Emperical Comparison of Simple Linear Regression, Knn and Ensemble Learning Methods

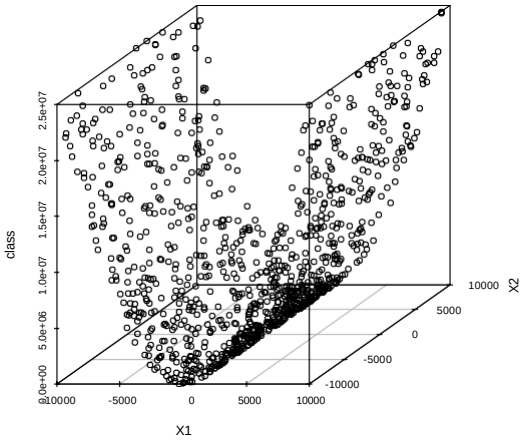
Sridhar Natarajan

This report illustrates the differences between simple linear regression (unregularized single layer), knn regressor and ensemble methods like bagging. To bring about the contrasts, we generate respective datasets in which each algorithm outperforms the other. Finally, we analyze the impact of bagging in improving generalization and understand tradeoffs in performance metrics in the ensemble method

Knn Tops Linear Regression:

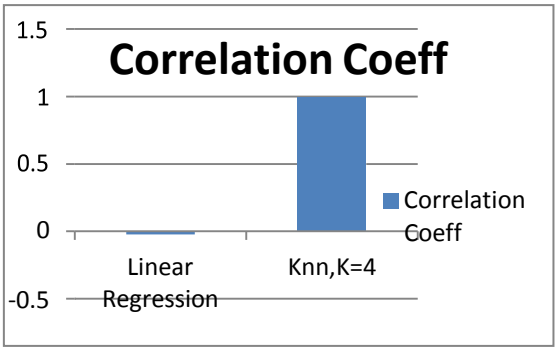
Steps to create knn favourable dataset(X1, X2 are input attributes. Y is the value that must be predicted)

1) **X1 = choose 1000 samples between -100000 and +100000**
2) **X2 = choose 1000 samples between -100000 and +100000**
3) **Compute Y = X1*X1//4**

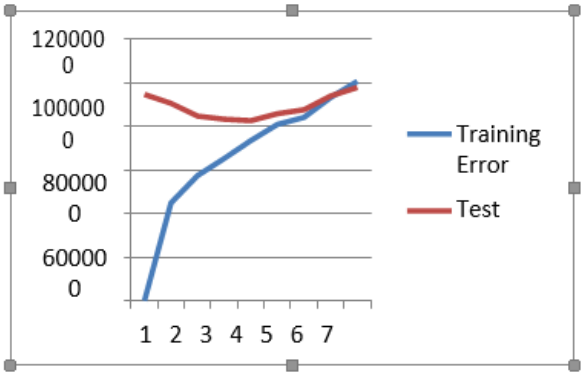


The dataset must perform relatively much better in the knn and worse in the linear regression learner. We exploit weakness of linear regression. It can work best with linear models. We introduce a non –linear relationship between input and output. This means the linear linear regressor will have higher errors between predicted and actual v**alues. Also, the sparse nature** of the data means that the errors due to non-linearity will get exacerbated. To help knn perform better we introduce the quotient operation which adds to abruptness and reduces errors when k gets closer to the divisor.

Higher correlation coefficient between test-data and predicted values (~1) implies higher quality of learning and lower rmse. Comparison performance of Knn vs Linear Regression is inferred from the below chart:



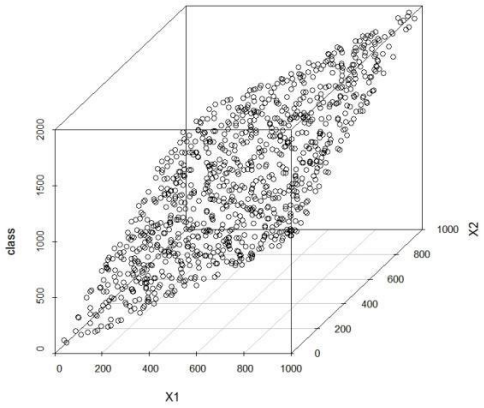
The plot of training error and test error against increasing values of K is shown. To its bottom are charts comparing train and test errors of linear regression and knn. The charts vindicate superior knn performance over linear regression



Linear Regression Tops Knn

Steps to create linear regression favourable dataset(X1,X2 are input attributes. Y is the value that must be predicted)

1) **X1 = choose 1000 samples between 0 and 1000**
2) **X2 = choose 1000 samples between 0 and 10000**
3) **Compute Y = X1+X2**

