CS 7646: MC3-Project 1

Chenzi Wang
cwang493

Dataset: best4linreg

The algorithm, best4linreg.py, produces a data set on which linear regression is able to come up with more accurate results that using k-nearest neighbors for regression. The data set is made up of two x dimensions (X1 and X2) and one y (Y) dimension. The output variable to be predicted, Y, is a linear function of X1 (Y = 5*X1), see Figure 2a. Y is given a small degree of random variation to simulate real world data. The values of X1 were divided into four different distributions, each spaced differently. X2 is a power function of X1 (X2 = X1^3) but this relation is hidden under a large degree random variation that increases linearly with increasing values of itself (X2 + random integer * X2), see Figure 2b. The data points were put into a random order before being export to a data file so that the data points chosen for training and testing are randomly distributed. Linear regression is able to produce superior results because it is able to approximate the linear relationship between Y and X1 without being heavily influenced by the non-linear random data points in X2. K-nearest neighbors does not perform as well because it does not recognize that Y has a strong relationship with X1 and not X2. Instead, its model depends on both X1 and X2 equally to approximate the data. The effects of the curved nature of X2 as a power function of X1, the large amount of random variation in X2, and the smaller amount of random variation in Y combine such that the nearest neighbors identified for a data point do not give a good approximation (they are not close to the actual value). The RMSE and statistical correlation values of linear regression and k-nearest neighbors on the data set can be seen in Figure 4. The closeness of each to model the data can be seen by analyzing the correlation; the closer the correlation is to 1 or -1, the better the approximation. The results show that linear regression provides a closer approximation, as expected from the reasoning given.

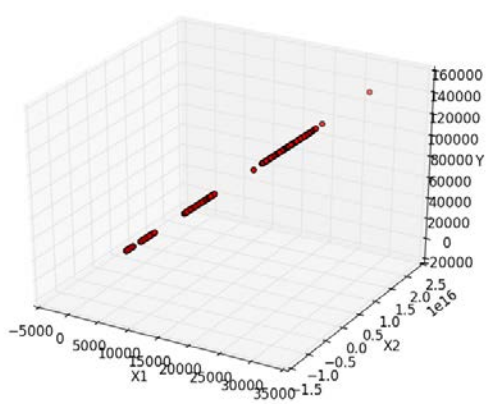
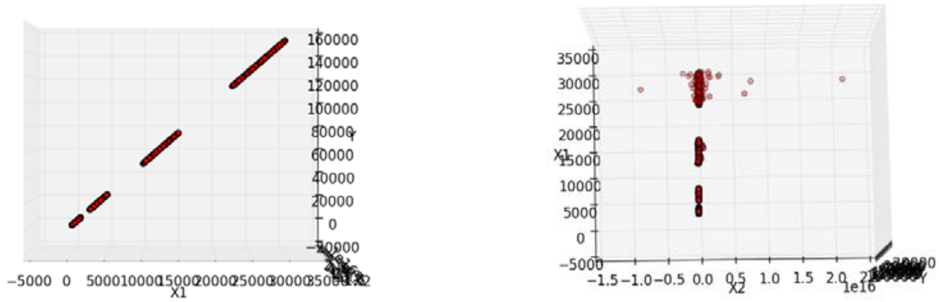


Figure 1: 3D plot of data set, best4linreg, that works better with linear regression than with KNN.



(a) (b)

Figure 2: 3D plots of best4linreg data set seen from (a) X1-Y and (b) X1-X2 viewpoints. Note the linear relationship between X1 and Y. X2 is a power function of X1, but the scale is sized to account for the data points where the value in the X2 variable has been given a random value. Y is not necessarily a function of X2.

