Project 2 Report **Chao Shi**
CS7646, Summer 2016 GTID#: 902694979

1 Data generation

1.1 Best for linear regression

- Algorithm:
  1. Choose a 2-D linear function, say, $z = ax + by + c$. Sample x, y independently from uniform distribution over a certain interval $[0, d)$
  2. For training, add a Gaussian noise term for each (x, y) to get the final z value, i.e. $z = ax + by + c + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$. (x, y) corresponds to X, and z to Y in the project statement. Linear regression would calculate coefficients a, b, c by least square.
  3. For testing, randomly generate another set of (x, y), and calculate z without the noise term. Then using this set of points, (x, y, z) to test both linear regression and KNN.

- Explanation:
  It is necessary to include noise in training. Otherwise two methods will give the same result, as the linear interpolation nature of KNN ensures its prediction lies in the same plane, thus give exactly the same result of linear regression.
  I choose zero-centered normal distribution to eliminate the bias in noise, making sure the points mean of noise component is close to zero. This hidden information can be captured by linear regression because it utilize all the points to estimate the coefficients, but not by KNN because KNN only interpolates from a few nearby points, tuned to the noise.

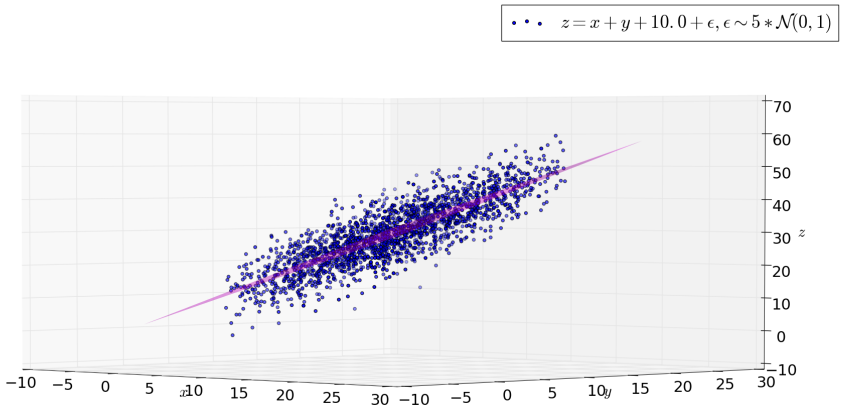

Figure 1: Slice view

Figure 1 shows a projection view of this type of data along the centering plane. We can see that Points are distributed more or less evenly across the plane $z = x + y + 10$. The

closer to the plane, the higher the density, due to the zero centered Gaussian noise.

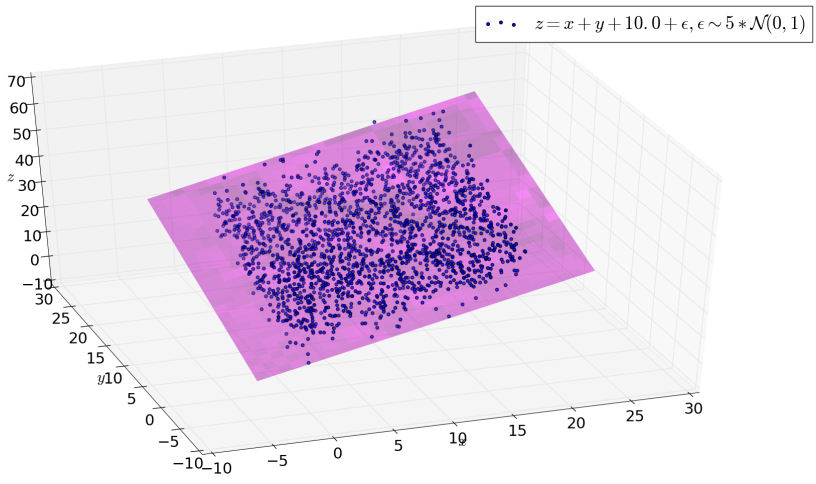

Figure 2: Side face view

Figure 2 shows another angle of the data distribution.

