K Nearest Neighbors

Muhammad Banatwalla

September 27, 2020

1 Introduction

In this assignment, we examine the performance of the K-Nearest Neighbor(KNN) algorithm on classification of fonts. Through this algorithm and the chosen data, we try to determine the best choice for k. Further, we then examine the accuracy on this chosen best k. Then, we use weights on the features of the data and compare the result against the unweighted data.

2 Preliminary Treatment of Data Set

Our data comes from the ”University of California Irvine Repository for Machine Learning Datasets.” The data we are working with consists of digitized images of typed characters. We chose 3 fonts: Cambria, Times, and Baskerville. Each image is described by a series of gray level image intensity values for different pixel positions. For this report, Cambria will correspond to CL1, Times will correspond to CL2 and Baskerville will correspond to CL3. In each of these classes, it only contains what is considered a ”normal” character so all of these images have strength = .4 and italic = 0. The CL1,CL2, and CL3 were unioned into a larger data set we call DATA. Further, we then standardized DATA to get a new set called SDATA.

3 Correlation

Using the standardized data set SDATA, we computed a correlation matrix on the gray level image intensity values for the pixel positions. Below are the top 10 highest correlation values.

X	Y	Correlation
r19c1	r19r17	.9173

6		
r19c0	r19c1	.9013
r19c18	r19c19	.9073
r19c2	r19c3	.9050
r19c17	r19c18	.9038
r0c0	r0c1	.9032
r9c2	r10c2	.9013
r12c2	r13c2	.9009
r19c9	r19c10	.9002
r12c1	r13c1	.8980

Table 1: Top 10 Correlation Values

The X and Y columns correspond to the pixel positions that have the correlation value in the correlation row. The highest correlation is between r19c16 and r19c17 and is .9173.

4 KNN

In this section, we used the knn() function of R with various k values. We used a train set made up of about 80% of each of the CL1,CL2,CL3 and the test set was made up of the other 20% of CL1, CL2, and CL3. These portions of each of the classes were chosen arbitrarily. The first value of k chosen was k = 12. The results are below:

$$trainperf_{12} = 79.02\%$$

$$testperf_{12} = 71.92\%$$

The results for this choice of k are not at the percentage that we would like so therefore it is necessary to try different values of k. The KNN process was then repeated for k = 5,10,15,20,30,40,50,100.

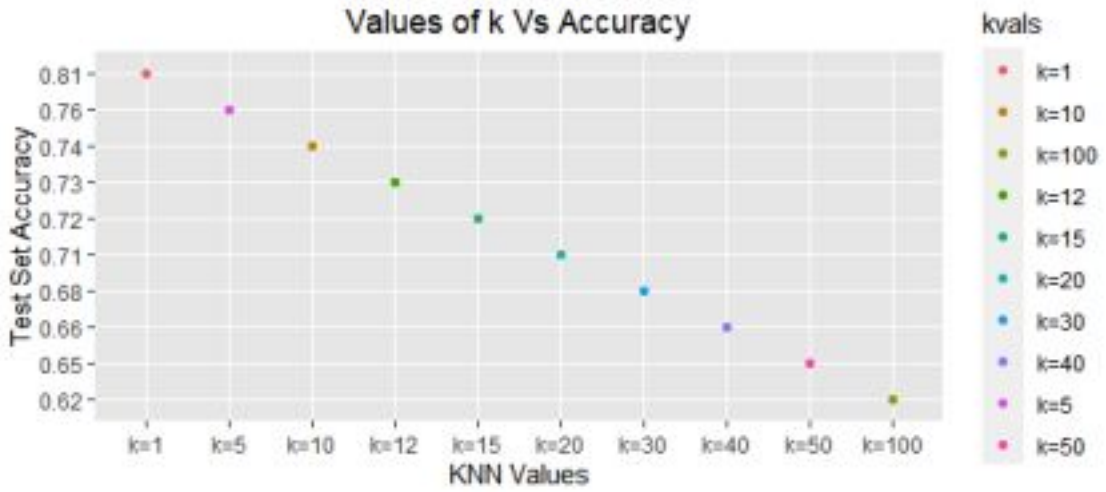
k	$trainperf_k$	$testperf_k$
1	80.0%	99.32%

5	76.29%	85.02%
10	74.17%	79.73%
15	72.16%	77.25%
20	71.38%	75.15%
30	68.34%	71.48%
40	66.40%	68.72%
50	65.25%	66.50%
100	62.10%	62.61%

Table 2: *T rainperf$_k$andT estperf$_k$*

2

We then plotted the k values against the corresponding *testperf$_k$* percentage to identify the best range of values for the integer k.



