Predicting Real Estate Prices from Property Description

Team: Singular Victory Division
Members: Yang Cao, Cheng Jia, Lanlan Pang

Project Overview

This project focuses on the relationship between the prices of real estate properties, and the description in their online advertisements. Log-prices of 20311 properties were provided in the training data set. Features provided for each property contains 7 dummy variables indicating the city where the property is located, frequencies of 5000 most frequent tokenized unigrams (words) in the documents, and frequencies of 5000 most frequent bigrams (words) in the documents. In addition, the features of another 20307 properties were provided without the corresponding prices in the test dataset. The goal of this project is to model and predict the log-prices of a certain real estate property solely based on the given curated features.

This report contains detailed record of the evolution of the models we have devised along the way, including the analyses and comparisons of said models.

Model #1

Due to the large number of features, our first attempts to run native functions in Matlab toolboxes, such as stepwise regression, OLS models, and LASSO, failed without dimensionality reduction. Recalling an R package that a team member (and statistician in training) has come across in the past, we attempted glmnet (http://cran.r-project.org/web/packages/glmnet/index.html), an extremely efficient algorithm that takes advantage of the X matrix written by Tibshirani et al. Fortunately glmnet has been ported to Matlab by Jiang and Qian, and that's what we used to fit our model #1.

Using 10-fold cross-validation, we were able to determine the optimal λ=0.005 for the LASSO penalty, as is shown in Figure-1. The RMSE for this method is 0.7434, beating the first baseline.

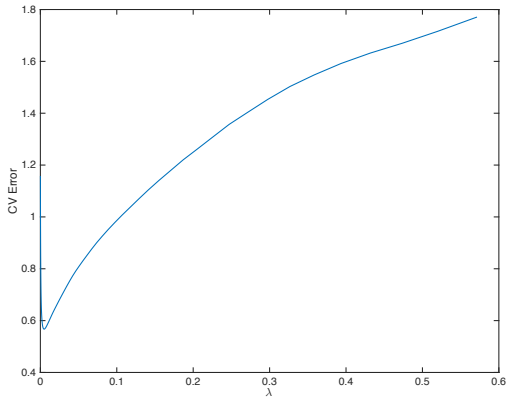


Figure-1 Cross-validation errors for all λs in glmnet training

Model #2

In order to refine our model, we attempted glmnet followed by Naïve Bayes classification. The idea behind this model is that different properties in different ranges of models may have different structures in their feature set, by classifying the properties into different categories of prices first, we may be able to fine tune our model to achieve a better RMSE.

We binned our Y (log-prices) into 15 classes, trained a NB model to predict the class label from X, and fitted one LASSO linear regression model for each class (15 models in total). When predicting, we would first use the trained NB model to predict the class label, and then pick the corresponding linear regression model for final prediction. As it turned out, this method actually increased the RMSE dramatically to 1.0199. The reason this failed to work is that a handful of classes had very little data, hence the trained model for that specific class is highly inaccurate. We did not include the element of NB in our final model.

Model #3

Another direction we were considering to refine our glmnet model is to perform a semi-supervised regression with SVD and glmnet. We used SVD on the pooled dataset containing both training and test Xs, calculated the V matrix, and project the training Xs on to V.

$$U_{n\times p}S_{p\times p}V^{T}_{p\times m}=\begin{pmatrix}X_{train}\\X_{test}\end{pmatrix}_{n\times m}$$

And we regress the training dataset $X_{n\times m}V_{m\times p}$ onto the training Y. We tried several different p in order to get the lowest CV error, as in Figure-2.

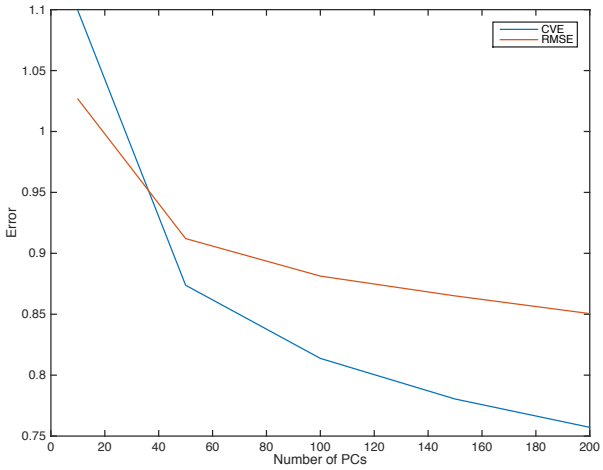


Figure-2 Cross-validation errors and RMSE for different number of PCs in SVD-LASSO.