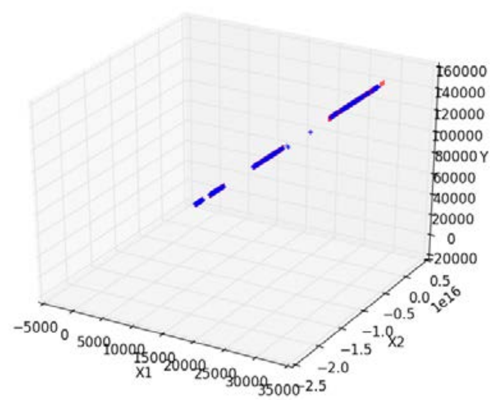


Figure 3: 3D plot showing the train (red) and test (blue) data used in the linear regression and KNN from the best4linreg data set. It is difficult to the red training data points in this figure, but they are equally distributed due to randomizing the order of the generated data prior to it being output to a data file.

(a)

```
In sample results
RMSE:  83.2505192923
corr:  0.999998614007

Out of sample results
RMSE:  79.0426498662
corr:  0.999998783173
```

(b)

```
In sample results
RMSE:  11241.8949155
corr:  0.974417228414

Out of sample results
RMSE:  16956.7720725
corr:  0.941736296179
```

Figure 4: Accuracy of results for the best4linreg data set using (a) linear regression and (b) k-nearest neighbors (k=3).

Dataset: best4KNN

The best4KNN data is also made up of two x dimensions (X1 and X2) and one y (Y) dimension. The data set follows one linear function through half the range of X1 before changing to another linear relationship. In the first linear relationship, Y is a function of X1 (Y = X1). In the second linear relationship, Y is a function of X2 (Y = -X2). When using linear regression to try to approximate this data set, it tries to approximate it as one linear function; however, this is clearly far from the truth so it has poor correlation to the real data, as can be seen in Figure 7. K-nearest neighbors, on the other hand, does not attempt to approximate the data with a function across the whole data set. Instead, it makes its predictions according to its nearest neighbors according to Euclidean distance. Consequently, the disparity in the two linear functions in the data set do not have a strong negative effect on the predictive capability of k-nearest neighbors. It is shown in Figure 7 that KNN does a superior job in approximating the data according to its correlation.

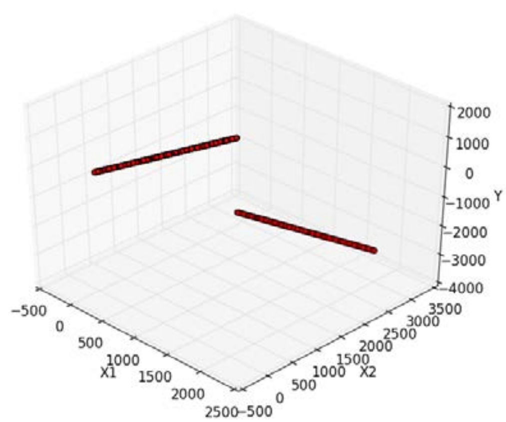


Figure 5: 3D plot of data set, best4KNN, that works better with k-nearest neighbors than with linear regression. Note that the data set is made up of two distinctly different linear functions. Linear regression attempts to approximate data as one linear function, consequently producing innacurate results by approximating a function somewhere between the two; whereas, k-nearest neighbors does not attempt to approximate the data as a function but looks for the nearest data points for its model.

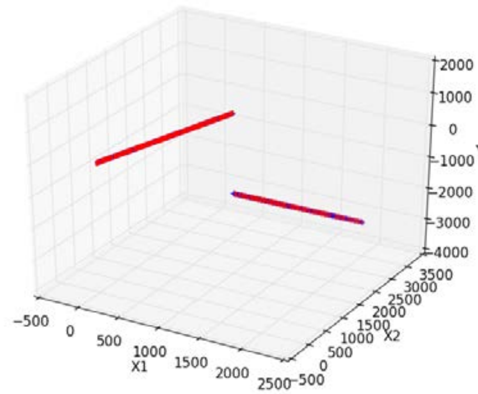


Figure 6: 3D plot showing the train (red) and test (blue) data used in the linear regression and KNN from the best4KNN data set. It is difficult to the red training data points in this figure, but they are equally distributed due to randomizing the order of the generated data prior to it being output to a data file.

