Kernel k-Nearest Neighbours

Name: Narendra Kumar Manoharan

UTA ID: 1001237691

Task: Machine Learning Assignment 2 - Report

Abstract

In this project, we aim to explore the use of kernels in k – Nearest Neighbours and how they affect the classification with respect to the normal k - Nearest Neighbours. Here, we try and compare the performance of the k - Nearest Neighbours algorithm which usually uses a Euclidean distance measure to calculate the distance between the neighbours but also we try to Kernelate the k - Nearest Neighbours algorithm using the widely used kernels for Support Vector Machines such as Polynomial and Radial Basis Function Kernel. We intend to compare the results of the Kernel k-NN with the normal k-NN results to see if using the Kernel really does improve the classification rate as the paper claims it does. We plan to implement a k-NN algorithm in conjunction with the 10-FCV algorithm that we implemented for the previous assignment and use 3 datasets to cross validate the results.

Introduction

k-NN which stands for k – Nearest Neighbour is an instance based lazy learning algorithm where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. Initially we take an example and the instances of the class with the class labels and try to predict the class label of the new example by calculating the distance from each instance to the example. We then, calculate the top-k nearest neighbours and assign the most frequent class label to the example. Usually, the training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

The algorithm mostly depends on how well the distance function is and for that we usually take the **Euclidean distance** measure to compute the distances between all the training examples and the testing sample. Also, the accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. The k-NN algorithm is also sensitive to the local structure of the data (i.e.) It is susceptible to the local maxima thereby behaving similar to a greedy algorithm. Feature extraction, Data reduction and Dimension reduction might also help increase the prediction accuracy of the algorithm. There are other distance measures that are also utilized in the normal k-NN algorithm such as Mahalanobis distance, City block metric, Minkowski metric,

Chebychev distance, Cosine distance, Correlation distance, Hamming distance, Jaccard distance and Spearman distance.

Kernel k-NN

Kernel k-NN stands for Kernel k – Nearest Neighbour which is an instance based lazy learning algorithm. It is identical to the normal k-NN but diverging in the use of a kernel function to calculate the distance or similarity in the place of using a normal distance measure. In this method we employ the kernels used in Support Vector Machines to take the place of the distance used in k-NN to find the similarity between examples or another metric to help predict the outcome of the test example. We intend to compare and contrast the results of both these approaches to k-NN.

Distance Approach

In this experiment, we are going to use the Euclidean distance measure to find the distances between the instances for the normal k-NN algorithm.

The Euclidean distance for an n-dimensional space is defined as

$$\mathrm{dist}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Here, x and y are the arbitrary vectors.

This distance measure is the default measure used in finding the relative distances between the examples. So that we can calculate the top-k neighbours and predict the class label for the new instance from there.

Kernel Approach

Here, rather than using the distance measure, we intend to use the kernels associated with SVM to find the similarity measure between the vectors.

The Polynomial Kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle)^p$$

Here, x and y are the arbitrary vectors whereas p is an adjustable parameters of the above kernel function and $\langle x, y \rangle$ is the inner product of the two vectors.

Another kernel that we use is the Radial Basis Function kernel that employs Gaussian kernel tricks to calculate the similarity between the vectors.

The RBF kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{\sigma^2}\right\}$$

Here, x and y are the arbitrary vectors whereas σ is an adjustable parameters of the above kernel function and $||x - y||$ is the Euclidean distance of the two vectors.

Datasets

Here, the datasets for the kernel k-NN algorithm are collected from the UCI Machine Learning repository which the most extensive library for ML datasets. The datasets attribute removal process is done to make sure that the unnecessary attributes for classification process are also along with the attributes labels.

E. coli Dataset

Source: http://archive.ics.uci.edu/ml/datasets/Ecoli
Number of Instances: 336
Number of attributes: 8
Names of attributes: Sequence name mcg, gvh, chg, aac, alm1, alm2
Purpose: It is used in the classification of the location site of the protein

Glass Dataset

Source: http://archive.ics.uci.edu/ml/datasets/Glass+Identification
Number of Instances: 214
Number of attributes: 10
Names of attributes: id, RI, Na, Mg, Al, Si, K, Ca, Ba, Fe
Purpose: It is used in the classification of the type of glass

