Homework 2 : Introduction to Data Mining (CSE 5243)

Ashwini Joshi
09/29/2017

Introduction:

This report contains implementation and analysis of results for k Nearest Neighbour (kNN) algorithm on Income dataset. For a given record in the unseen data, kNN predicts value of the class attribute depending upon the values of class attributes in the k closest matching rows in the training data. Results can be enhanced by varying the value of k and changing the proximity function used for calculating similarity between the records.

Section 1: Approach for kNN

As kNN is a lazy learner, it classifies data at run time rather than actually building a model. The data is trained by giving different inputs of the test data to see which particular set of characteristics (value of k, set of attributes etc.) is optimizing the results. The program outputs two files predicting class attribute with two proximity functions for calculating similarity - Euclidean Distance and General approach. In General approach, Cosine similarity is used for continuous attributes. In homework 1, L1 norm was used in General approach. As the dataset has two sparse columns - capital_gain and capital_loss. Cosine similarity is used for continuous attributes because it gives better results for the sparse data. Values in both the training as well as the test data are normalized to fall in the range of (0, 1) to make sure that all the attributes have equal impact on the proximity function.
The goal is to predict income value of a record in test dataset from the given set of training data. Following are some steps to train and test the kNN model:

1. Value of k: Accuracy is evaluated against some range of values of k to find the optimal value of k for the given dataset. Since the dataset has 288 records, values ranging from 9 to 17 are tested. This range is important as very large or very small values of the data can cause issue of overfitting or underfitting.
2. Proximity Function: kNN is run using both Euclidean Distance and Cosine Similarity measures to compare the accuracy in order to find which one is more suitable for the given dataset. Confusion matrices were observed in order to compare the performance.
3. Posterior Probability: Class values can be predicted using different functions of calculating posterior probabilities. The simple approach of taking majority of the class values is compared with the weighted approach where weight is given to each record according to its distance from the test record. In weighted approach, various functions are tested such as square of the distance, cube of the distance to find which relationship gives improved output.
4. Cross Validation: Random rows are chosen from the training dataset and classifier is run in order to test how the algorithm works on training data.
5. Attributes: If correlation for some attributes is calculated for the class variable of the dataset, it is observed that some attributes have very small value which might act as noise in the data. Hence, algorithm is run considering various combinations of the attributes to decide which attributes to include and which to exclude. From the analysis, it can be said that though excluding some attributes may give improved accuracy in some cases, it does not hold true in general. Therefore, all the attributes are considered while computing similarity.

While trying varied combinations of attributes for the proximity function, it is observed that some rows are exactly matching with zero dissimilarity. It creates a problem while calculating weighted posterior as the distance was zero. To overcome this, if the row in the training data matches exactly with the test record(distance = 0), weight assigned to it is twice the maximum weight in the set of k records of the training data for that particular record.

Section 2: Analysis of Results

This section analyses the results obtained after running the code to examine the impact of different parameters on kNN. In The first part, confusion matrices are built to see how the accuracy and error rates vary as value of k changes. ROC curve are also plotted to see the relationship between True Positive Rate (TPR) and False Positive Rate (FPR).

Following tables (Table 2.1 to 2.6) summarize confusion matrices and error rates for three different values of k with both Cosine and Euclidean Distance. Also, ROC curve is plotted in order to see how well is the classifier classifying the test data.

K = 6	Actual Class				Actual Class		
Predicted Class	Cosine	Class + (<=50K)	Class - (>50K)	Predicted Class	Euclidean	Class + (<=50K)	Class - (>50K)
	Class + (<=50K)	TP (+/+) 195	FP (-/+) 26		Class + (<=50K)	TP(+/+) 189	FP(-/+) 32
	Class - (>50K)	FN (+/-) 44	TN (-/-) 23		Class - (>50K)	FN (+/-) 46	TN(-/-) 21

