Empirically Verifying and Innovations on Several Machine Learning Theorems and Techniques

Yisheng Jiang, CS 229 Tel: 650-804-6836 E-mail: yishengjiang@yahoo.com

Abstract

In this project I will perform several machine learning techniques on a training set of 500 samples and 13 features, along with 500 of their perspective labels. The techniques I use include: stochastic gradient descent for linear regression, closed form linear regression, forward-search feature selection, using high-order variables in linear regression, k-fold cross-validation, training error vs. generalization error, and lastly, an innovative combination of logistic regression and multi-tier cross-referencing that allows us to predict continuous variables with binary-classification methods.

Introduction

In CS 229 we learnt a massive array of tools to operate on data. This paper is motivated by the desire to put these tools to work and possibly, using trials and error, to improve upon these tools.

Stochastic gradient descend for linear regression

I implemented the stochastic gradient descend for linear regression as follows: starting with a zero vector of coefficient, I look for the direction of steepest descend in adjusting that vector, as expressed in the cost function. I then scale that vector with an learning rate.[i] Most learning rates I tried caused the prediction to diverge with each iteration. Attempts were made to "scale" the size of the adjustment: normalizing the vector to keep the direction of the adjustment the same, but the magnitude smaller than 5% of the original vector. It did not work, the error gap was still diverging.

Then I tried something I learned on the PCA lecture notes: normalizing the X matrix so they are expressed with zero mean and in terms of their standard deviations from the mean[ii]. This confined the X values from -1 to 1, and it allowed me to get much better empirical results.

First I used alpha =0.0004. The error margin improved for the first 30 iterations (figure 1), but got worse after that.

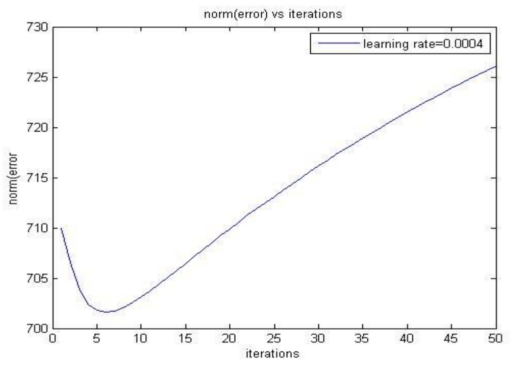

Figure 1 Error margin at alpha=0.0004

Similarly, learning rate at 0.0001 also diverged and gave terrible predictions (Figure 2):

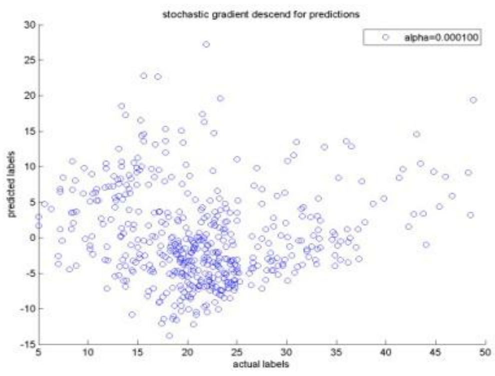

Figure 2 Prediction at alpha=0.0001

Then I used alpha = 0.00009. The error margin improved for the first 42 iterations, but got worse after that (figure 3).

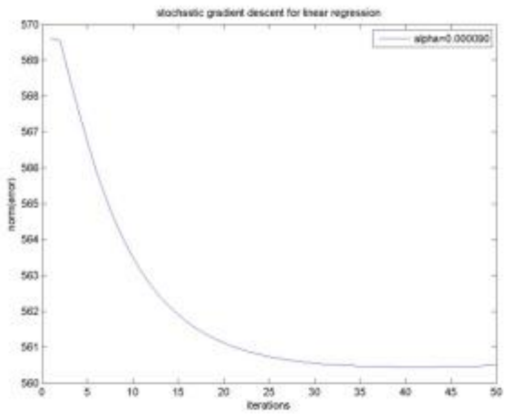

Figure 3 Error margin at alpha=0.00009.

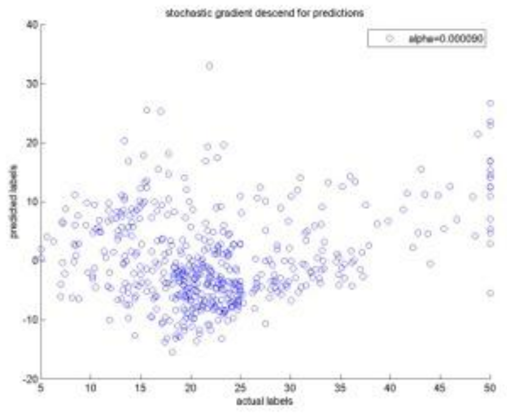

Figure 4 : Prediction at 0.0005 learning rate after 35 iterations before it diverged.

