Department of Computer Science

Assignment 1 – K-Nearest Neighbours and variants

Module Name: Practical Machine Learning–COMP9061

Student Name: Sreekanth Palagiri - R00184198

About K Nearest Neighbour

K-Nearest Neighbour is an instance-based learning algorithm. KNN predicts output by comparing a new instance with instances in training data. KNN is used for both classification and regression.

KNN Algorithm Pseudo code:

1) Find K similar instances to x-test that are in x-train
2) Get the labels Y for the similar instances found in 1
3) Predict by combining labels.

We need to specify below for KNN:

1) K – No. of nearest neighbour
2) A distance metric
3) Aggregation technique (Majority Voting vs Distance Weights etc)

Accuracy Achieved in Assignment:

Technique	K	Accuracy/R2 Score
Simple Voting - Classification	1	89.5%
Simple Voting – Classification	6	92.3%
Distance Weighted - Classification	10	92.7%
Distance Weighted - Regression	5	84.25%
Distance Weighted - Regression	10	85.23 %

Part 2: Improving KNN Classifier

In this section we will look at ways of improving KNN. Methods below are applicable for almost every ML algorithm. First, we will look at tuning Hyper Parameters like value of K, distance metric, weighing methods etc. We will then investigate feature improvement techniques like feature scaling, feature selection etc

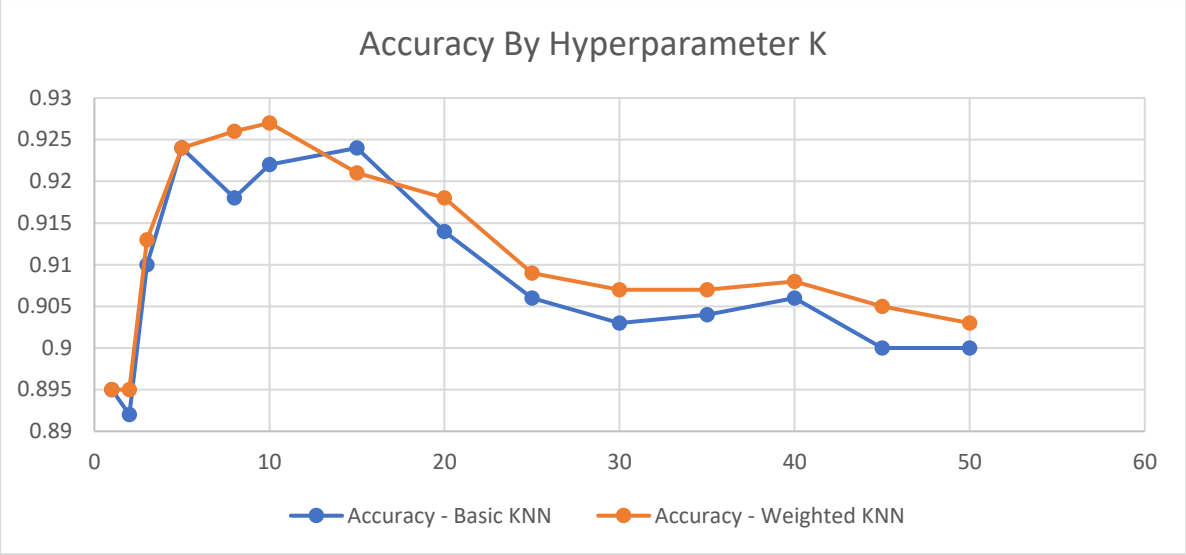
Tuning Hyper Parameters:

There are many ways to tune hyper parameters for a given for a learning algorithm. Techniques such as grid search, random search, Bayesian optimization etc are widely used. We will use simplest technique of grid search, where we run our algorithm for a set of values of hyper parameter and select the parameter which is giving us highest accuracy.

Tuning K

We run basic and weighted KNN for different values of K ([1,2,3,5,8,10,15,20,25,30,25,40,45,50]). When we plot accuracy against K, we see below figure. Based on below we can conclude that best K value for Basic KNN on our data is 5 and Weighted KNN is 10.