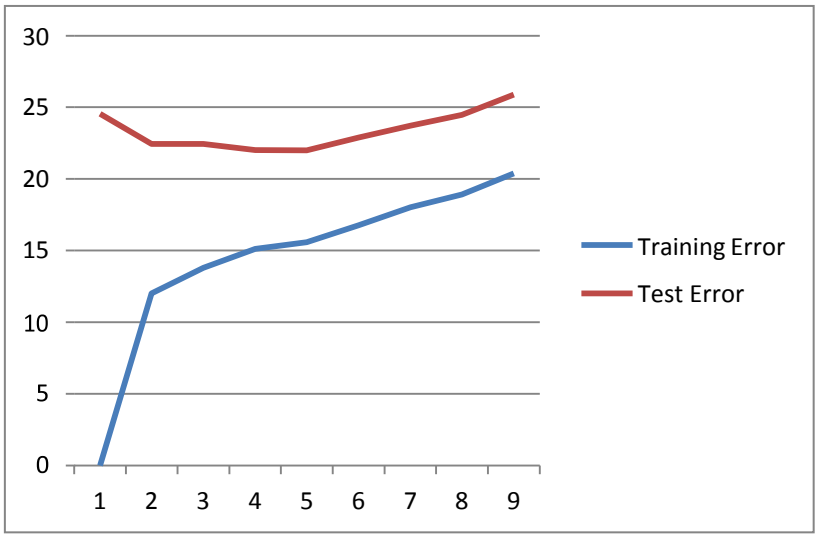


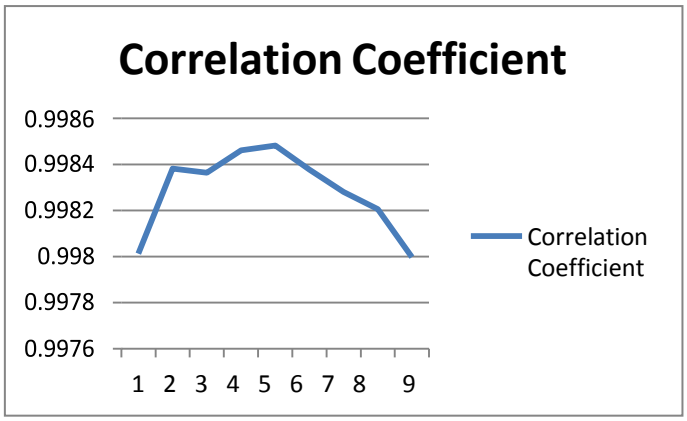
Now X1 and X2 have a perfect linear relationship. In this case, we expect a parametric model method like linear regression to perform better than knn.As it can be seen, the linear model in 3d space is a plane and all the generated points lie in the plane.

Test Results From Linear Regression: Training Error=0,Test Error = 0, Correlation Coefficient =1

Test Results From Knn:

Results from Knn are shown below.The training and test error are plotted against increasing K values. The following graph shows progression of error (or decsreasing correlation coefficient with increasing K).In this dataset, the superior performance of linear regression is obvious:

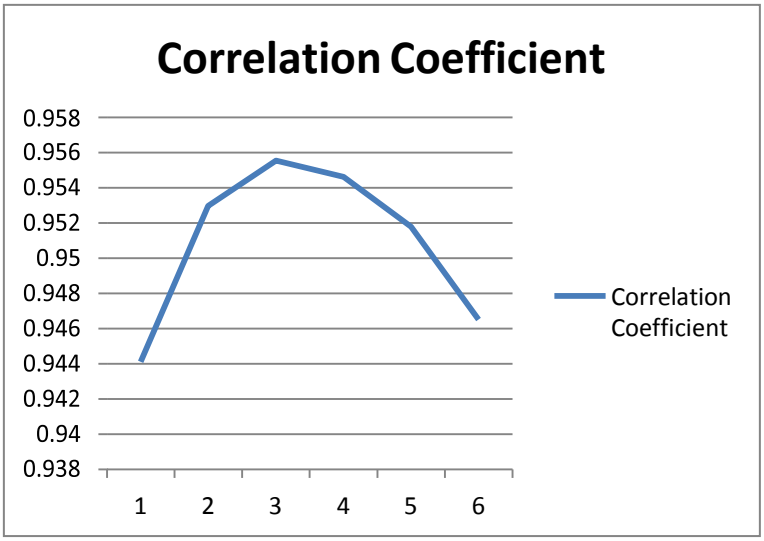


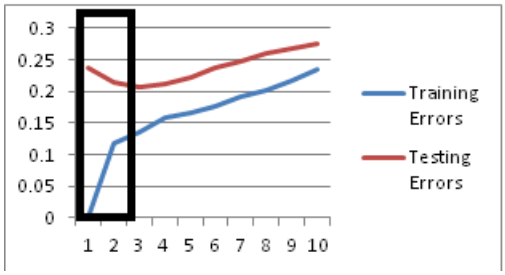


Overfitting

Overfitting happens when your prediction model tracks the training data too closely, that it performs poorly in unseen (or test data). After all, the real purpose of building a learning model is that it generalizes well and performs better on training data. Overfitting occurs when the complexity of the model is high. In knn, the model complexity is high when K=1.Geometrically, this means that the model is more abrupt in adapting to the training data. As K increases, the model becomes smoother and less abrupt.This results in improved performance(less RMSE) against test data. So we want to increase K (or decrease model complexity) until the test error bottoms out and use that value of K for the model. In this case, for K=3, the test error bottoms out exiting the overfitted region. Hence optimal k value for this data set is 3.