Having tried a few other learning rate values, I was able to find one that does not cause the prediction to violently diverge, using learning rate of 0.00004, and iterating 10000 times, I was able to make a somewhat better prediction of the data:

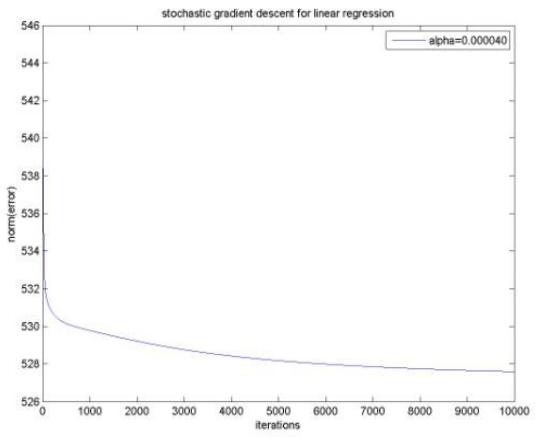

Figure 5   A learning rate, 0.00004 that worked

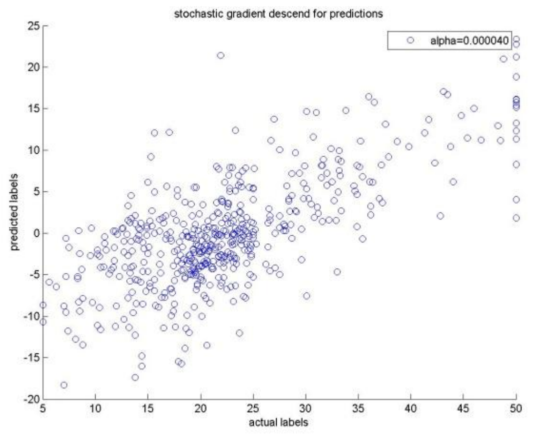

Figure 6   A not-so-great prediction after 10000 iterations

As you can see here the error gap dropped sharply in the first few iterations, and they continue to drop slowly up to 10000 iterations. Our conclusion regarding the stochastic gradient descend is that for our set of data, it is hard to a small enough learning rate that the prediction does not diverge, and once you find a small enough learning rate, it takes many iterations to find a good prediction.

Linear Regression Closed form

The linear regression in closed form is implemented as theta = (x'*x)\x'*y[iii]. The graph of the predicted labels vs. actual labels is shown in Figure 7a.

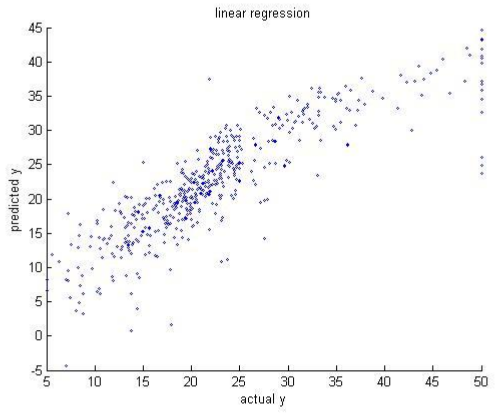

Figure 7a: closed form linear regression

Using the error-measurement metric of norm((actual-predicted)./actual), we find that we have a score of 5.5694. This will be the starting point of improvement when we do feature selection.

Feature Selection

The previous section's linear regression uses only first-order variables. What if we combine that with second, third, and fourth order feature variables and perform linear regression on that? Concretely, we would have the sample matrix of [x, x.^2, x.^3, x.^4] for a total of 52 features, then we use forward search feature selection[iv]: starting with a zero feature vector, making predictions with an extra feature and picking the most relevant one, repeating until we have n-top most relevant features. We will test if that improves our prediction.

Having implemented the feature-selection algorithm, we find that the 8 most relevant features are x6^3, x12^2, x13, x14, x6, x11, x13^2, x8. Three of those are in higher order dimensions. Using the same prediction error metric as the last section, we have a score of 5.366, an immediate improvement over the raw linear regression with first order terms.

Using this method to choose n-top most relevant features, the table 1 shows the training errors we found by varying the parameter of n-top.