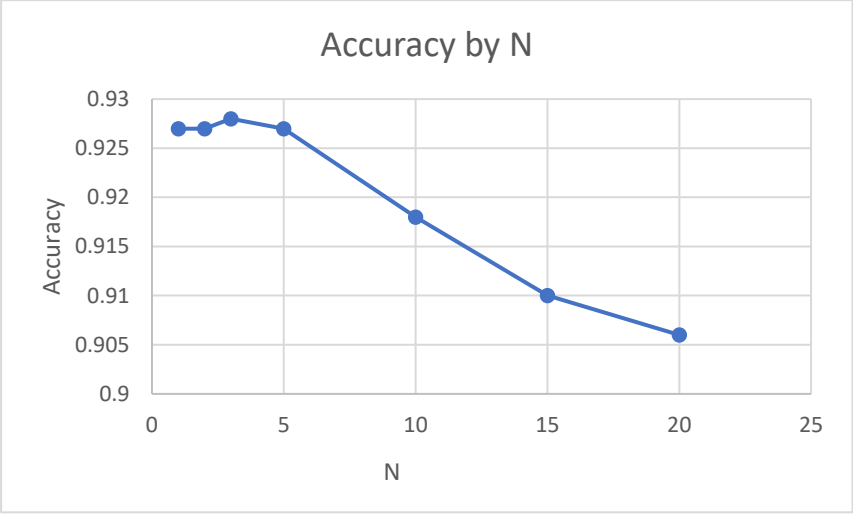
Higher values of K might make the model underfit on the data and lower value of K might make the model overfit the on data. K must be selected in such a way that it results in high accuracy on test data.



Tuning N (Weighted KNN only):

In distance weighted KNN, each neighbour class is decided based on voting of each of its neighbour, where each vote is equal to distance raised to the power of N. N when increased has the effect of giving nearest neighbour instance more weight in voting than instances which are far. We execute our algorithm for different values of N ([1,2,3,5,10,15,20]). We keep K constant at 10.

Plot below shows impact of increasing N on accuracy, we see accuracy slightly higher for N=3 but decreasing with value of N. This because very Higher values of N might have the same effect of reducing K since weights of neighbours that are far would become negligible. From below plot, we can conclude that high accuracy is achieved at K=3.



Distance Metrics

We can use different metrics to measure distance between two features. Below are some of the most popular ones:

- Euclidean Distance
- Manhattan Distance

- Minkowski Distance
- Chebyshev Distance

Implementation of different distance metrics is in the file Part2.2_KNNClassifier_withdistmetrics.py. Program will take distance metric as a parameter and prompts for value. On selection, appropriate distance metric is used. Below table list different distance metrics and accuracy against them. We notice Euclidean distance has the highest accuracy.

Metric	Accuracy
Euclidean	0.927
Manhattan	0.924
Minkowski	0.92
Chebyshev	0.918

In our analysis, we have evaluated one hyper parameter at a time. We can also combine all three hyper parameters into different sets and select the set having the best accuracy. This method is more stable and will work well in cases where parameters are interdependent.

Feature Improvement

Feature Scaling (Normalization)

In a multifeatured data set, it is very likely that each feature has a different scale(range) of values. This might affect many machine learning algorithms especially distance based one's where one feature having higher range can dominate the distance metric. Feature scaling is the technique use to normalize all the feature of data into one small range usually 0-1. Data of KNN classification has a range of 0-1 so this has not been implemented in our program.

Missing Values

Missing data can also affect the performance of KNN severely. Missing data can be NaN or some arbitrary value like -99 or -1 etc which can cause wrong distance calculations. It is important we analyse the data to see which data is missing and take appropriate action. Some of the technique that can be used are:

1) Ignore the row:
   This easy solution can be used if missing data % is negligible. If percentage of rows with missing data is high, then we may lose important data for our model. If a column has mostly missing values, we can also drop the column only if empty/Nan row doesn't mean anything in the context of data. We can also add a new feature to analyse missing data like isNaN (0/1).
2) Replace with Constant:
   This can be used if we can approximate values of missing feature to a default class or value.
3) Replace with mean/median:
   This is most suitable method for missing values in many algorithms. This will ensure that distance metric values don't deviate when values are missing.

Our regression data has no missing data; hence this technique has not been utilized.

Feature Selection

In ML models, irrelevant or unimportant features can affect the performance of the model. Feature selection is method of selecting features which have more impact on output. Feature selection helps in improving performance by removing noise from the data. There are many methods which help to select features:

1) Univariate Analysis: Indicates which features which have strongest relationship with output.
2) Feature Importance: Indicative scoring of each feature according to the value of the feature in deciding the output.
3) Covariance Analysis: Indicates how feature is related to output.