- Results:
  I compare error of the two methods over 10 training sets and 10 testing sets. Each training set contains 2000 points, while testing set contains 100 points.
  From Figure 3 we see that the training error of LinReg is constantly about 20% higher than KNN, which is expected because KNN only interpolate from nearby points, easy to be tuned to noise thus tend to overfit on linear problem. The right panel which shows that the testing error of KNN is an order of magnitude higher than LinReg, further demonstrates that.

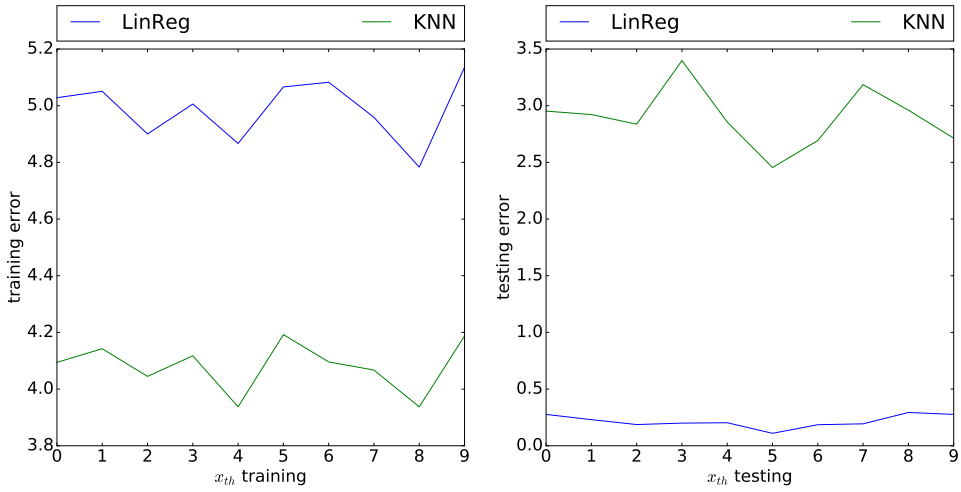

Figure 3: LinReg wins KNN

1.2 Best for KNN

- Algorithm: Simply construct a local 2-D "step-function" in two square areas:

$$z(x,y) = \begin{cases} 0 & if \quad (x,y) \in [0,10) \times [0,10) \\ 100 & if \quad (x,y) \in [10,20) \times [10,20) \end{cases}$$

  Both for training and testing. An sample of data points is shown in Figure 4:

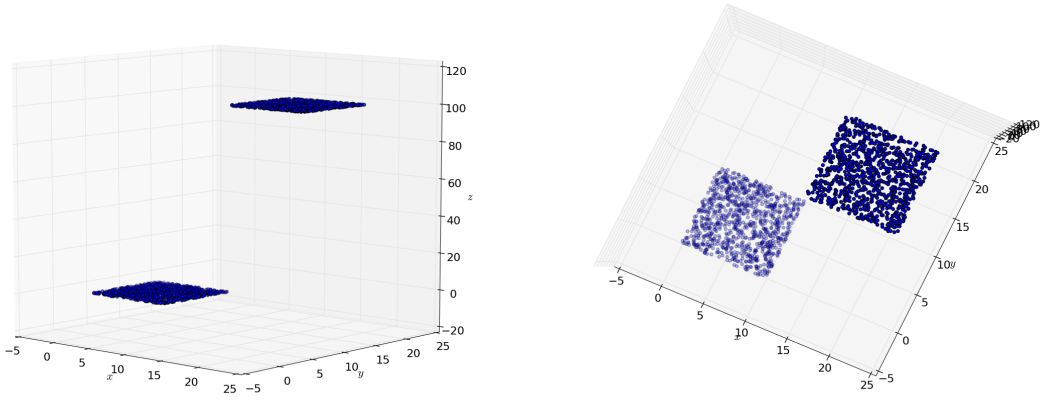

Figure 4: 2-D step function

- Explanation: The sharp change at the intersection of two "floors", is highly non-linear. KNN on the other hand,it's locality, is very suitable for this kind of data. In fact it's just two well separated clusters. The result from linear regression would just be a plane cutting through the negative diagonal of the two squares.