Number of top features	Training error
8	5.3661
9	5.1681
12	4.5590
14	4.5588
18	4.2139
25	4.1670
27	4.1933
28	4.2071
30	4.1974
35	4.2360

Table 1 Average error as a function of num of top features

We see that our method of feature selection is robust enough that selection of up to 35 top features will get better results than the raw first-order linear regression. Our prediction got much better, as can be seen from Figure 7b.

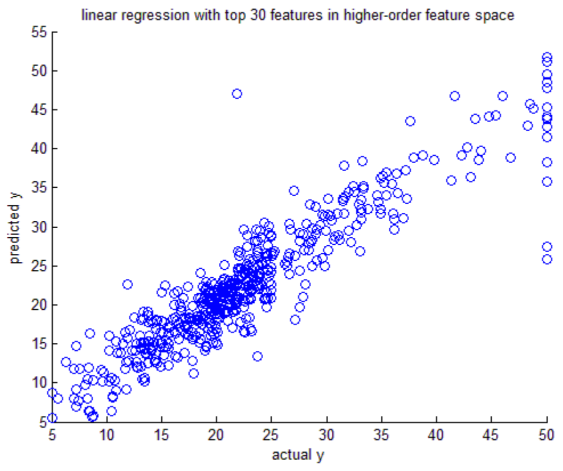

Figure 7b Closed form linear regression with top 30 features

Training Sample Size vs. Generalization Error

I now attempt to find the relationship between training sample size and the generalization error by using the k-fold cross-validation mechanism, which randomly splits our training set into k randomly permutated subsets. Looping through these sets, using one set for generalization and other sets for training, we average the generalization errors with trained parameters in each permutated subsets[v]. By controlling the number of sets, we also control number of training examples.

We know that training error will decrease as the sample size get larger[vi], but what about the generalization error? Figure 8 shows the empirical relationship of the mean generalization error vs. the training sample size.

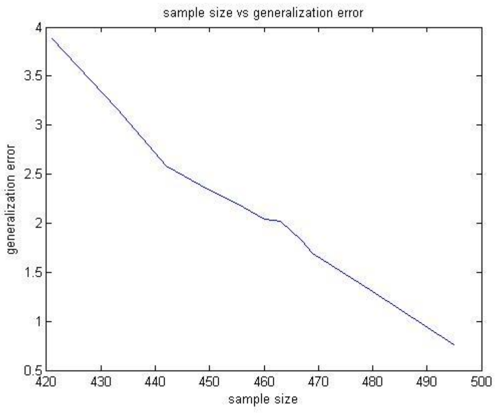

Figure 8 Generalization error average vs sample size

I was also told in the lecture notes that the variance in generalization error will increase as the training sample increase[vii]. Variance in generalization error was obtained by randomly permutating the sample size in 20 different trials, which produces a set of 20 generalization errors in each iteration. I understand that is more of an upper-bound theorem and will not always be the case, as I saw empirically (Figure 9):

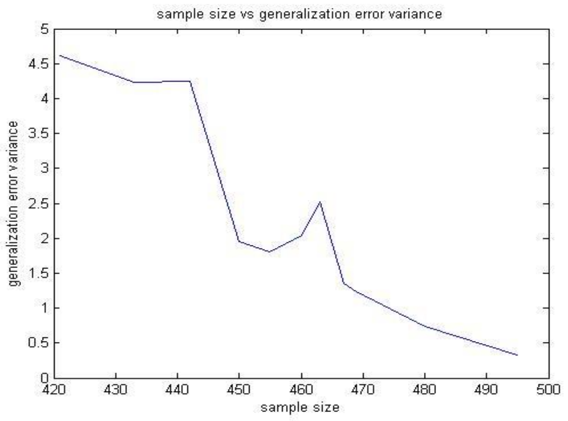

Figure 9 Generalization error variance

Multi-tier Cross-Referencing Logistic Regression

Even though our data is not binary, we can still make a series of binary predictions on the data using logistic regression, and then cross-reference these predictions to group these data sets into several different intervals. I call this multi-tier cross-referencing logistic regression. Concretely, if make predictions on whether y<10, y<20, y<30 and so on, and cross reference these predictions, we can group the labels into several buckets. And if the number of buckets is large enough, we can simulate continuous value predictions with binary classifications.