Section 2: Table 2.1

Calculated Measures (k = 6)		
	Cosine	Euclidean
Accuracy	0.75964	0.72916
Error Rate	0.24305	0.27083
True Positive Rate (Sensitivity/ Recall)	0.81589	0.80425
True Negative Rate (Specificity)	0.46938	0.39622
False Positive Rate	0.53061	0.60377
False Negative Rate	0.18410	0.19574
Precision	0.88235	0.8552
F1 Score	0.84726	0.82894

Section 2: Table 2.2

K = 9	Actual Class				Actual Class		
Predicted Class	Cosine	Class + (<=50K)	Class - (>50K)	Predicted Class	Euclidean	Class + (<=50K)	Class - (>50K)
	Class + (<=50K)	TP (+/+) 192	FP(-/+) 29		Class + (<=50K)	TP (+/+) 188	FP (-/+) 33
	Class - (>50K)	FN (+/-) 48	TN(-/-) 19		Class - (>50K)	FN (+/-) 46	TN (-/-) 21

Section 2: Table 2.3

Calculated Measures (k = 9)		
	Cosine	Euclidean
Accuracy	0.73263	0.72569
Error Rate	0.26736	0.2743
True Positive Rate (Sensitivity/ Recall)	0.8	0.80341
True Negative Rate (Specificity)	0.39583	0.38888
False Positive Rate	0.60416	0.61111
False Negative Rate	0.2	0.19658
Precision	0.86877	0.85067
F1 Score	0.83297	0.82637

Section 2: Table 2.4

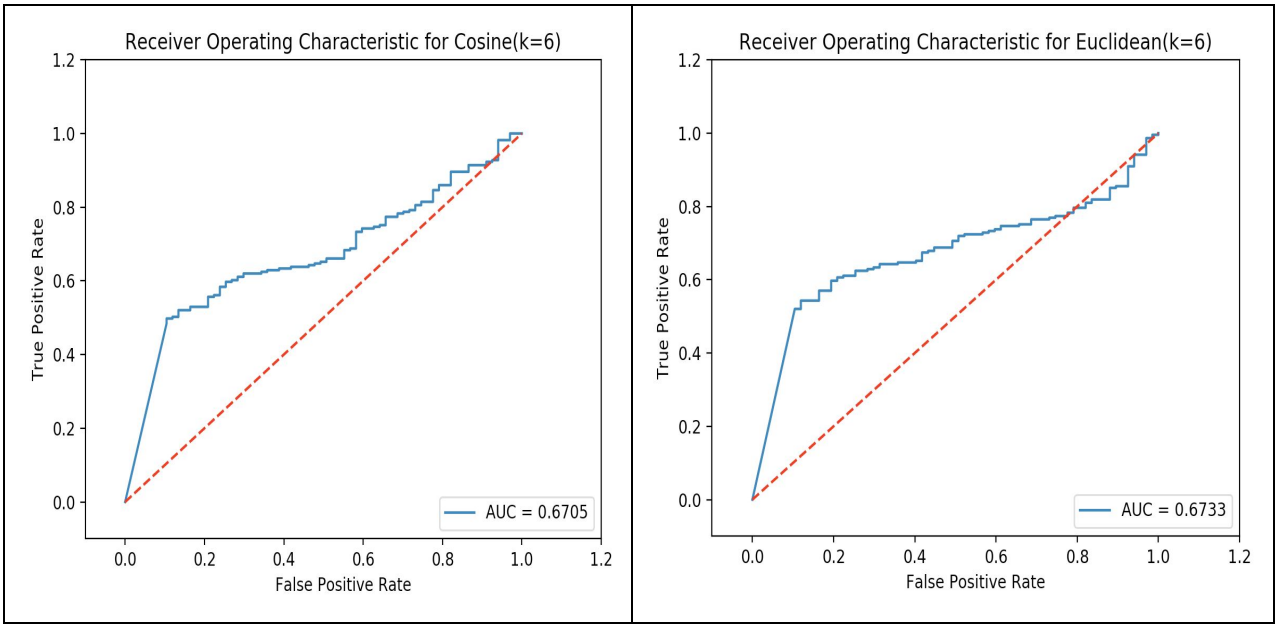
K=11	Actual Class				Actual Class		
Predicted Class	Cosine	Class + (<=50K)	Class - (>50K)	Predicted Class	Euclidean	Class + (<=50K)	Class - (>50K)
	Class + (<=50K)	TP (+/+) 196	FP (-/+) 25		Class + (<=50K)	TP (+/+) 195	FP (-/+) 26
	Class - (>50K)	FN (+/-) 44	TN (-/-) 23		Class - (>50K)	FN (+/-) 45	TN (-/-) 22