(a)
```
In sample results
RMSE:  731.585670056
corr:  0.859340313552

Out of sample results
RMSE:  708.03496215
corr:  0.865977518177
```

(b)
```
In sample results
RMSE:  1.88586359037
corr:  0.999999131681

Out of sample results
RMSE:  83.500385894
corr:  0.998260007576
```

Figure 7: Accuracy of results for the best4KNN data set using (a) linear regression and (b) k-nearest neighbors (k=3).

Ripple Data Set: Changing Value for k in KNN

It can be seen that correlation to approximating the ripple data set increases with increasing numbers of nearest neighbors evaluated for KNN up to k = 3, after which increased number of nearest neighbors lowers the correlation. At the lower number of nearest neighbors evaluated, the model becomes overfitted. On the other hand, with larger numbers of nearest neighbors evaluated, correlation is lost due to loss in definition (oversmoothing, loss of detail).

(a)
```
In sample results
RMSE:  0.0
corr:  1.0

Out of sample results
RMSE:  0.237716222234
corr:  0.944111176194
```

(b)
```
In sample results
RMSE:  0.117863434564
corr:  0.985971678512

Out of sample results
RMSE:  0.213547134947
corr:  0.952952269598
```

(c)
```
In sample results
RMSE:  0.136590187312
corr:  0.981360326901

Out of sample results
RMSE:  0.207762150054
corr:  0.955537498166
```

(d)
```
In sample results
RMSE:  0.158319703229
corr:  0.975383222588

Out of sample results
RMSE:  0.212486985302
corr:  0.954609910672
```

(e)
```
In sample results
RMSE:  0.166240346459
corr:  0.973427600347

Out of sample results
RMSE:  0.221620653532
corr:  0.951772965431
```

(f)
```
In sample results
RMSE:  0.17594069633
corr:  0.97161540002

Out of sample results
RMSE:  0.237689111246
corr:  0.946508234451
```

Figure 8: Results for approximating the ripple data set using k-nearest neighbors with the number of nearest neighbors evaluated k = (a) 1, (b) 2, (c) 3, (d) 4, (e) 5, and (f) 6.

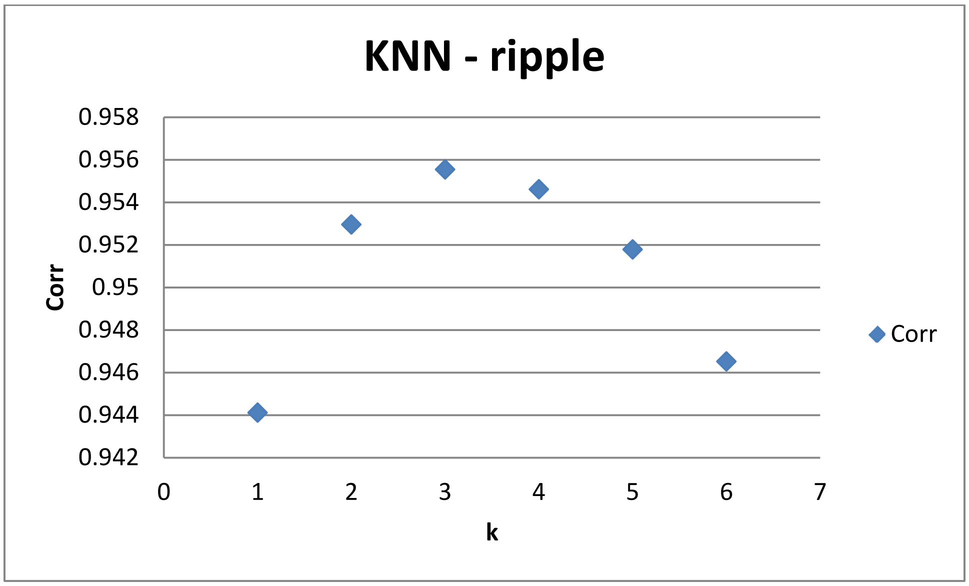


Figure 9: Correlation to the ripple data set as a function of varying the number of nearest neighbors (k) evaluated by KNN

