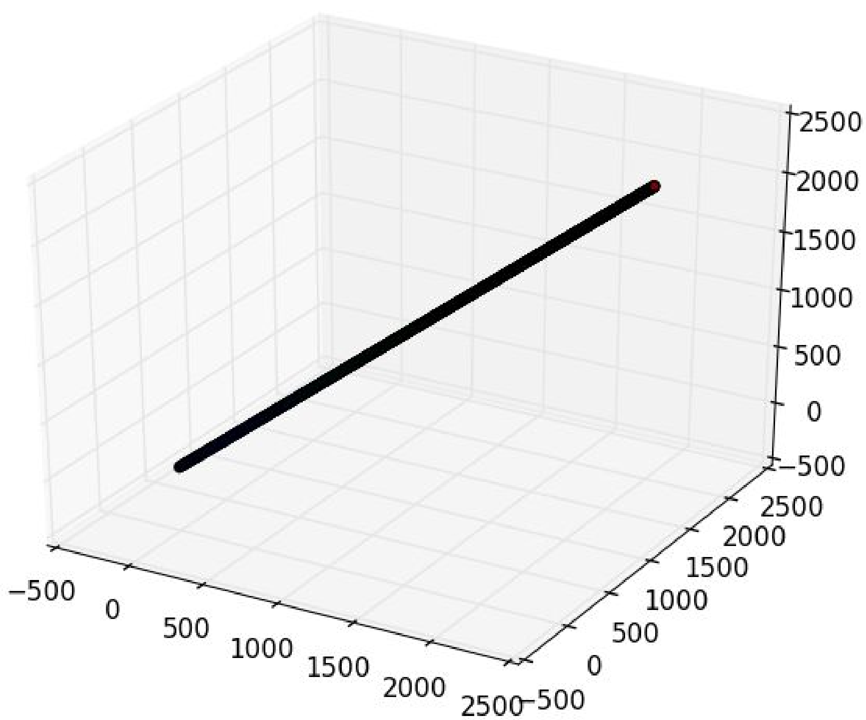
student: Yan Si

MC3-Project-1

best4linreg

Linear regression is best used when you can guess at the math equation underlying the data. I generated the new X data using a formula y=x1=x2.



Lets look at the results and discuss areas where linear regression is better than KNN:

Output of Learners Operating on best4linreg.csv

Linear Regression	KNN
linreg training time: 0.000999927520752 linreg query time: 0.0 In sample results RMSE: 3.16010812843e-13 corr: 1.0 Out of sample results RMSE: 7.35150466251e-13 corr: 1.0	knn training time: 0.0 knn query time: 0.27599978447 In sample results RMSE: 0.0408248290464 corr: 0.99999999309 Out of sample results RMSE: 463.179770715 C:\Anaconda\lib\site-packages\numpy\lib\function_base.py:1957: RuntimeWarning: invalid value encountered in true_divide return c / sqrt(multiply.outer(d, d)) corr: nan

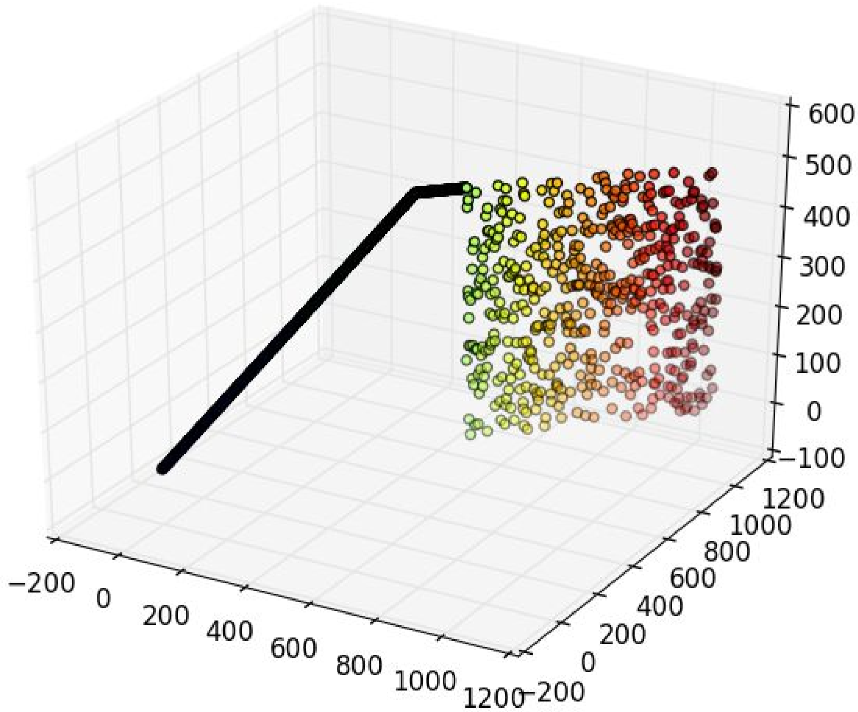
1. In terms of space efficiency, Linear is better because if you inspect the code, the LinRegLearner does not have any variables stored in self. KNNLearner has to store self.xTrain and self.yTrain. For the data I generated in

Data/best4linreg.csv, that's 28kB of RAM that was saved if linear regression is used.

