CS5785 HW1

1. Variance of asum. Show that the variance of a sum is var[X−Y]=var[X]+var[Y]−2cov[X,Y], where cov[X,Y] is the covariance between random variables X and Y.

According to the definition:

1. Bayes rule for quality control. You’re the foreman at a factory making ten million widgets per year. As a quality control step before shipment, you create a detector that tests for defective widgets be- fore sending them to customers. The test is uniformly 95% accurate, meaning that the probability of testing with positive outcome given that the widget is defective is 0.95, as is the probability of testing negative given that the widget is not defective. Further, only one in 100,000 widgets is actually defective.
   1. Suppose the test shows that a widget is defective. What are the chances that it’s actually defective given the test result?

According to Bayse-rule:

in which, is the possibility that a widget is actually defective, is the possibility of testing a widget is defective, is the possibility a defective widget testing to be defective.

so,

* 1. If we throw out all widgets that are defective, how many good widgets are thrown away per year? How many bad widgets are still shipped to customers each year?

The probability of good widgets testing to be defective is:

is the possibility that a widget is actually not defective, is the possibility a not defective widget testing to be defective.

The probability of defective widgets testing to be non-defective is:

is the possibility that a widget is actually defective, is the possibility a not defective widget testing to be not defective.

1. In *k*-nearest neighbors, the classification is achieved by majority vote in the vicinity of data. Suppose our training data comprises *n* data points with two classes, each comprising exactly half of the training data, with some overlap between the two classes.
   1. Describe what happens to the average 0-1 prediction error on the training data when the neighbor count *k* varies from *n* to 1. (In this case, the prediction for training data point *xi* includes (*xi* , *yi* ) as part of the example training data used by *k*NN.)

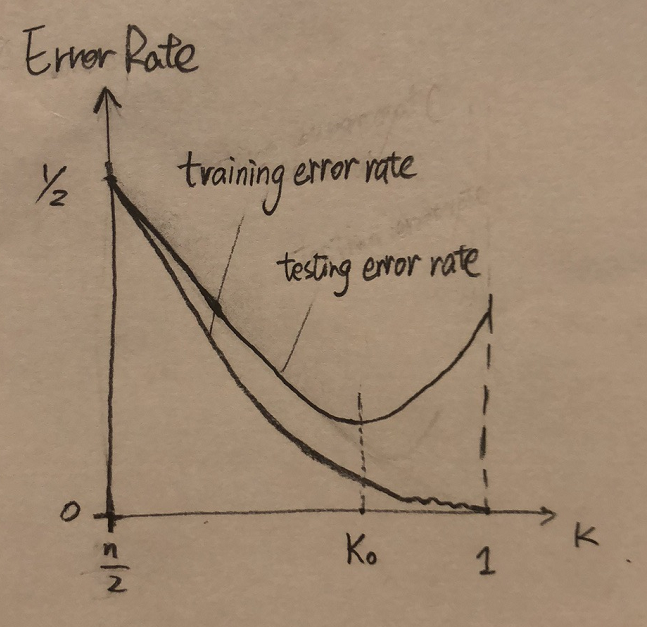
The average prediction error would be huge at fisr, when k = n(which equals to 0.5,this is due to the equal classification of the dataset). However, as the k becomes smaller and smaller, the error would become smaller.

And when k equals to 1, the prediction error would euqal to one. Because the prediction for training data point is a part of training data.

So, the range of prediction error rate would be monotonically increasing from zero to 1/2, when k grows from 1 to k.

* 1. We randomly choose half of the data to be removed from the training data, train on the remaining half, and test on the held-out half. Predict and explain with a sketch how the average 0-1 prediction error on the held-out validation set might change when *k* varies? Explain your reasoning.

The sketch is below:



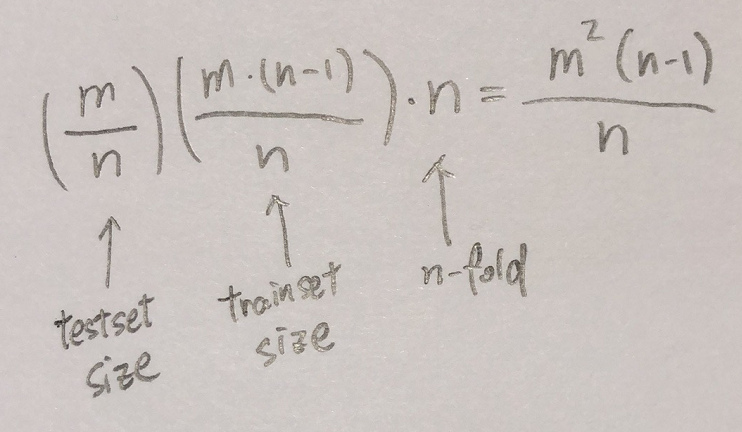
where, x-axis is the value of k, and the y-axis is the prediction of error rate.

In the training-set, when k equals to n/2, the error rate would be 1/2 at first as the even distribution of two classification, and when k decreasing, the error rate is decreasing also. But according to the overlap of data, the error will have a fluctuation behavior as k goes down. And equals to zero when k=1, as the nearest neighbor is itself.

In the test-set, when k equals to n/2, the error rate would also be 1/2 with the same reason, and when k decreasing, the error rate would decrease since it get less and less under-fitting. However, when reach the point where k = , the error rate would increase again as the difference between training-set and test-set, since it starts to be overfitting.

* 1. We wish to choose *n* by cross-validation and are considering how many folds to use. Compare both the computational requirements and the validation accuracy of using different numbers of folds for *K*NN and recommend an appropriate number.

Since KNN is lazy learning algorithm, there is no computation cost on training the model. The main cost is finding K nearest neighbors by iterating the whole training set. Let’s look at the cost of iteration with n-fold cross validation with m records (data size):



The overall computation cost is O(m^2), which doesn’t increase a lot as the #fold n increased. However, as the #fold n increase, the cross validation process actually used more data for training data set for each fold, and at the same time, keep the size of the whole testing data space (sum all folds) the same, thereby increasing the validation accuracy.

Conclusion: increasing the #fold n especially for KNN (lazy learning algorithm), doesn’t significantly increase the computation cost, but that increase the validation accuracy.

* 1. In *k*NN, once *k* is determined, all of the *k*-nearest neighbors are weighted equally in deciding the class label. This may be inappropriate when *k* is large. Suggest a modification to the algorithm that avoids this caveat.

It is the even weight of the neighbors caused the inappropriate. So what we can do is turning the even weight to the uneven weight of each neighbors, according to the distance between neighbors and the test data.

* 1. Give two reasons why *k*NN maybe undesirable when the input dimension is high.

1) The curse of dimensionality: need more computational cost or lack of training dataset to achieve accuracy. One of a very important value of kNN is distance. As the dimension increase, the ratio of the number of other sample points within a unit distance from a sample point decreases, which causes us to go further distance to find neighbors, and the data in a unit space get more and more sparse. The sparse data cause instability or less robustness of statistics, thereby leading to poor behavior on testing data set. To get rid of that, we need exponentially increased training data set, but that make the computation cost increase exponentially. Therefore, kNN is undesirable when the input dimension is high.

2) kNN strictly require each dimension is evenly important, like computer vision. If the dimension is high, it is possible that some dimension is not that important as others. In the case that some dimensions don’t have the same importance as others, since kNN mainly use the distance as classification method, those less important dimension are still treated as same weight with other dimensions, and thereby the kNN is not a suitable classification anymore.