Based on analysis done we can then remove features which have no value in prediction of the output.

Assigning weights to features

We can extend analysis done in feature selection to define weights for each feature and then use them in distance calculation.

Part 3: Problem with Euclidean Distance

In a multifeatured data set, it is very likely that each feature has a different scale(range) of values. This might affect many machine learning algorithms especially distance based one's where one feature having higher range can dominate the distance metric. Euclidean distance is more prone to this problem. This problem can be corrected using feature scaling or normalization.

Feature Scaling (Normalization)

Feature scaling is the technique used to normalize all the feature of data into one small range usually 0-1. By this, no feature would have unequal dominance in the distance metrics.

Below is the range of each feature from our data, as we can see range of each value is slightly if not largely different:

	Min	Max
Feature 1	-4.26	4.05
Feature 2	-4.37	3.69
Feature 3	-3.78	4.04
Feature 4	-3.82	3.52
Feature 5	-4.16	4.19
Feature 6	-3.97	4.22
Feature 7	-3.27	3.65
Feature 8	-3.42	3.45
Feature 9	-3.58	3.64
Feature 10	-3.55	3.44
Feature 11	-3.11	3.38
Feature 12	-4.62	3.65

There are many different techniques to normalize data. We will investigate some of them and we will use SK-Learn library to implement them.

Standard Scaler

Standard scaler standardizes features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False.

Ref: https://scikit-learn.org/StandardScaler

Min-Max Scaler

Transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one. Formula of each feature is given below:

$$X_{inew} = x_i – min(x)/max(x) – min(x)$$

Ref: https://scikit-learn.org/min-max-scaler

Normalizer

The normalizer scales each value by dividing each value by its magnitude in n-dimensional space for n number of features. Say your features were x, y and z Cartesian co-ordinates your scaled value for x would be:

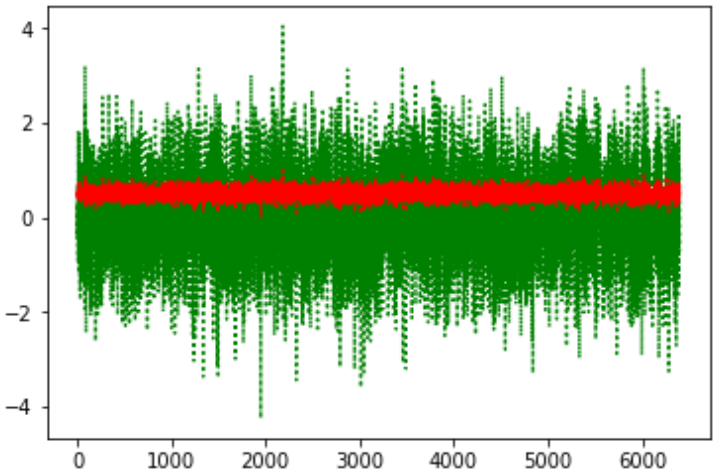
$$x_i / \sqrt{x_i^2 + y_i^2 + z_i^2}$$

Each point is now within 1 unit of the origin on this Cartesian co-ordinate system.

Ref: http://benalexkeen.com/feature-scaling-with-scikit-learn/

Feature scaling has been implemented in file Part3.2_KNNRegression_withscaling.py as part of this assignment. SK-Learn library has been used for scaling. On executing the program, program prompts for scaler to be used. On selecting scaler, it is applied to our data.

Below plot shows feature 1 range, pre and post scaling, from our test data. In green are pre scaled values, we can see that the range is -4 to +4. In red are scaled values using min-max scaler in the range of 0.5 to 1.



Below table shows R2 score against each scalar, keeping K constant at 10. We can see from below that applying Normalizer has resulted in best R2 score.

Scaler	Accuracy
No Scaling	0.8523
Standard Scaler	0.8545
Min-Max Scaler	0.8594
Normalizer	0.8787

Summary

As part of this assignment, we have developed and evaluated many variations of K-Nearest Neighbour Algorithms for regression and calculations. We have performed analysis of Hyper Parameters K, N and distance metric for classification problem and different scalers for regression problem. Based on analysis we can conclude that for classification, KNN with distance metric Euclidean and K=10 has best accuracy for our data set. For regression, KNN at K=10 and with Normalizer provide best R2 score.