Ripple Data Set: Changing Value for Number of Bags in Bagging

It appears that with lower numbers of bags (below 30), overfitting does occur, giving correlation values lower than those achieved without bagging on KNN with k = 3. However, with 30 a more bags, improvements are seen over the approximation without bagging. All bags 30 and above are an improvement and work as an improvement by reducing the overfitting of the KNN without bagging; however, it is seen that the correlation produced by bagging drops somewhere above 50 bags. This is likely due to oversmoothing the data.

(a)
```
In sample results
RMSE:  0.169693720822
corr:  0.975485890328

Out of sample results
RMSE:  0.232823455429
corr:  0.950062151042
```

(b)
```
In sample results
RMSE:  0.169774754086
corr:  0.976304006084

Out of sample results
RMSE:  0.2280607985
corr:  0.95449387325
```

(c)
```
In sample results
RMSE:  0.16356961356
corr:  0.97800363337

Out of sample results
RMSE:  0.217902188911
corr:  0.958430755008
```

(d)
```
In sample results
RMSE:  0.162369819428
corr:  0.97851865089

Out of sample results
RMSE:  0.217304026263
corr:  0.95874628687
```

(e)
```
In sample results
RMSE:  0.164129403517
corr:  0.978348111963

Out of sample results
RMSE:  0.220903314406
corr:  0.958091304814
```

(f)
```
In sample results
RMSE:  0.167172714057
corr:  0.977862621734

Out of sample results
RMSE:  0.223974190719
corr:  0.956519231592
```

Figure 10: Results for approximating the ripple data set using bagging (with KNN, k=3) with the number of bags = (a) 10, (b) 25, (c) 30, (d) 50, (e) 75, and (f) 100.

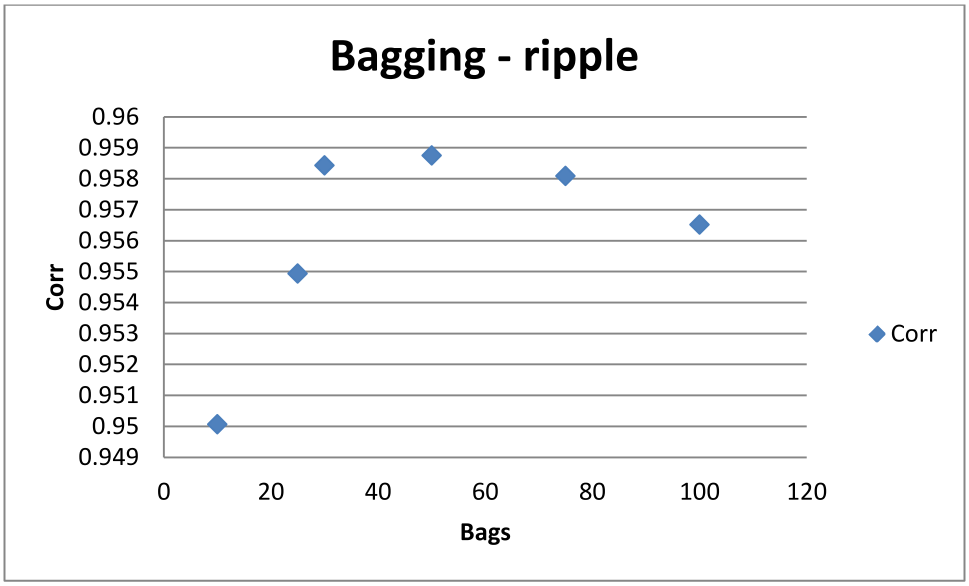


Figure 11: Correlation to the ripple data set as a function of varying the number of bags evaluated by bagging KNN (k=3).