From the above figure, it was apparent that this method did not further increase the accuracy of our results because the benefits gained from learning the structure of X using the test dataset were cancelled out by the loss of information during dimensionality reduction. For this method, with a maximum of 200 principle components, we were able to achieve an RMSE of 0.8505.

Model #4

After two failed attempts to fine-tune our glmnet model, it dawned on us that the potential of the regularized linear regression model has been fully exploited, since we had obtained a well sparse solution and fairly good result, with the use of all of the features and without dimension reduction from glmnet.
Inspired by the boosting method, we decided to build a new model to fit the residuals of the glmnet model. This new model cannot be in the family of linear regression because otherwise glmnet would've already discovered it through CV.

We turned to an instance-based method, KNN. Subsequently, two issues arose with the frequency data provided. The first issue is that the default Gaussian metric cannot directly reflect the true distance of the words. The second is that high dimension data like ours (p=10007) is not computationally feasible with KNN.

In order to solve the first problem, we decided to see what words appear the most in the given dataset. In the word cloud below, the most frequent words are “room”, “windows”, “city”, “park”, none of which are particularly informative of the actual price of the property. So we decided to try inverse document frequency (IDF) weighting on our features before using KNN.

Figure-3 Word frequencies of a subset of the data.

The IDF was calculated by

$$IDF_i = -\log\left(\frac{N_w^{(i)}}{N_{total}}\right), \quad i = 1, 2, \ldots, M$$

where $N_w^{(i)}$ was the times that the i[th] feature appeared in the N_{total} samples, and M was the total number of features. Note that inspired by the semi-supervised methods, IDF was calculated from both the training and the testing X.

Using the IDF-weighted features from both training and testing data sets, we performed SVD to reduce the dimensionality to 30, a number determined by our former experience with SVD-LASSO as well as the computational limitations of our equipment. Subsequently, we used KNN to predict the glmnet prediction residuals. Eventually, these residuals were added back to the predictions by glmnet, and produce our final predictions. The performance using K-nearest neighbor to predict residual and compensate the prediction of glmnet, with or without IDF weighting along the variation of K, was plotted in Figure-4.

It became apparent that combining the IDF weighting and KNN method to predict the residuals performed better than the same method without IDF weighting. The reason behind the reweighting is that reweighted features better reflect the amount of information each word contains in predicting the prices of the property. And K=19

was chosen for the KNN method from the cross-validation.

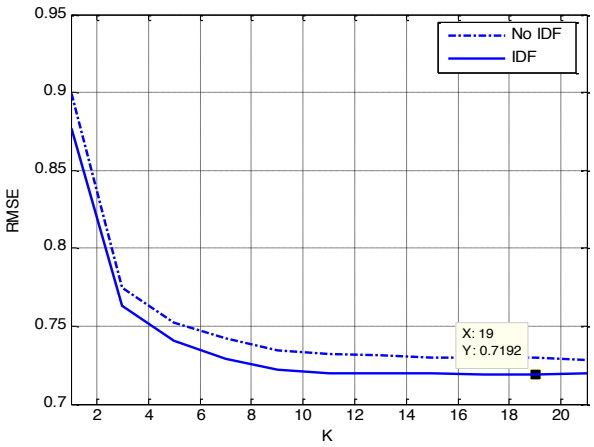


Figure-4 Test RMSE comparison, with and without IDF weighting

Note: this figure was obtained by comparing the predictions from our model to the given price_test data, because during the competition we used 10-fold Cross Validation to choose K, it is highly time-consuming and the original data was unfortunately lost.

The final model

Two more additions to Model #4 made our final model the third-best performing model amongst all competitors. The first is kernel KNN. While KNN was quite impressive by itself, by using a kenerlized version, we were able to control more parameters of this method, thus achieving more flexibility and potentially better accuracy. We selected 2 parameters associated with the kernel and SVD. With the K=19 fixed, a 15-fold cross-validation was performed to choose the best kernel width σ and the number of principle components in SVD p. Figure-5 shows the optimization surface, and the best results we have achieved were from the combination of σ=6, p=19, with the lowest CV RMSE = 0.7181.