2. **In terms of query time, linear regression just had to "plug into an equation" to get an answer. KNN had to consult its nearest neighbors.**
3. **In terms of RMSE, linear regression was very low for in sample and out of sample because it was able to model the equation and extrapolate that. KNN's error was higher for in sample. For out of sample, it went crazy because KNN is bad at extrapolating.**
4. **In terms of correlation, linear regression has good correlation in sample and out. KNN has a good number for in sample, but out of sample is undefined due to it being bad at extrapolating.**

best4KNN

KNN is better in situations when you can't guess at the underlying math equation. The example used in the lectures is to model the richness of a food source and the number of bees visiting. My algorithm generates the data that looks like this when plotted:



Let's say that there's a colony of 500 bees. There's food source where richness in sugar and pollen are denoted by the x1, x2 coordinates. How many bees visit that food source is denoted by y. So my graph shows that as the richness increases, the number of bees visiting it increases, but it tops out at 500 because that's the size of the colony. For a while, that number stays constant due to the colony size limitation. Then as the bees are full and develop eating disorders and bee diabetes, they start randomly visiting the food source as their health permits.

Output of Learners Operating on best4knn.csv

Linear Regression	KNN
linreg training time: 0.000999927520752 linreg query time: 0.0 In sample results RMSE: 97.6412854082 corr: 0.790342989765 Out of sample results RMSE: 427.125808035 corr: -0.0172251930416	knn training time: 0.0 knn query time: 0.0650000572205 In sample results RMSE: 36.8437720879 corr: 0.972913037091 Out of sample results RMSE: 141.847124815 corr: nan

1. In terms of training time, KNN does not require any, it simply stores the data whereas linear regression has to calculate the model.
2. In terms of RMSE, you can see that the KNN approach resulted in lower error
3. In terms of correlation, you can see the KNN approach has higher correlation.

Ripple KNN

K	RMSE
4	In sample results RMSE: 0.158319703229 Out of sample results RMSE: 0.212486985302
3	In sample results RMSE: 0.136590187312

	Out of sample results RMSE: 0.207762150054
2	In sample results RMSE: 0.117863434564 Out of sample results RMSE: 0.213547134947

For which values of K does overfitting occur?

When is out of sample error increasing and in sample error decreasing? When k=2.

Ripple KNN With Bagging

How does performance vary as you increase the number of bags?

Bags	Results
20	bagging training time: 0.000999927520752 bagging query time: 1.3220000267 In sample results RMSE: 0.130949460668 corr: 0.984169601791 Out of sample results RMSE: 0.196122000232 corr: 0.962787453667

40	bagging training time: 0.00100016593933 bagging query time: 2.60099983215 In sample results RMSE: 0.124500549103 corr: 0.985886922161 Out of sample results RMSE: 0.19329263732 corr: 0.96386445095
80	bagging training time: 0.00300002098083 bagging query time: 5.45799994469 In sample results RMSE: 0.125685949562 corr: 0.985680579574 Out of sample results RMSE: 0.194809937664 corr: 0.963754982468

As the number of bags increases, the training time and query time increases. The effects on RMSE and correlation numbers for both in sample and out of sample are negligible.

Does overfitting occur with respect to the number of bags?

Yes, overfitting can occur if the number of bags is 1. As you increase the number of bags, chances of overfitting lessens.

Can bagging reduce or eliminate overfitting with respect to K for the ripple dataset?

In the section above, I found that for k=2, overfitting occurs. If you increase the number of bags when k=2, you can reduce or eliminate overfitting. In the data I gathered below for k=2 as constant and the number of bags varying, you can see that as the number of bags increased, the RMSE decreased.

K=2

Bags	Results
20	In sample results RMSE: 0.107046489271 Out of sample results RMSE: 0.191844131836
40	In sample results RMSE: 0.100451743618 Out of sample results RMSE: 0.191512837061
80	In sample results RMSE: 0.100594924747 Out of sample results RMSE: 0.182838642023