CSC411 Project#1 Bonus

Yifei Dong

February 5th 2018

1 Introduction

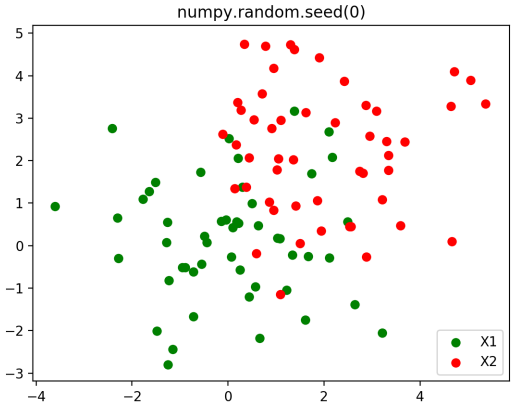
K-Nearest Neighbours is a technique that could be used to classify the labels of objects. Assume there are already some objects with different labels on a graph and there is another object that we don't know its label. One of the methods that could be used to determine the label of the object is k-Nearest neighbours. Suppose the object the we want to classify is a. We can first find the k nearest objects around a. Then we find the label with largest number of objects among the k objects. Suppose the label is x. Then object a has label x.

For a classifier build by k-Nearest neighbours, the magnitude of k might influence the accuracy of the classifier. How the data is generated might also have an impact on the performance of the classifier. In class, we saw that the performance of k-NN on the training set is 100% for k=1, with the performance stabilizing to the proportion of the plurality label in the training set when k becomes very large. I want to explore the question of whether, and under what circumstances, the performance on the training set can improve when k becomes larger.

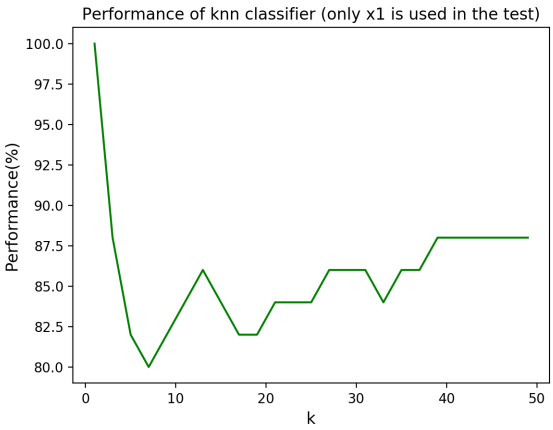
In this project, I build classifier to classify the label of objects. The technique of k-Nearest neighbours is used. I first build two random samples in Gaussian distribution with different means. The first set has mean (0, 0) and covariance ((2,0),(0, 2)) and has label $x1$. The second set has mean (2, 2) and covariance ((2, 0), (0, 2)) and has label $x2$. The two sets together are my training set. The size of $x1$ and $x2$ are both 50. I calculated the distance between points by Euclidean distance. Moreover, I'm only using odd numbers as k to avoid the case where both labels have the same amount and the most common label could not be defined.

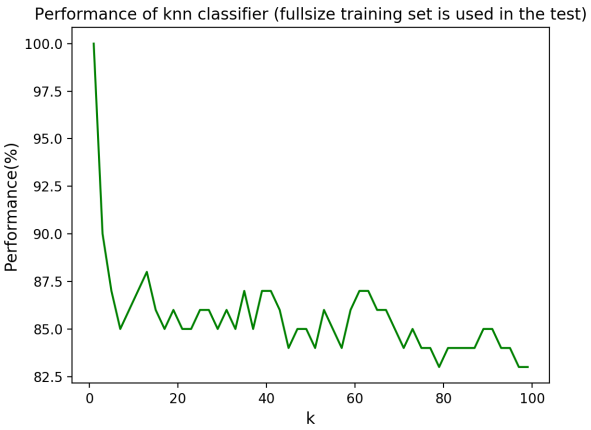
2 Examples

I first set numpy.random.seed(0).

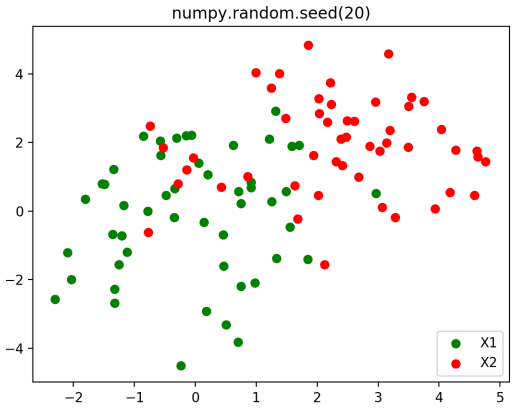


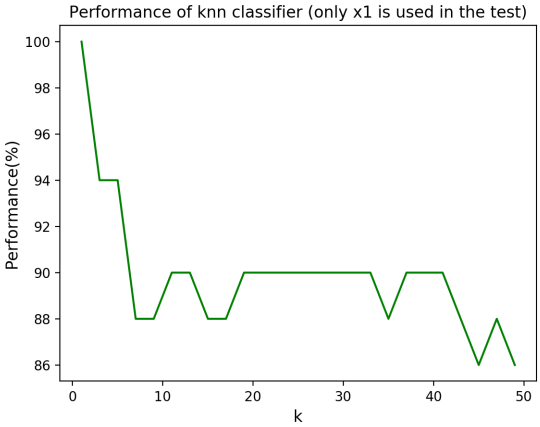
I tested the the classifier in two ways. One is using only half of the training set which is $x1$, another is using the full training set. Below are the graphs of the performance of the classifier with different k. The x-axis represents k and the y-axis represents the performance in percentage.

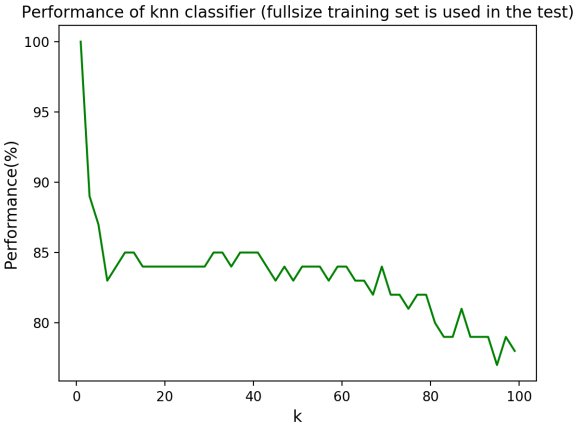




Then I set numpy.random.seed(20) and tested the same thing accordingly. Below are the graphs.







3 Discussion

From the above examples, we could see that the performance of the classifier decreases as k increases in most cases, especially when the full training set is used.

The only case that the performance increases as k increases is when I was only using $x1$ to test the classifier and the seed is set to 0. The performance increases as k increases in the range k = [9, 40]. It is shown that the points of the two Gaussian distribution are mixed in range [-1, 3] in the first graph, where the points are merely mixed when seed is 20 in the fourth graph. Therefore, the performance is more likely to increase as k increases when the points are mixed.

Suppose we are classifying a point in the range [-1, 3], where points with different labels are mixed. With a smaller k such as 10, the number of neighbours labelled "x1" might be close to neighbours labelled "x2". The

neighbours with label "x2" might be more common since the points are mixed. However, if k is 40, we are evaluating more neighbours of the points, and since it is a bigger range, the points outside the range [-1, 3] will also be neighbours. More neighbours will have label "x1" in this case. Therefore, in this special case, the performance of the classifier increases as k increases in range [9, 40].