- Results: KNN have almost zero error for both training and testing. The only non-zero error happens on the points that are close to the intersection point of these two areas, in which case one of their closest neighbors might be in the other area, thus can "drag" the predicted value out of the plane. LinReg has very high RMSE as expected(Figure 5).

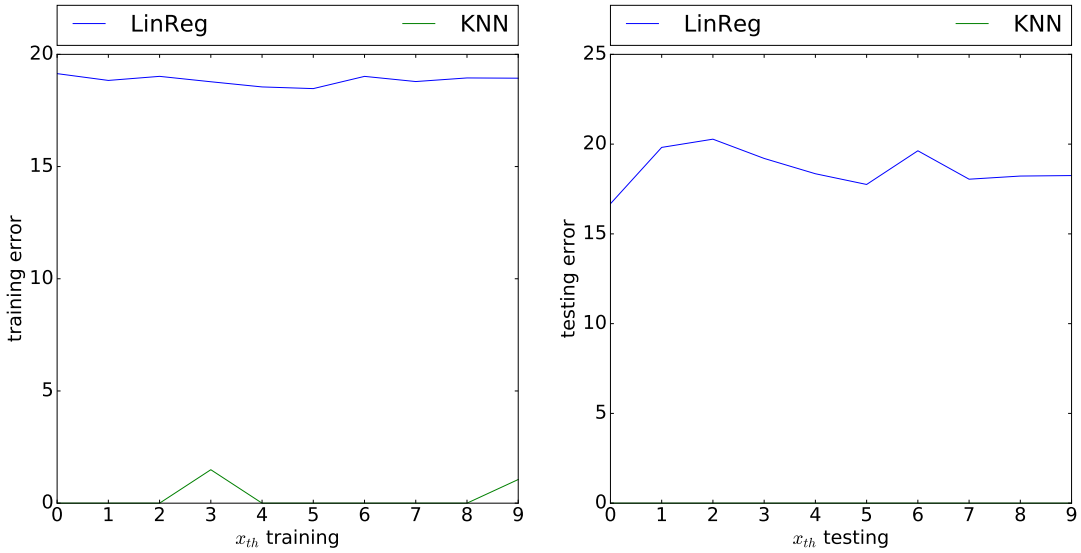

Figure 5: Training and Testing error on step-function-like data

To see visualize linear regression result , we can
plot the plane fitted by least square on top of this data set, in Figure 6:

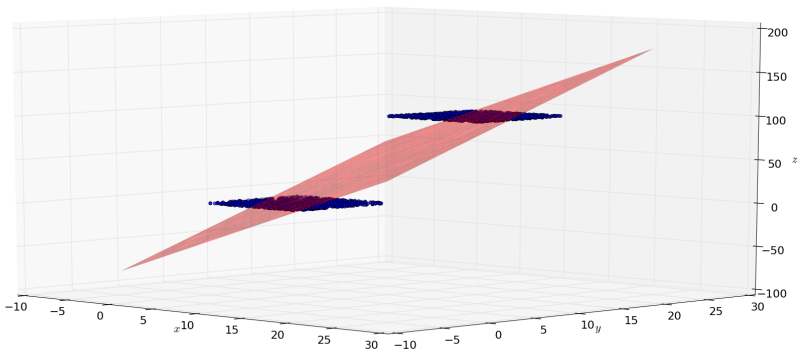

Figure 6: Training and Testing error on step-function-like data

We see that it looks exactly as expected. Although the error is huge, we can easily see that this is the best linear regression can do in this case.

