Assignment 2 - Handwriting Recognition

Christopher Graham

November 11, 2015

Problem

We are attempting to set up an algorithm that predicts a handwritten digit based on a 4x4 bmp, where each pixel is represented by a 0-255 grey scale value.

We are provided with a training set of 7493 observations and a test set of 3497 observations.

For this analysis we will be using a knn algorithm to identify digits.

The data is provided in a cleaned format, with all pixel data using an identical scale. Thus no pre-processing is necessary and we can proceed directly to knn implementation.

```
# Import libraries and data
require(RWeka)
require(caret)
require(e1071)
test_set <- read.csv('pendigits.tes.csv')
train_set <- read.csv('pendigits.tra.csv')
```

Model Development - choosing a value for k

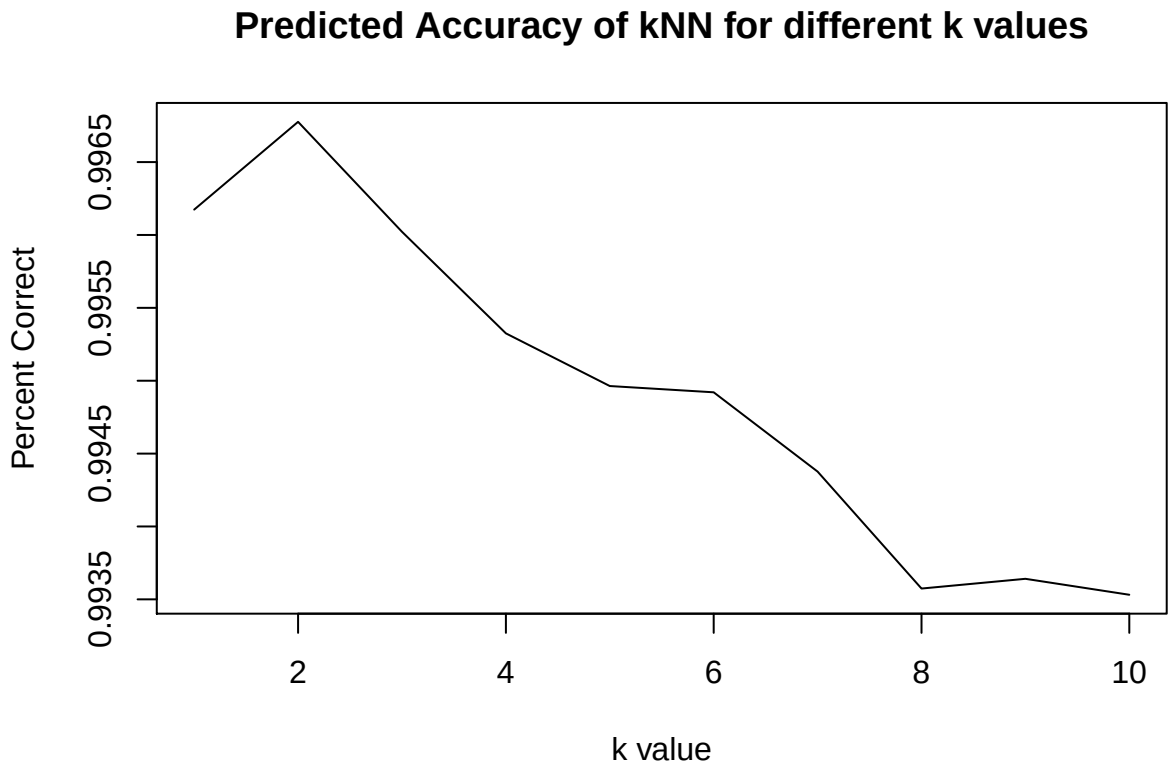
The first thing to do is to figure out the k-value that optimizes the predictive value of our model. In order to avoid any possible risk of over-fitting with our test data, this part of the analysis will be conducted solely on the training data, using k-fold cross-validation to determine our optimal k value. Distance is calculated as Euclidian distance.

```
correct_pct <- data.frame(k = NULL, pctCorrect = NULL)
for (k in 1:10) {
    set.seed(2465)
    classifier <- IBk(X8~., data = train_set,
                  control = Weka_control(K = k))
    eval_data <- evaluate_Weka_classifier(classifier, numFolds = 10)
    correct_pct[k,1] <- k
    correct_pct[k,2] <- eval_data[[2]][1]
}
colnames(correct_pct) <- c('k', 'pct_correct')
correct_pct
```

```
##     k pct_correct
## 1   1   0.9961740
## 2   2   0.9967759
## 3   3   0.9960218
## 4   4   0.9953248
## 5   5   0.9949636
## 6   6   0.9949205
```

```
## 7   7   0.9943766
## 8   8   0.9935742
## 9   9   0.9936409
## 10 10   0.9935316
```

```
plot(correct_pct$k, correct_pct$pct_correct, type='l',
     ylab = "Percent Correct",
     xlab = 'k value',
     main = 'Predicted Accuracy of kNN for different k values')
```



