MC3-Project-1 - Evaluate Learning Algorithms

Magahet Mendiola

November, 2015

Overview

We will evaluate the use of linear regression and KNN learning algorithms on various datasets. We will also explore the effect of various algorithm parameters on performance and overfitting.

Custom Datasets

Best Cases for Linear Regression

The following is a simple algorithm for generating a dataset that performs better with linear regression than KNN:

```
    for x in xrange(1000):
        point = ([x, x], 1000 * x)
```

The points follow a straight diagonal line with a high slope. A small about of Gaussian noise is added to each point to keep things interesting. Also, the data is randomized before being split between the training and test sets. The reason this works well with linear regression is that it exactly follows a linear model. The learner will find precise parameters for the equation:

$$y = mx + b$$

The learner will then be able to interpolate any point along this line with high accuracy.

KNN, on the other hand, does not attempt to learn an underlaying model. It simply finds the average value of the closest training points. If those points lay to one side of the test point, KNN will not be able to extrapolate from a model. Rather, it will return the average of the closer points.

Here is an example with 1D x:

```
training points = [0, 1, 2]
test points = [3, 4, 5]
```

In this case, KNN will return the average of k training points when queried with the test points. In every case, the estimate will be off from the actual value. In the case of k = 3 and using the data generating model from above, we get the following result:

```
actual, estimate
3000, 1000
4000, 1000
4000, 1000
```

On randomly distributed data, this limitation of KNN presents itself as under and over estimates of any points that need to be interpolated. As we see in Figure 1, which shows a zoomed plot of testing data against both KNN and linear regression. The resulting RMSE is 28 and 1080 for linear regression and KNN respectively.

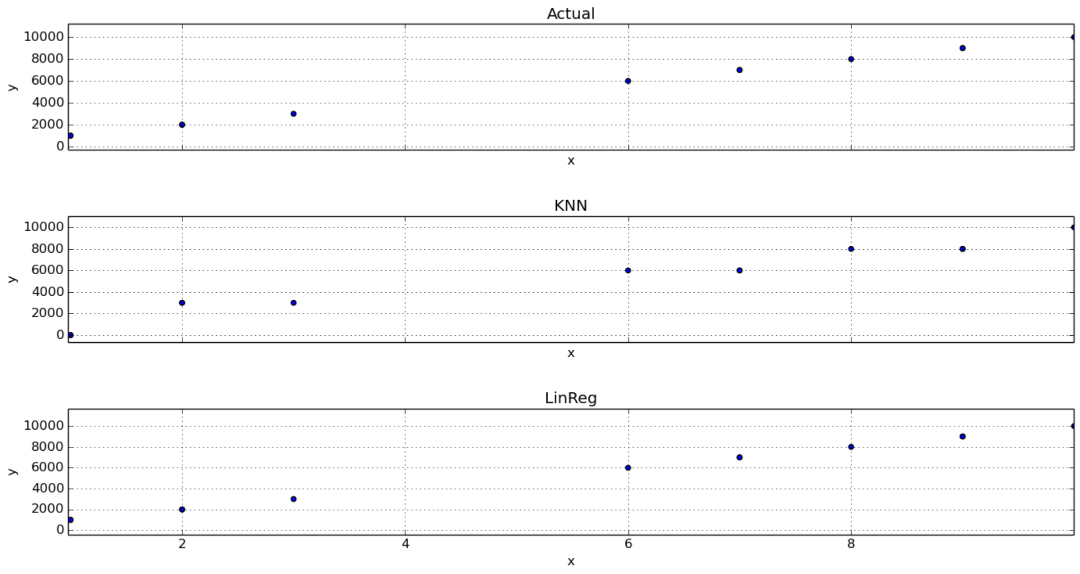

Figure 1: Linear Regression Suited Data

Best for KNN

The following is a simple algorithm for generating a dataset that performs better with KNN than linear regression:

```
for x in x_list:
    print '{},{},{}'.format(x, x, add_noise(x % 200))
```

It generates a non-linear repeating pattern that can be seen in Figure 2. As before, data is randomly distributed between the test and training sets. The subplots each show Y values from actual test data, KNN estimates, and linear regression estimates.

KNN is able to adapt to the steps from high Y values to low. Some queries will have issues when the nearest neighbor includes both high and low points. However, compared to the overall performance of linear regression, this accounts for a minor increase in error.

Linear regression performs poorly on this dataset, as it cannot cope with the non-linearity. Queries against the linear regression learner will estimate points along the regression line, which for most of the feature space will be far from actual Y values.

