SYDDANSK UNIVERSITET

K-nearest neighbor

Bjarki PÃąll
Abdulrahman Abdulrahim
Miguel de la Colina

February 20, 2017

ABSTRACT

THE PURPOSE OF THIS REPORT IS TO USE THE K-NEAREST NEIGHBOR ALGORITHM ON A DATA-SET MADE OF CIPHERS WE WILL BE DIVIDING THE DATA IN TWO SETS ONE FOR TRAINING AND ONE FOR TESTING AND WE WILL BE ANALYZING THE RESULTS WITH MULTIPLE K AND DPI'S TO SEE HOW THIS ALTERS OUR RESULTS. ALSO WE WILL BE APPLYING THE GAUSSIAN SMOOTHING WITH VARIOUS SIGMAS IN ORDER TO SEE HOW DOES THIS ALTER THE RESULTS.

1 K-NEAREST NEIGHBOR

We will be using R which is a statistical language in order to test the k-nearest neighbor algorithm, firstly we will have to generate our data-sets which are taken from scanned ciphers and loaded through the function loadSinglePersonsData, once the data is loaded we shuffle the data with a seed for reproducible results with this done we split the data into test and train so that we are able to test the data after we have trained and be able to use different data from the one that was trained.

For this test we will be varying the number of k to see how the result actually varies when we begin to change it's value we will check how the speed and test recognition are affected by this change. This is to see how important really is to select the correct k and to see if having selected the wrong one could affect your results substantially.

We will also be doing cross validation of the results in order to see if the results of the trained model will fit for other hypothetical, set of data this will be done by running 10 times a 90%/10% split of the data-set.

```
M_xval <- list()
for (i in 1:10) {
  # Split matrix into 10 parts
  M_xval[[i]] <- M_shuffled[((i-1)*nrow(M)/10+1):(i*nrow(M)/10),]
}
for (i in 1:10) {
  # Recombine 9 parts for training and keep 1 for testing
  M_xval_test <- M_xval[[i]]
  M_xval_train <- do.call(rbind, M_xval[-i])
  true_class_xval <- M_xval_train[,1]
  class_xval = knn(M_xval_train, M_xval_test, true_class_xval, k_it)
  true_class_xval <- factor(true_class_xval, levels(class_xval))
  success_xval <- sum(true_class_xval == class_xval)/length(class_xval)
  cat("Result", i, ":", success_xval, "\n")
}
```

Finally after testing it with the smoothing implementation that was in the loadImage file we have to implement the smoothing using a different method. We used the Gaussian smoothing with various sigmas, for the implementation we used the R function gblur which receives as parameter the image and the sigma.

RESULTS

RESULTS WITH DPI=100

DPI=100			
K	Training Set	Test Set	Time
1	1	0.9995	3.853
10	0.9945	0.9945	3.334
25	0.991	0.986	3.386
50	0.984	0.986	3.395
100	0.9865	0.989	3.807

Results with DPI=200

DPI=200			
K	Training Set	Test Set	Time
1	1	0.9995	3.419
10	0.9945	0.9945	4.170
25	0.991	0.986	3.393
50	0.984	0.986	3.419
100	0.9865	0.989	3.58

Results with DPI=300

DPI=300			
K	Training Set	Test Set	Time
1	1	0.9995	5.011
10	0.9945	0.9945	3.484
25	0.991	0.9875	4.956
50	0.984	0.986	3.779
100	0.9865	0.989	4.087

As we see the results we can see that in the change of DPI the training set and test set results do not vary that much between them but we can see a variation when the k is being altered with k= 1 being the best result for the training set and also for the test set.

Something else we noticed was that the higher the DPI even if the results didn't change the time did increase we think because with more DPI it takes more time to process because there are more pixels to be taken into account.

Cross-validation k=50 DPI=100

DPI=100 k=50	
Number	Result
Result 1	0.8675
Result 2	0.9225
Result 3	0.8525
Result 4	0.9525
Result 5	0.915
Result 6	0.8275
Result 7	0.82
Result 8	0.7975
Result 9	0.9525
Result 10	0.8875

mean: 0.8795
standard deviation: 0.055450178

Cross-validation k=50 DPI=200

DPI=100 k=50	
Number	Result
Result 1	0.8675
Result 2	0.9225
Result 3	0.8525
Result 4	0.9525
Result 5	0.915
Result 6	0.8275
Result 7	0.82
Result 8	0.7975
Result 9	0.9525
Result 10	0.8875

mean: 0.8795
standard deviation: 0.055450178

Cross-validation k=50 DPI=300

DPI=100 k=50	
Number	Result
Result 1	0.8675
Result 2	0.9225
Result 3	0.8525
Result 4	0.9525
Result 5	0.915
Result 6	0.8275
Result 7	0.82
Result 8	0.7975
Result 9	0.9525
Result 10	0.8875

mean: 0.8795
standard deviation: 0.055450178

Results for Gaussian Smoothing

DPI=100			
sigma	Training Set	Test Set	Time
.1	0.6405	0.6235	66.464
.5	0.7235	0.702	66.900
1	0.8215	0.828	66.404
2	0.932	0.941	66.231
5	0.9895	0.9915	67.592
10	0.9985	1	66.365