Section 2: Table 2.5

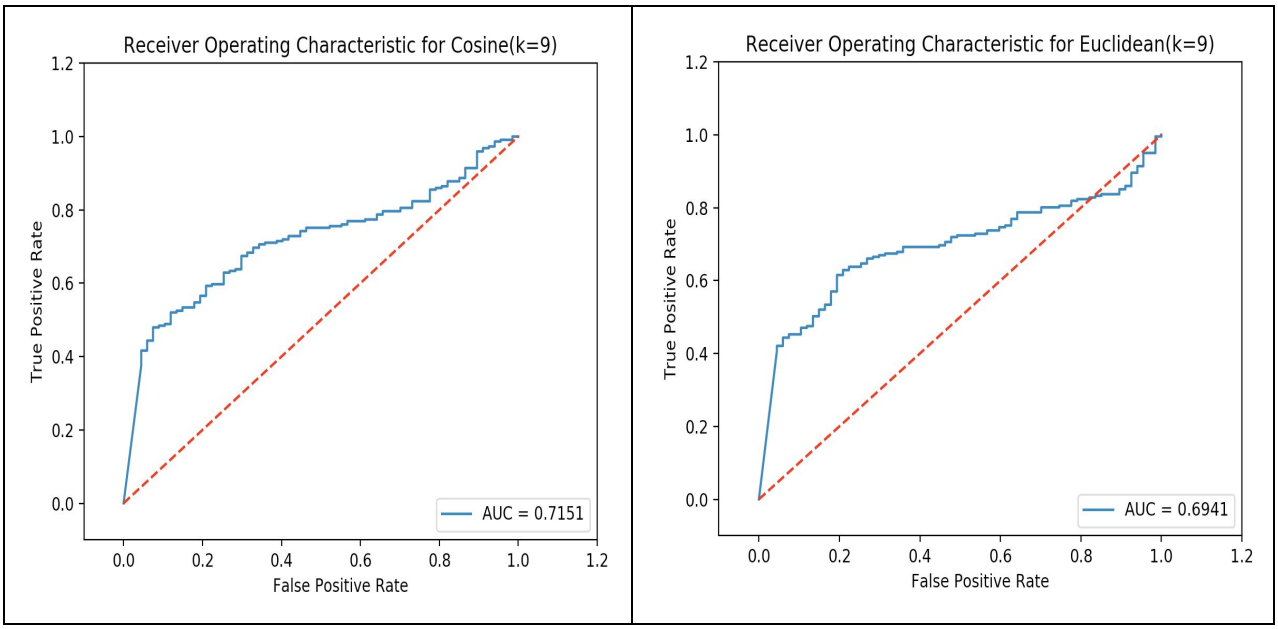
Calculated Measures (k = 11)		
	Cosine	Euclidean
Accuracy	0.76041	0.73611
Error Rate	0.23958	0.26388
True Positive Rate (Sensitivity/ Recall)	0.81666	0.81115
True Negative Rate (Specificity)	0.47916	0.41818
False Positive Rate	0.52083	0.58181
False Negative Rate	0.18333	0.18884
Precision	0.85032	0.83259
F1 Score	0.88687	0.85520