The following shows the error results for each learner:

KNN results

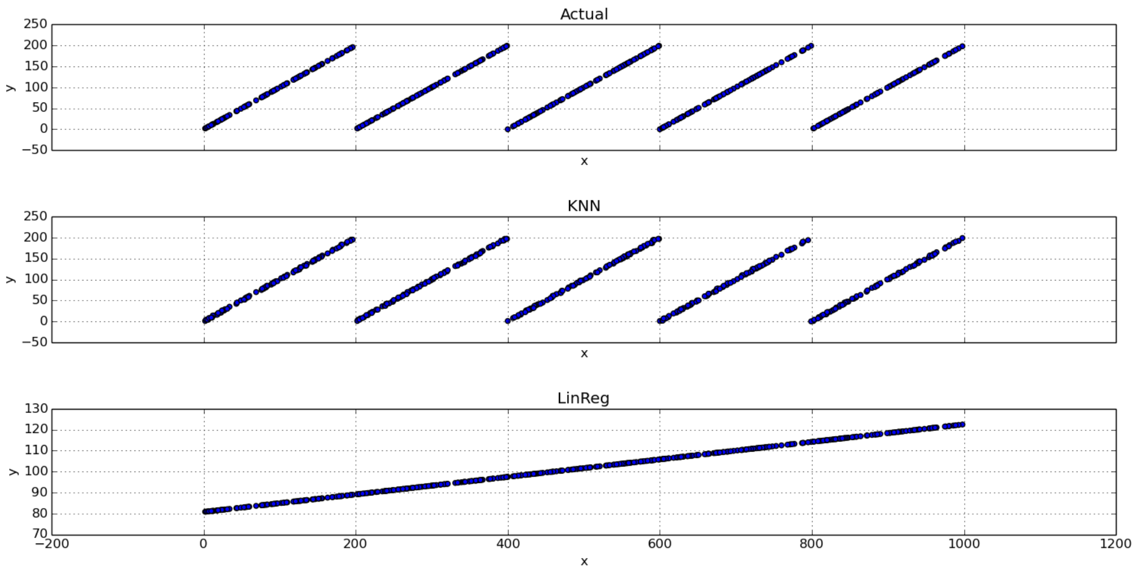

Figure 2: KNN Suited Data

```
In sample results
RMSE:  7.19196543288
corr:  0.992129724033

Out of sample results
RMSE:  11.0406799595
corr:  0.981941492455
```

LinReg results

```
In sample results
RMSE:  55.9885148725
corr:  0.212757185305

Out of sample results
RMSE:  57.5249782041
corr:  0.177523761057
```

KNN Evaluation

Effect of K on Overfitting

Taking the largest value of K, which is K = number of samples, our learner would return a constant (the mean of all points) for each query. As we decrease K, our queries on test and training data moves away from this singular value to more localized means. This allows the learner to fit the data better. On the other extreme, K = 1, the learner perfectly fits the training data. However, it loses it's ability to interpolate points between those observed in training data. Test data will, therefore, only return correct values if the points happen to land directly on a previously observed point in the training set.

Figure 3 shows that overfitting begins to occur at K = 2. Before this point, error on the test dataset decreases with smaller values of K. At K = 2, the error on testing data increases while it continues to decrease for the training set.