The below charts indicate the shaded region where training error is relatively less but test error is very high. Another indicator for test error bottoming out is the correlation coefficient obtained from the test data and predicted values. As it can be seen at K=3, the correlation peaks and that is where test error(rmse) hits bottom.

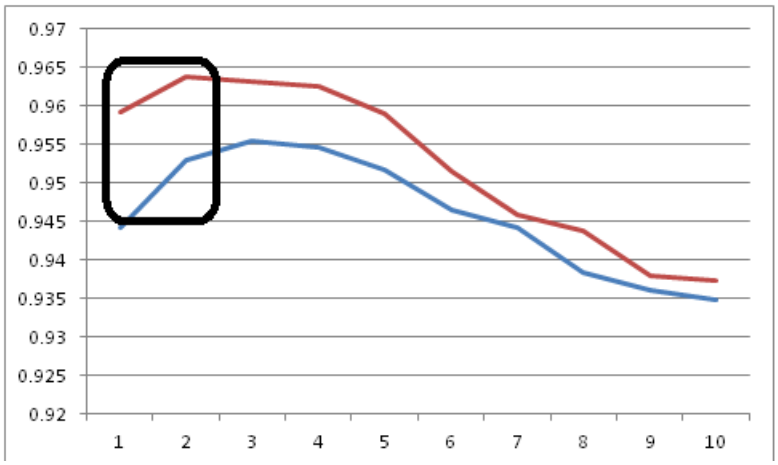




Overfitted Region is present in the box indicated by the thick black rectangle in the above chart. Overfitting occurs when K=1, 2.

Ensemble Methods - Bagging

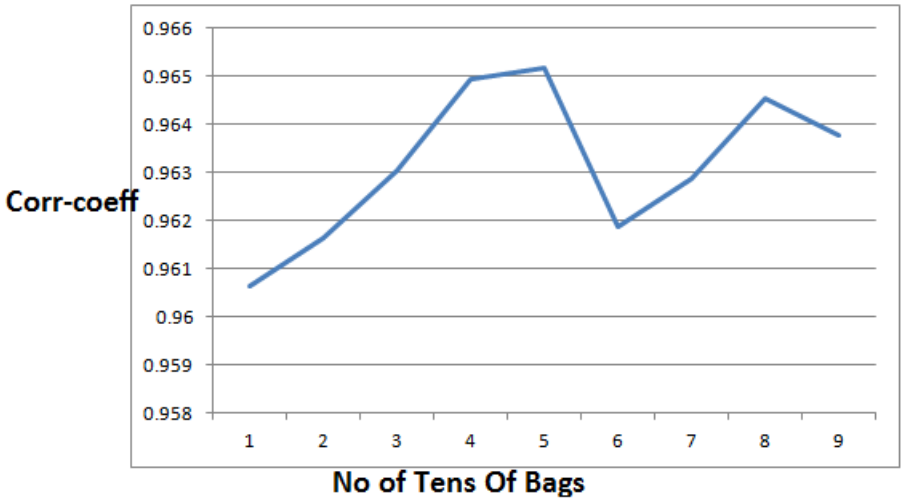
Now, we use the results of bagging and understand how averaging helps in reducing test errors.The Red line indicates results from bagging and blue indicates results without bagging. The y axis is plotted with correlation coefficient. It is understandable from the chart that bagging does help reduce the rmse (and increase correlation coefficient of predicted test data).It helps reduce the testing errors in the region of overfitting (k=1, k=2) in the below chart. ***Hence, bagging does help reduce overfitting.The extent of reduction in overfitting is indicated by the box in the below chart.***



Bagging simply performs averaging on samples that may have repeated values. The sampling with replacement helps reduce the occurrence of any values that may have high variance from the rest of the population. This provides a learning model that has less variability than its unbagged counterpart

Below are the results of correlation coefficient against the number of bags.

When the bags hit 40, the correlation coefficient seems to have peaked.Hence 40 can be a reasonably optimal bag size to use for the ripple dataset



Trade Offs - Computational cost of bagging

Increasing the bag size comes at a very high computational cost of training and prediction (mostly prediction).To understand the cost to benefit, please infer the below normalized charts that indicate the computing time and progression in correlation coefficient with increased number of bags(tens of bags, so 9 in y axis means 90 bags).Hence, depending on the data set, we have to settle for an optimal bag size that we can afford in terms of performance