Testing and Evaluating Model

While there is minimal variation in the predictive quality, this analysis suggests an optimal result with k=2.

Based on these results, we now apply the model to our test data set to see how it actually performs.

Note, that for this part of the analysis, we've decided to use the knn function from the class package just for a change of pace. This algorithm also uses Euclidian distance.

```r
require(class)
test_pred <- as.integer(as.character(knn(train_set, test_set,
                                         train_set$X8, 2)))
conf_matrix <- confusionMatrix(test_pred, test_set$X8)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7   8   9
##          0 352   0   0   0   0   0   0   0   0   0
##          1   0 352   3   3   0   0   0  11   0   2
##          2   0  10 361   0   0   0   0   1   0   0
##          3   0   0   0 332   0   7   0   0   0   3
##          4   0   1   0   0 356   0   0   0   0   0
##          5   0   0   0   0   8 325   0   0   2   6
##          6   6   0   0   0   0   0 336   0   0   0
##          7   0   1   0   0   0   0   0 350   0   4
##          8   5   0   0   0   0   0   0   2 333   1
##          9   0   0   0   1   0   3   0   0   0 320
##
## Overall Statistics
##
##                Accuracy : 0.9771
##                  95% CI : (0.9716, 0.9818)
##     No Information Rate : 0.1041
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9746
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9697   0.9670   0.9918  0.98810   0.9780  0.97015
## Specificity            1.0000   0.9939   0.9965  0.99684   0.9997  0.99494
## Pos Pred Value         1.0000   0.9488   0.9704  0.97076   0.9972  0.95308
## Neg Pred Value         0.9965   0.9962   0.9990  0.99873   0.9975  0.99683
## Prevalence             0.1038   0.1041   0.1041  0.09608   0.1041  0.09580
## Detection Rate         0.1007   0.1007   0.1032  0.09494   0.1018  0.09294
## Detection Prevalence   0.1007   0.1061   0.1064  0.09780   0.1021  0.09751
## Balanced Accuracy      0.9848   0.9805   0.9941  0.99247   0.9889  0.98254
##                      Class: 6 Class: 7 Class: 8 Class: 9
## Sensitivity           1.00000   0.9615  0.99403  0.95238
## Specificity           0.99810   0.9984  0.99747  0.99873
## Pos Pred Value        0.98246   0.9859  0.97654  0.98765
## Neg Pred Value        1.00000   0.9955  0.99937  0.99496
## Prevalence            0.09608   0.1041  0.09580  0.09608
## Detection Rate        0.09608   0.1001  0.09522  0.09151
## Detection Prevalence  0.09780   0.1015  0.09751  0.09265
## Balanced Accuracy     0.99905   0.9800  0.99575  0.97556
```

Conclusions

When applying the model to test data, we see a small drop in overall predictive accuracy (from 0.997 to 0.977) for the model. A drop-off in prediction is completely normal in moving from a model's training to test set. This is especially true when we consider that the digits in the training set were written by a different set of individuals than the digits in the test set, indicating that our model has likely been somewhat over-fit to the peculiarities of the handwriting of the individuals who contributed to the training data. (Details an the original data set can be foundat: http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits)

It's also kind of interesting to look at how the model performed in terms of predictive ability for different digits. The model is best a predicting the digit 6 (specificity = 1) and worst at predicting 9 (specificity = 0.952)

Overall, the results are consistent with the knn results posted on the MNIST database of handwritten digits (http://yann.lecun.com/exdb/mnist/). While other models (notably: convolutional nets and neural nets) seem to offer better error rates, the results provided by a simple knn analysis are fairly strong.

Note, however, that this predictive value is for digital images generated in a laboratory experiment in which individuals were instructed to write a single digit inside a box on a pressure-sensitive tablet. We can anticipate that in actual OCR applications of normal handwritten digits, our predictive performance would be worse.