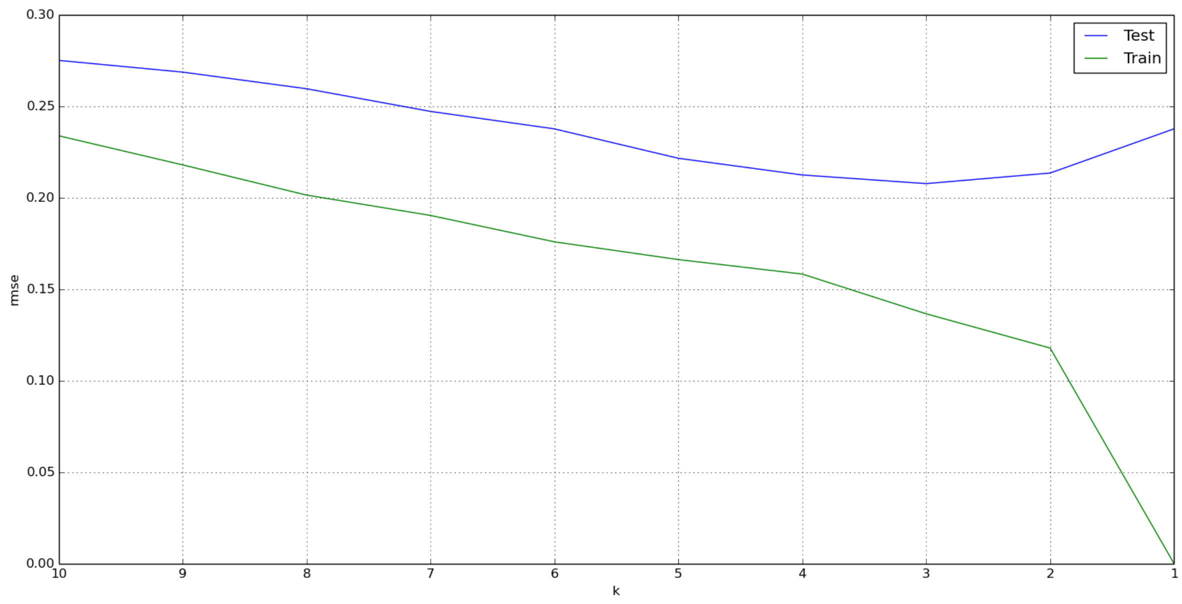

Figure 3: Test and Train error for various K

Effect of Bag Count on Performance

First, we notice only minor improvement in accuracy when using bagging with our KNN learner (k = 3, bags = 20):

```
In sample results
RMSE:  0.136590187312
corr:  0.981360326901

Out of sample results
RMSE:  0.207762150054
corr:  0.955537498166


In sample results
RMSE:  0.128889773001
corr:  0.984803655202

Out of sample results
RMSE:  0.200259933825
corr:  0.961287005578
```

However, we will see in later examples that bagging adds significantly to reducing testing error in overfit learners (k = 1).

Runtime performance of the bagging learner increases linearly with the number of bags. We see this in Figure 4. This result is expected, as each KNN learner requires an equal amount of time to train and query.

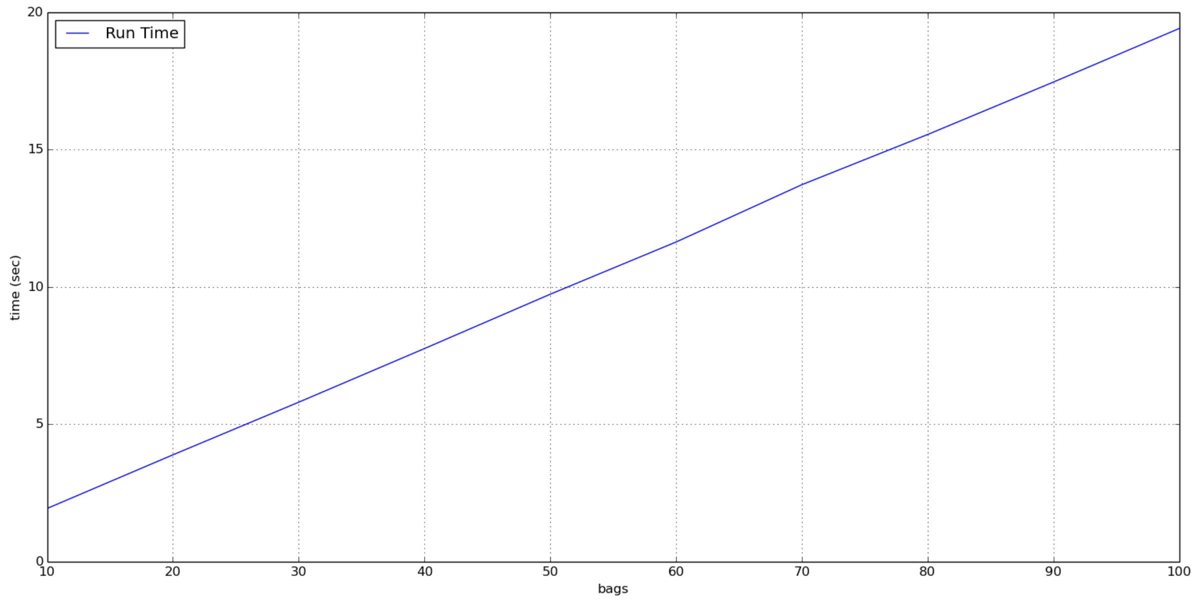

Figure 4: Runtime Performance for Various # of Bags

Effect of Bag Count on Overfitting

The number of bags does not not seem to contribute to overfitting. Figure 5 shows that error decreases identically for both the training and testing datasets as the number of bags increase. We also see that error does not drop significantly after a reaching 20-30 bags.

Effect of Bagging with respect to K values and overfitting

Figure 6 shows the error for various number of bags with a KNN learner set with K = 1. As we observed earlier, that value of K causes the worst overfitting. Training error will be zero, and testing error will be higher due to the loss of generalization. Bagging averages various learners' results, thus adding generalization. The effect is that bagging does, indeed, reduce overfitting.