Section 2: Table 2.6

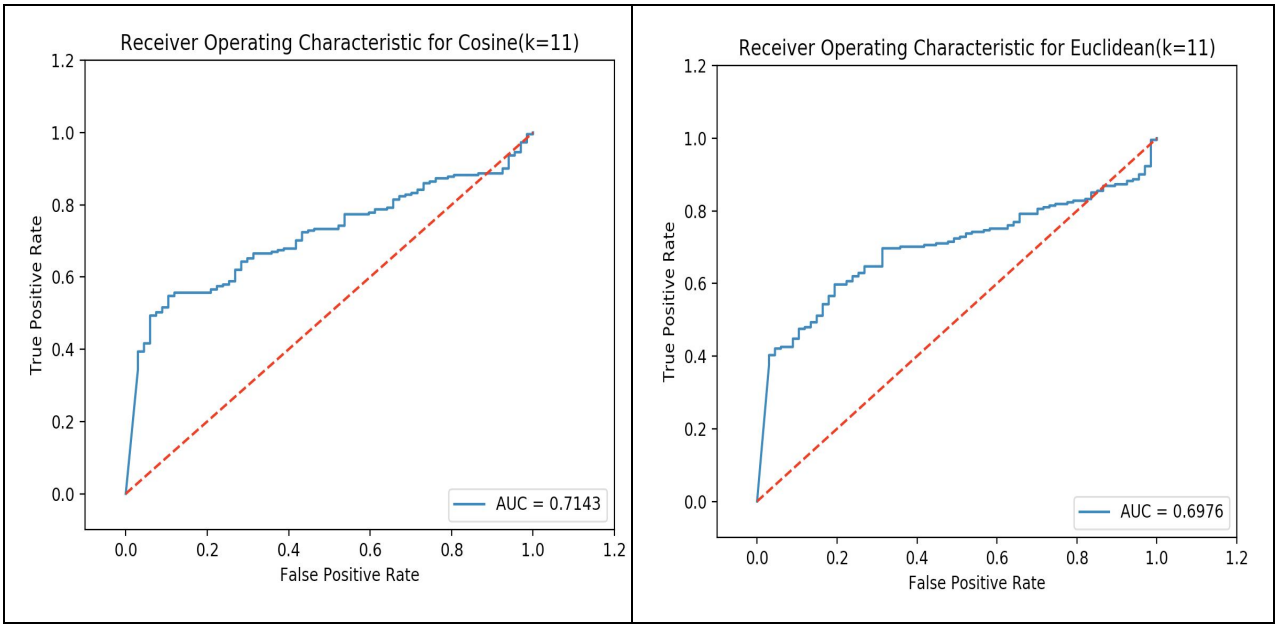
Figure 2.1, 2.2 and 2.3 show ROC curve for Cosine and Euclidean for different values of k:



Section 2: Figure 2.1



Section 2: Table 2.2



Section 2: Table 2.3

Observations:

- From the tables 2.1 to 2.6, it is observed that between three values, value k = 11 is giving the highest accuracy for Cosine and Euclidean Distance. For Euclidean Distance, accuracy is increasing with the increasing value of k from 6 to 11.
- Area under ROC curve is maximum for k = 9 in case of Cosine while k = 11 for Euclidean. For Cosine similarity, though area is greater for value 9, it is comparable with that of k = 11. Also, initial behavior of the curve is better for value 11 because FPR is growing at a slower rate. From the accuracies and ROC curve analysis, both Cosine and Euclidean give the highest accuracy for k = 11 between the three values.

Also, value of F1 measure and true Positive Rate is the highest when k is 11. Hence, value k = 11 is the most suitable value for both the proximity measures.

- Table 2.7 summarizes Accuracies for some values of k in the range 9 to 17. It shows that for Euclidean, accuracy is more for values like k = 15, 17. It shows that proximity function plays a major role in deciding the optimal value of k. But, taking too high values for k might cause the problem of overfitting and it might also result in algorithm being very slow. Hence as a balance between accuracy and speed, k = 13 can also be used as optimal value of k for Euclidean Distance. For Cosine similarity, there is no specific relation with the value of k as the values are fluctuating between 0.73 to 0.74. For k = 11, there is a sharp increase in accuracy to 0.76041. For Euclidean Distance, accuracy is gradually increasing as the value of k increases in this range. It also shows that in general, accuracy of Cosine is greater than that of the Euclidean for the same of value of k in the most cases. Hence, it can be concluded that Cosine Similarity works better for the given dataset rather than Euclidean Distance.

