ onto the subspace W to get the best approximation of $\mathbf{y}$ called $\hat{\mathbf{y}}$. In this case, $\hat{\mathbf{y}}$ symbolizes the orthogonal projection of $\mathbf{y}$ onto the subspace W. The Best Approximation Theorem tells us that $||\mathbf{y} - \hat{\mathbf{y}}|| < ||\mathbf{y} - \mathbf{v}||$ for any vector $\mathbf{v}$ within W.

2.3 Multiple Linear Regression

The geometric intuition provides a nice basis for expanding our concepts to multiple linear regression. In the perfect scenario, we desire a set of features within the data matrix $\mathbf{X}$ that have no influence on eachother (equivalent to $\langle \mathbf{x_j}, \mathbf{x_k} \rangle = 0$ for j $\neq$ k). We can achieve this by making the data matrix orthogonal. In most real world environments, inputs are typically not orthogonal. Therefore, we have to carry out a process to orthogonalize the matrix. We typically use the *Gram-Schmidt* procedure for this task, which ties in nicely with **QR** decomposition where $\mathbf{X} = \mathbf{QR}$.

2.3.1 Gram-Schmidt and QR Decomposition

Algorithm *The Gram-Schmidt Process*

```
1. Let $\mathbf{v}_0 = \mathbf{x}_0 = \mathbf{1}$.
2. For $j = 1, 2, ..., p$
      Project $\mathbf{x}_j$ onto $\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{j-1}$ to produce the coefficients $\hat{c}_{ij} = \langle \mathbf{z}_i, \mathbf{x}_j \rangle / \langle \mathbf{z}_i, \mathbf{z}_i \rangle$, $i = 0, 1, ..., j-1$ and the orthogonal basis vector $\mathbf{z}_j = \mathbf{x}_j - \sum_{k=1}^{j-1} \hat{c}_{kj} \mathbf{z}_k$
3. Project $\mathbf{y}$ onto the subspace $W = \text{span}\{\mathbf{z}_0, ..., \mathbf{z}_p\}$ to get the estimate $\hat{\beta}$
```

For $k = 1, ..., n, \mathbf{x}_k$ is in $\text{span}\{\mathbf{x}_1, ...\mathbf{x}_k\} = \text{span}\{\mathbf{z}_1, ...\mathbf{z}_k\}$. This means there are constants $r_{1k}, ..., r_{kk}$ such that $\mathbf{x}_k = r_{1k}\mathbf{z}_1 + ... + r_{kk}\mathbf{z}_k + 0 \cdot \mathbf{z}_{k+1} + ... + 0 \cdot \mathbf{z}_n$ and $\mathbf{r}_k = [r_{ik}, ...r_{kk}, ...0, ..., 0]$.

So, there exists an orthogonal basis matrix $\mathbf{Q}$ composed of $\mathbf{z}$ vectors created from step 2 of the Gram-Schmidt Process multiplied by an upper triangle matrix $\mathbf{R}$ composed of vectors $\mathbf{r}_1, ..., \mathbf{r}_n$ that is equal to the input matrix $\mathbf{X}$.

$$\mathbf{X} = \mathbf{QR} \tag{3}$$

$$\hat{\beta} = \mathbf{R}^{-1}\mathbf{Q}^T\hat{\mathbf{y}} \tag{4}$$

$$\hat{\mathbf{y}} = \mathbf{Q}\mathbf{Q}^T\mathbf{y} \tag{5}$$

2.3.2 Regularization Methods

In prediction tasks, we'll often look for methods to help avoid overfitting data. High variance models have issues generalizing to new data. One way of creating models with less variance is to squash the values of its coefficients. This results in a less complex model that is more likely to perform well on new data. Two common types of methods are Ridge and Lasso Regularization, also referred to as **L2** and **L1** Regularization respectively.

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{6}$$

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{7}$$

Here, we introduce a penalization parameter λ, which squashes coefficients closer to zero as it scales larger. Notice here, that if $\lambda = 0$, then the estimates will be equivalent to the solution for linear least squares. In the case of ridge regularization, the solution will push the least important coefficients very close to zero. But these coefficients will never be exactly equal to zero. The constraint region for ridge is $\beta_1^2 + ... + \beta_j^2 \leq t$, which forms a disk; while lasso's region is $|\beta_1| + ... + |\beta_j| \leq t$ and forms a diamond. In the case of lasso regularization, the least important coefficients can get pushed to exactly 0, equivalent to performing feature selection for us.

3 Experiment Results

During experiments, we used Ridge Regression and Lasso Regression as predictor models. In order to find the best lambda parameter, we used scikit learn's grid search method. For both ridge and lasso, the best lambda parameter came to be 100. In terms of model performance, both models also predicted with a roughly 63% accuracy over the test set.

4 Conclusion

A natural extension of this experiment would be to explore different models that also serve as predictors of continuous variables. Tree-based methods, as well as neural networks can also be used for such tasks. Polynomial regression could also better approximate the data. However, with a few lines of code, we were able to produce decent results.