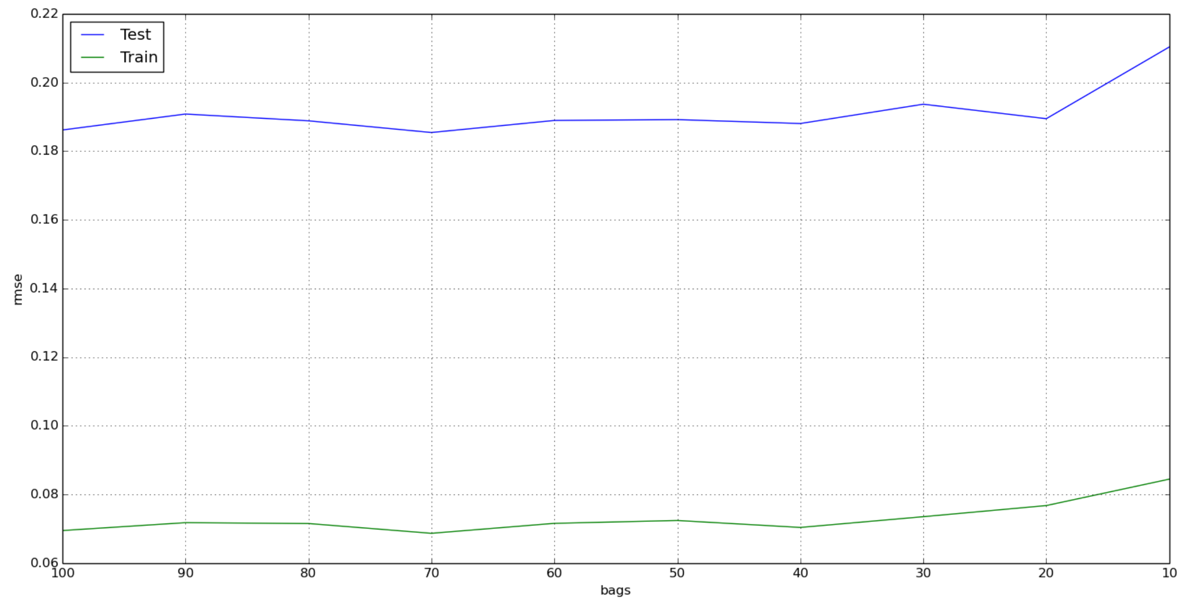

Figure 5: Test and Train Error for Various # of Bags

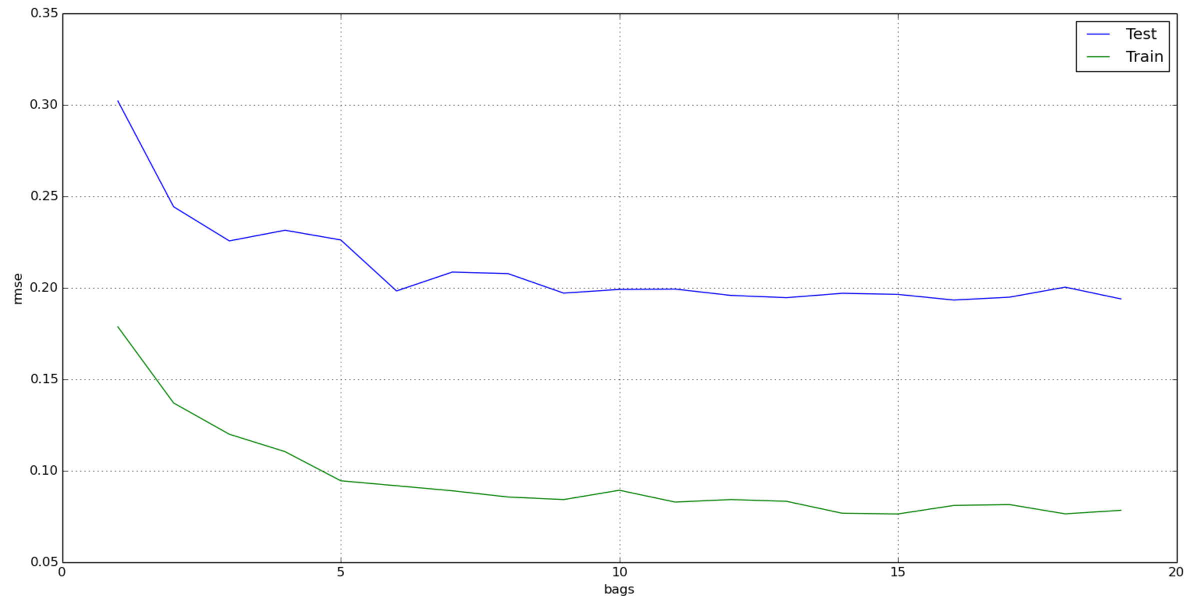

Figure 6: Test and Train Error for Various # of Bags with K = 1