	Accuracy	
Value of k	Cosine	Euclidean
9	0.73263	0.72569
10	0.72222	0.73958
11	0.76041	0.73611
13	0.73958	0.75342
15	0.74305	0.75347
17	0.73263	0.75347

Section 2: Table 2.7

- Table 2.8 shows comparison between weighted and unweighted(majority of votes) posterior for Cosine and Euclidean. For Cosine, the weighted approach is giving better output as compared to unweighted approach. On the contrary, an interesting pattern is observed in unweighted posterior. It accuracy is improving accuracy significantly in case of Euclidean. Hence, for Euclidean Distance, unweighted approach can be used to improve the performance.

	Cosine Accuracy		Euclidean Accuracy	
k	Weighted	Unweighted	Weighted	Unweighted
6	0.75964	0.73263	0.72916	0.73611
9	0.73263	0.73611	0.72569	0.74305
11	0.76041	0.75	0.73611	0.76041
15	0.74305	0.73611	0.75347	0.76388

Section 2: Table 2.8

- Table 2.9 shows the comparison between the accuracies when only 50% of the records are chosen from the training dataset randomly. For values of k=9 and 11, the accuracy increase slightly while for k = 6 accuracy is less.

k	Cosine		Euclidean	
	Complete	50%	Complete	50%
6	0.75964	0.73263	0.72916	0.71527
9	0.73263	0.76041	0.72569	0.73611
11	0.76041	0.77083	0.73611	0.74305

Section 2: Table 2.9

Section 3: Off-the-shelf kNN implementation

This section compares the performance of implemented algorithm with Scikit's inbuilt kNN classifier. Records in the dataset are preprocessed before giving input to the classifier. Only categorical attributes are binarized as continuous variables are transformed automatically by classifier during runtime. Accuracies and other measures are compared with both Cosine and Euclidean. The major difference observed in the execution was speed.
Following table shows accuracies of kNN inbuilt vs implemented algorithm:

- From table 3.1, it is seen that for Cosine similarity accuracy is significantly greater than inbuilt classifier. For Euclidean, kNN is giving slightly better results in case of off-the-shelf implementation.

k	kNN inbuilt	Cosine	Euclidean
6	0.72916	0.75964	0.72916
9	0.73611	0.73263	0.72569
11	0.73611	0.76041	0.73611
15	0.73263	0.74305	0.75347
17	0.73958	0.73263	0.75347

Section 3: Table 3.1

- Table 3.2 compares the confusion matrices for k = 11 for Cosine and inbuilt kNN. It is observed that though the percentage of True Positives in inbuilt is higher, proportion of False Negatives is also much higher as compared to Cosine.

K=11	Actual Class				Actual Class		
Predicted Class	Cosine	Class + (<=50K)	Class - (>50K)	Predicted Class	kNN inbuilt	Class + (<=50K)	Class - (>50K)
	Class + (<=50K)	TP (+/+) 192	FP(-/+) 29		Class + (<=50K)	TP (+/+) 204	FP (-/+) 17
	Class - (>50K)	FN (+/-) 48	TN(-/-) 19		Class - (>50K)	FN (+/-) 59	TN (-/-) 8

Section 3: Table 3.2

- Table 3.3 shows comparison of various measures between implemented and inbuilt classifier. Amongst three, accuracy is the highest for Cosine while Euclidean and inbuilt have the same value. False Positive Rate is the highest in inbuilt. Value of F1 score for inbuilt indicates that its performance is slightly better as compared to implemented Cosine and Euclidean algorithm.

Calculated Measures (k = 11)			
	Cosine	Euclidean	kNN inbuilt
Accuracy	0.76041	0.73611	0.73611
Error Rate	0.23958	0.26388	0.26388
True Positive Rate (Sensitivity/ Recall)	0.81666	0.81115	0.77566
True Negative Rate (Specificity)	0.47916	0.41818	0.32
False Positive Rate	0.52083	0.58181	0.68
False Negative Rate	0.18333	0.18884	0.22433
Precision	0.85032	0.83259	0.84297
F1 Score	0.88687	0.85520	0.92307

Section 3: Table 3.3