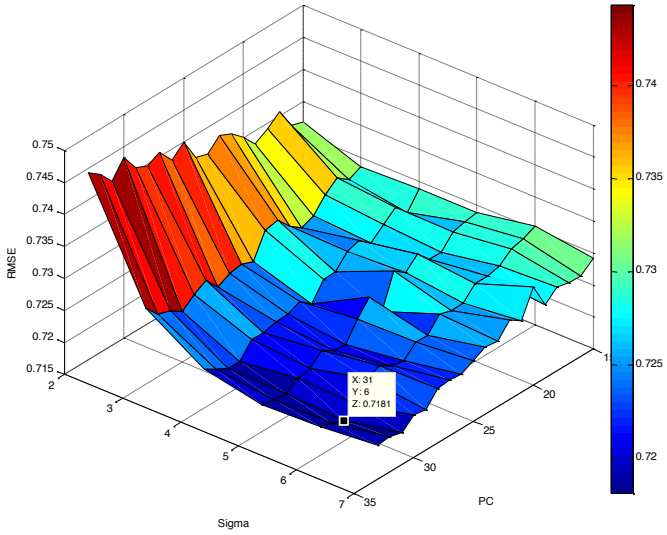
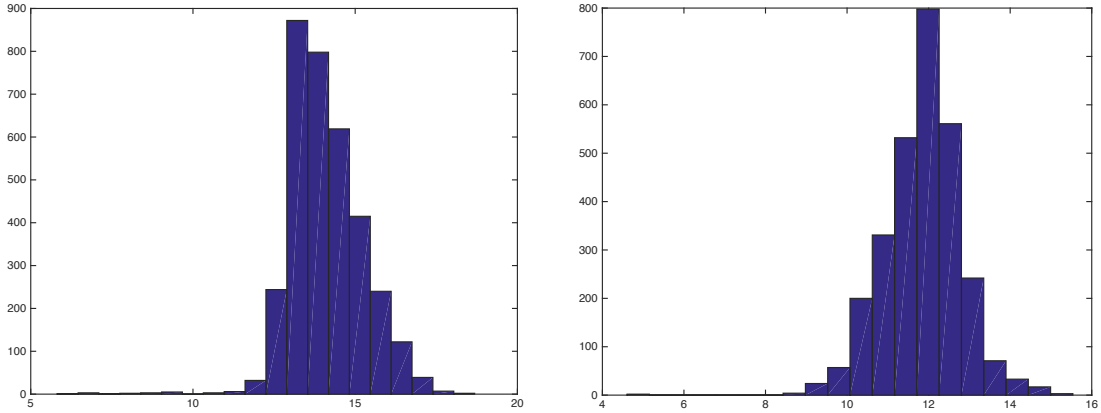


Figure-5 Parameters Selection by 15-fold Cross-Validation.

The second addition to our Model #4 is to refit a separate model for each city. This was inspired by the following observation in Figure-6, which illustrated the price distribution of the listed properties in New York and

Philadelphia.



New York Philadelphia

Figure-6 Log-Prices of real estate properties in New York and Philadelphia

The mean (sd) of the log-prices of the properties in New York is 14.0950 (1.1257), while in Philadelphia the mean(sd) is 11.8561 (0.9342). What's more interesting is that the two distributions assumed slightly different shapes. This suggested that by separating the cities and fit a model for each city with our model, we might be able to take advantage of the subtle structural differences between cities. However, when we trained each city separately, the resulting RMSE became larger, which we reasoned is due to the limited number of data points in some cities. For example, there were only 402 properties for Boston. We again used a method inspired by stacking methods of ensemble learning, and averaged the predictions from the model trained on all the data, and the model trained on each separate city, which produced much better results.

The methods and their RMSEs were summarized in the following table.

Method	RMSE
SVD+glmnet	0.8505
NB+glmnet	1.0199
Glmnet	0.7434
glmnet + residual adjustments (SVD+KNN)	0.7291
glmnet + residual adjustments (IDF+SVD+KNN)	0.7192
glmnet + residual adjustments (IDF+SVD+kernel KNN)	0.7163
(City + Total)/2	0.6767

Table-1 Selected methods and their RMSEs.