2 Overfitting

2.1 KNN only

As shown in the Figure 7 below, training error decreases when K decreases, and the the curve became much steeper right after K reaches down to 3. On the other hand, test error decreases until K reaches down to 3, and starts increasing afterward. So overfitting happens when K became smaller than 3.

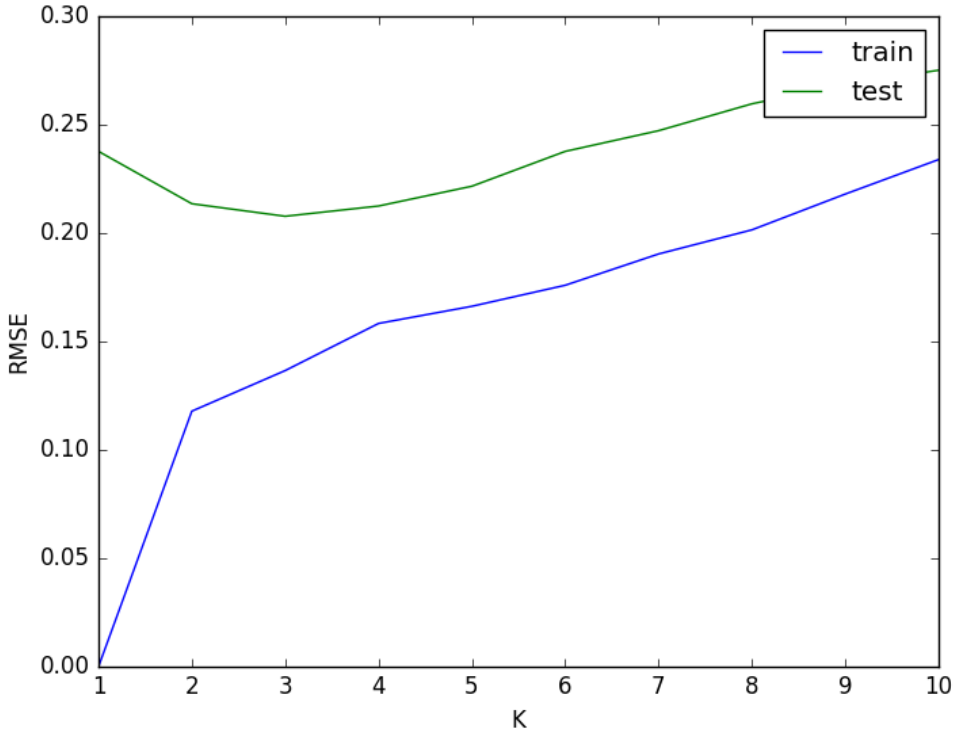

Figure 7: Overfitting test of KNN

2.2 Fixed K = 3 with varying bagging

Both training and testing error decreases at first when number of bags increases. Then stays around a certain level after #bags reaches 20. No overfitting because bagging is just an averaging process, which suppresses overfitting by smoothing out high frequency noise. See Figure 8.