As is apparent by the graph our two best values are between k= 1 and k = 5, this might be due to a huge difference in the number of observations between our selected fonts.We then explored the knn for more values within this range as shown in the table below:

k	*testperf$_k$*	*trainperf$_k$*
1	80.0%	99.32%

2	79.14%	88.84%
3	76.47%	88.49%
4	75.99%	85.40%
5	76.29%	85.02%

Table 3: $T\ rainperf_k and T\ estperf_k$

The best performance for our k values was at k = 1. We now look at the confusion matrices for $trainperf_1$ and $testperf_1$:

	Predicted: BASKERVILLE	Predicted: CAMBRIA	Predicted: TIMES
Actual: BASKERVILLE	95.83%	3.13%	1.04%
Actual: CAMBRIA	.0%	100%	0%
Actual: TIMES	0%	.96%	99.04%

Table 4: Confusion Matrix for k = 1 (*T rainingData*)

	Predicted: BASKERVILLE	Predicted: CAMBRIA	Predicted: TIMES
Actual: BASKERVILLE	27.66%	19.15%	53.2%
Actual: CAMBRIA	1.09%	72.85%	26.05%
Actual: TIMES	1.46%	11.24%	87.30%

Table 5: Confusion Matrix for k = 1 (*T estingData*)

On the training data for k=1, the performance is very accurate. Specifically, the CAM BRIA font was perfectly classified. On the testing data for k = 1, the performance is not as good. We believe the reason for the poor performance is the comparatively low amount of observations from the BASKERVILEE. The BASKERVILLE specifically did not perform well, but overall the testperf for k = 1 was highest for this k value.

5 Confidence Intervals

We examined the confidence intervals for the 3 diagonal terms of the confusion matrix

for k = 1 for testing and training data.

Font	Confidence Int. for Train	Confidence Int. for Test
BASKERVILLE	(91.65%, 97.93%)	(14.87%, 40.45%)
CAMBRIA	(100%, 100%)	(69.41%, 76.30%)
TIMES	(98.85%, 99.43%)	(85.20%, 89.41%)

Table 6: Confidence Intervals for k =1 Confusion Matrices

Examining the differences between these intervals, we can see that none of these intervals between train and test, overlap. This means that there is a significant difference in the performance of the train and test set.Although, we can see that specifically, the baskerville font intervals are very far from each other. Ideally, we do not want to see such a big difference between training and testing set. However, a small difference is expected. We could improve this performance by training on a larger dataset or making sure we have the same amount of data for each of the fonts.

6 Weighted Distances

In order to create weights for the data's features, we narrowed down the amount of features for the training and testing sets to run. For the first run, we used only the features with names rLcM where L = 0,1,2,...,9 and M = 0,1,2,...,9 and called this data set PACK1. We then repeated this process using L = 0,1,2,...,9 and M = 10,11,...,19 and called this PACK2, and formed PACK3 with L = 10,11,...,19 and M = 10,11,...,19, and formed PACK4 with L = 10,11,...,19 and M = 0,1,2,...,9. These features correspond to the 100 pixel intensities displayed displayed in a specific 10x10 window of our 20x20 pixels images. We then ran the KNN algorithm using these data sets with k = 1. The results for $testperf_1$ for each of these data sets is below:

Data Set	$testperf_1$
PACK1	81.5%
PACK2	80.35%

PACK3	80.29%
PACK4	82.23%

Table 7: $testperf_1$ for PACK Data

We can observe from the table above that the difference in weights is small and so this implies that each of the data sets have an approximately equal affect on our response variable. We then used these $testperf_1$ values to add weights to the data. Using the $testperf_1$ we weighted the features that correspond to the features in the data set. After computing the weights, we ran the knn process on the weighted data:

$testperf_{weighted} = 80.59\%$

	Predicted: BASKERVILLE	Predicted: CAMBRIA	Predicted: TIMES
Actual: BASKERVILLE	29.78%	17.02%	53.2%
Actual: CAMBRIA	2.03%	75.04%	26.05%
Actual: TIMES	1.25%	11.97%	86.78%

Table 8: Confusion Matrix for k = 1, Weighted Data (*T estingData*)

The performance on our weighted data is similar to the non-weighted data as our weights were very close together. Additionally, because we chose our columns for the PACK data arbitrarily, it is possible that a few columns had a big impact on data but were canceled out by other columns in other PACKs. To get a better understanding we could study these columns to see which will have a bigger impact on our data and put them separately to increase their weights and hence improve the effectiveness of our predictions. Furthermore, when our weights were very close we should have run a few more tests to try to find a combination of selections where our weights would become more meaningful. If we wanted to try to increase our overall performance we could try training on a data set that has an equal number of each of the fonts and increasing the size of the data set.