The actual predictions employ Newton's Method for logistic regression[viii]. The result is surprisingly good, below is the graph of actual vs predicted-label with several different number of buckets (Figure 10):

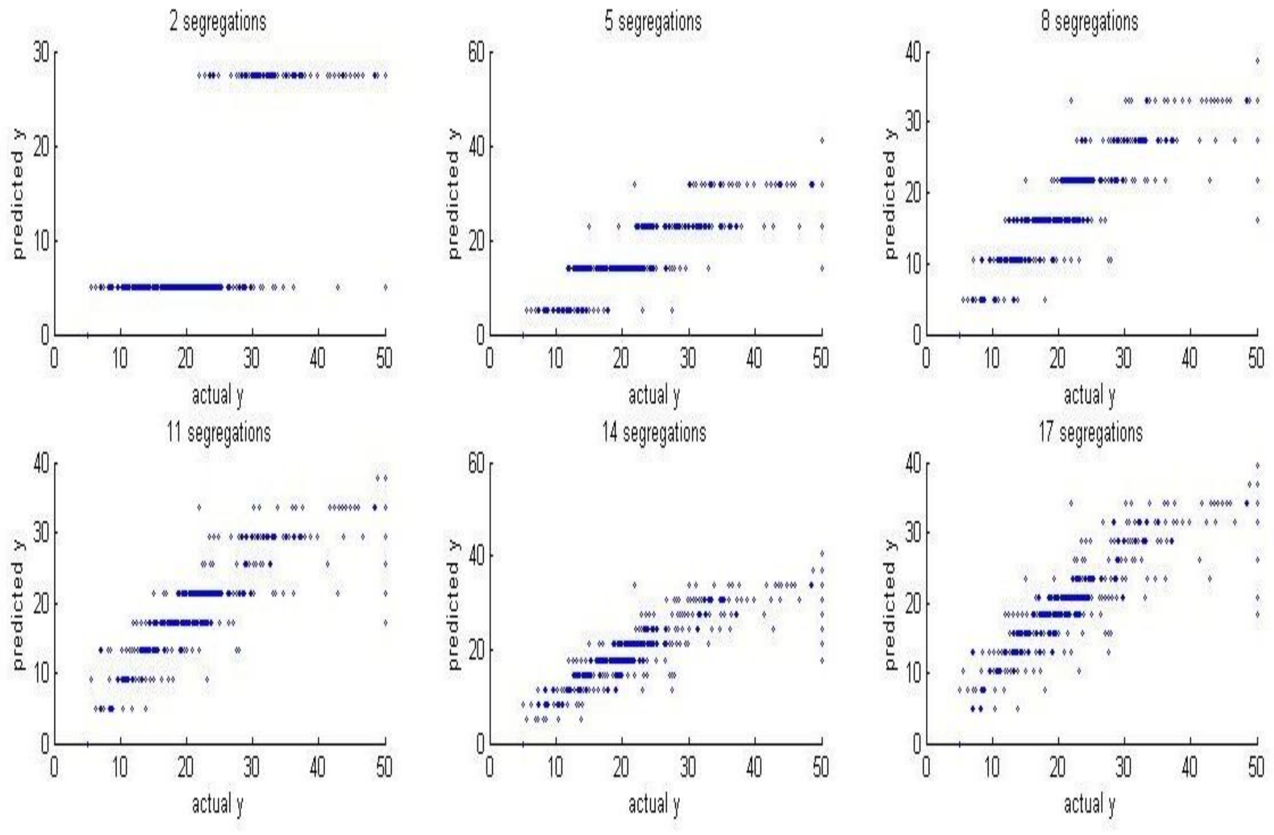

Figure 10 Multi-tier cross verification.

Conclusion

We find that several theorems and techniques taught in CS 229 are empirically sound. Stochastic gradient descend is volatile; closed-formed linear regression is both faster and more reliable; Higher-order feature variables can be relevant, and can be found using forward-search feature selection; k-fold cross-validation is a good way to verify generalization errors and its relationship to sample size; and that with Multi-tier Cross-Referencing Logistic Regression, we can model continuous variable labels with a series of binary classifications.

Acknowledgement

I like to thank Professor Ng and the TAs: Richard Socher, Marius Iordan, David Kamm, Abhik Lahiri, Andrew Maas, Peter Pham and Tao Wang who are all brilliant and helpful.

References

[i] Andrew Ng, CS 229 Lecture Notes: Supervised Learning
[ii] Andrew Ng, CS 229 Lecture Notes: Primary Component Analysis
[iii] Andrew Ng, CS 229 Lecture Notes: Supervised Learning
[iv] Andrew Ng, CS 229 Lecture Notes: Regularization and Model Selection
[v] Andrew Ng, CS 229 Lecture Notes: Regularization and Model Selection
[vi] Andrew Ng, CS 229 Lecture Notes: Learning Theory
[vii] Andrew Ng, CS 229 Lecture Notes: Learning Theory
[viii] Andrew Ng, CS 229 Lecture Notes: Supervised Learning