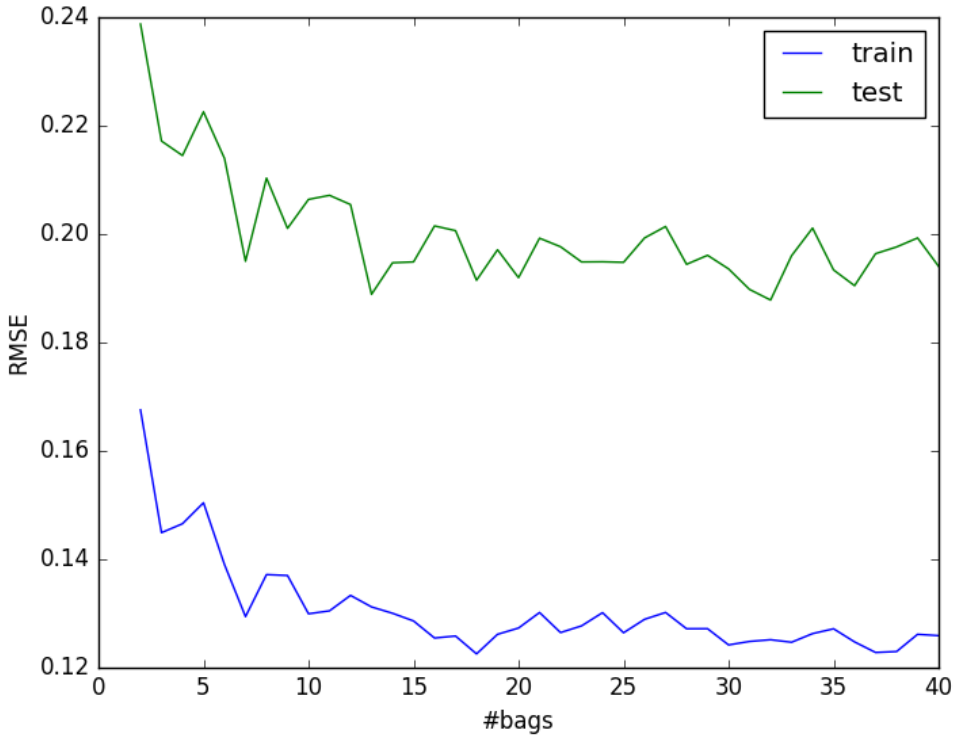

Figure 8: Overfitting test of Bagging with fixed K

2.3 Varying K with fixed bags=20

As mentioned in section 2.2, bags should suppress overfitting because it is essentially an averaging process among different samples of the original training data. The experiment shown in Figure 9 agrees with the thought:

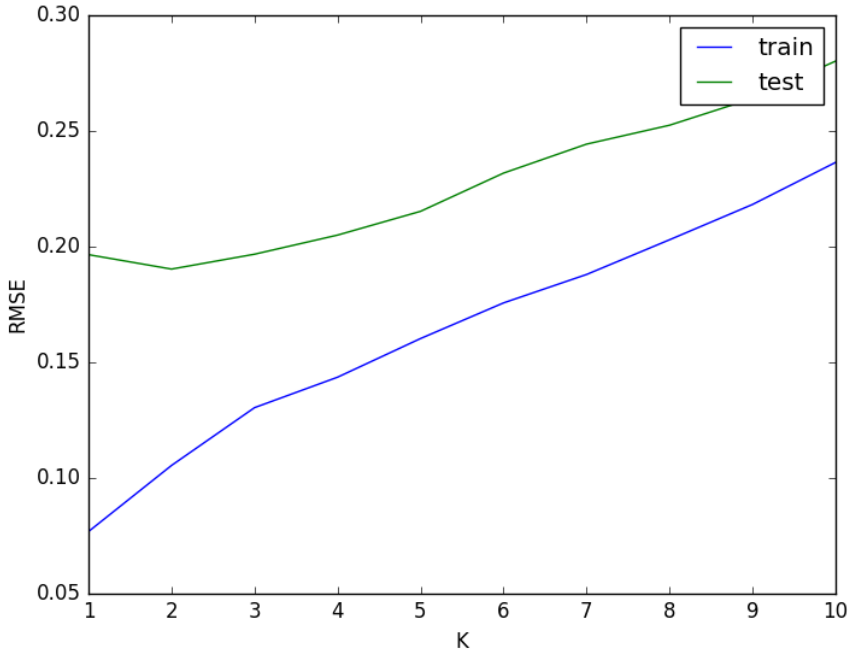

Figure 9: Overfitting test on Bagging with fixed bags and varying K

As we can see from the plot above: 1.testing error doesn't start to increase unitl K reaches 2, as opposed to the critical point of K = 3 without bagging; 2. Even when K further decreases from 2 to 1, the testing error just increased slightly, the slope is much lower than without bagging; 3. The decreasing rate of training error at small K is also significantly lower thant that of no bagging, as can be compared to plot in section 2.1. These facts demonstrate that bagging can suppress overfitting very well, if not eliminate it completely.