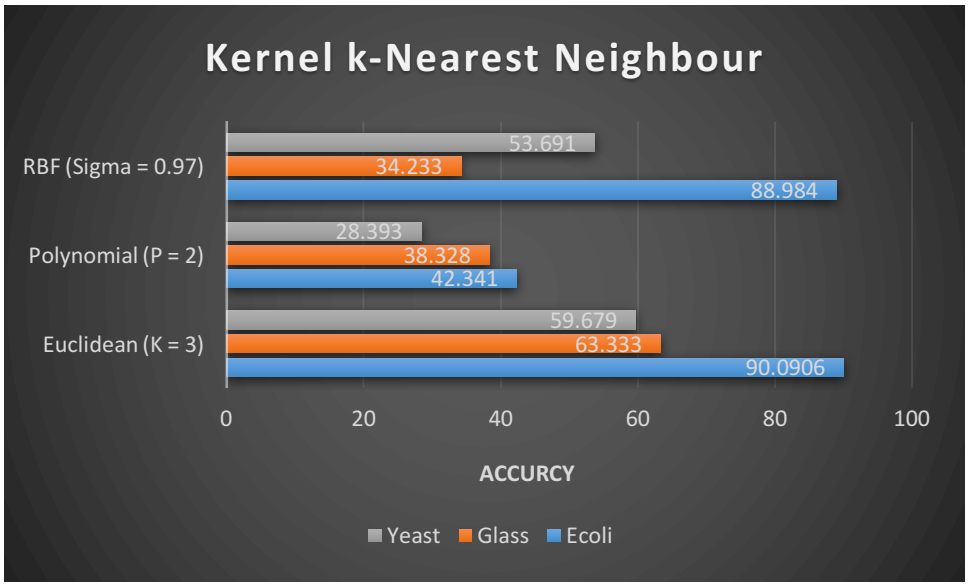
Yeast Dataset

Source: http://archive.ics.uci.edu/ml/datasets/Yeast
Number of Instances: 1484
Number of attributes: 8
Names of attributes: Sequence name, mcg, gvh, alm, mit, erl, pox, vac, nuc
Purpose: It is used in the classification of the location site of the protein

Experiments and Results

Here, we have three datasets (E. coli, Glass, Yeast) that we use to test the accuracy of the k-NN and Kernel k-NN algorithms. We implement the Euclidean distance measure for the normal k-NN algorithm also implement Polynomial and Radial Basis Function kernel for the Kernel k-NN method. We can see the experimental results in the tabulation below.

DATASET\KERNEL	EUCLIDEAN (K = 3)	POLYNOMIAL (P = 2)	RBF (SIGMA = 0.97)
ECOLI	90.0906	42.341	88.984
GLASS	63.333	38.328	34.233
YEAST	59.679	28.393	53.691



In this graph, we can see the experimentation results of both the algorithms of all the datasets. Further analysis of the result reveals the following. In this experiment, we use the following values as constant for all values of k = 3, p = 2 and sigma = 0.97. We use 10-fold cross validation to create the test and train datasets where we randomize the dataset each and every time to create new dataset to get a greater precision and accuracy.

From the graph we can see that, normal k-NN constantly outperforms both forms of kernel k-NN which uses the Euclidean distance measure to find the distance between the nearest neighbours. This might be due to the fact that distance is a better measure than similarity when it comes to k-Nearest Neighbours. Although we can see that RBF kernel comes close in terms of accuracy with the Euclidean, normal k-NN still outperforms it. There seems to be a threshold as to the accuracy of RBF kernel k-NN which is at most equal to the performance of the normal k-NN. We can see that implementing a similarity measure for the k-NN method does not necessarily increase the accuracy of the classification.

Discussion

Here, we will discuss why implementing the kernels in k-NN does not improve the accuracy but rather reduces the accuracy from the already existing k-NN. In the paper that we have chosen, it is said that the higher the dimensions then finding the linear classifier should be that much easier. But since higher dimensions increases the time complexity this might be more difficult so implementing Kernels is said to help in that regards. But in regards to k-NN linear separability doesn't really help since that does not mean that the vectors of the same classes are going to be closer together. So this might be the reason that this kernel approach does not yield any positive result other than the RBF kernel coming close to the performance of the normal k-NN measure.

But the results posed by the authors of the paper might be true for that specific dataset where they have selected certain features and mapped the features using feature mapping. In this instance, they might have seen a result where kernel k-NN produces a better classification. This might be purely coincidental as the result cannot be reproduced using other datasets.

Result

From the experimental results, we can conclude that Kernel k-NN does not perform any better than the normal k-NN algorithm which uses Euclidean distance measure to classify the instances. So, the kernels do not benefit the k-NN algorithm and the implementation of kernels in k-NN does not necessarily increase the classification accuracy.