

Handwriting Recognition Using KNN

Andresa de Andrade

March 26, 2015

Introduction

In this document the methodology KNN is applied in order to find patterns in the hand writing of numbers from 0-9 and try to predict them using these groups.

In this aspiration we have challenges such as: - each person has their own way of writing number; - some numbers are very similar to others, like 1 and 7, or 6, 8 and 9. - find the better number of neighborhood in order to optimize the algorithm.

Data Explanation

In order to apply this algorithm we have each number divided in a matrix 4 x4 resulting in 16 variables presented as columns in our data set. The last column is the number class.

For this project we have two files, one for training and one for testing. The testing data set contains 3497 samples and the training has 7493.

KNN

So training our model to use k=5, our confusion matrix has too many 7 classified as number 1 and too many 6 as number 0.

```
## Warning: package 'RWeka' was built under R version 3.1.2
```

```
##
## === Summary ===
##
## Correctly Classified Instances      3414          97.5986 %
## Incorrectly Classified Instances    84           2.4014 %
## Kappa statistic                     0.9733
## Mean absolute error                 0.0062
## Root mean squared error            0.0621
## Relative absolute error             3.4262 %
## Root relative squared error        20.7078 %
## Coverage of cases (0.95 level)     98.8279 %
## Mean rel. region size (0.95 level) 10.6175 %
## Total Number of Instances          3498
##
## === Confusion Matrix ===
##
##      a   b   c   d   e   f   g   h   i   j   <-- classified as
## 354    0    0    0    0    0    7    0    2    0 |  a = 0
##    0 347   15    0    1    0    0    1    0    0 |  b = 1
##    0    2 362    0    0    0    0    0    0    0 |  c = 2
##    0    1    0 333    0    0    0    0    0    2 |  d = 3
##    0    0    0    0 354    9    1    0    0    0 |  e = 4
##    0    0    0    6    0 328    0    0    0    1 |  f = 5
##    0    0    0    0    0    0 336    0    0    0 |  g = 6
##    0   15    1    0    0    0    0 347    1    0 |  h = 7
##    1    0    0    0    0    1    0    0 334    0 |  i = 8
##    0    2    0    9    0    1    0    4    1 319 |  j = 9
```

So we have a accuracy of 97.59% of the cases, and in this project we need to find the best K for set of data. Running the same algorithm for other cases we found that k=1 has the best Correctly Classified Instances:

```
##
## === Summary ===
##
## Correctly Classified Instances      3419      97.7416 %
## Incorrectly Classified Instances    79      2.2584 %
## Kappa statistic                     0.9749
## Mean absolute error                 0.0048
## Root mean squared error            0.0672
## Relative absolute error            2.6397 %
## Root relative squared error        22.39 %
## Coverage of cases (0.95 level)     97.7416 %
## Mean rel. region size (0.95 level) 10 %
## Total Number of Instances          3498
##
## === Confusion Matrix ===
##
##      a   b   c   d   e   f   g   h   i   j   <-- classified as
## 354   0   0   0   0   0   6   0   2   1 | a = 0
##   0 349  13   0   1   0   0   1   0   0 | b = 1
##   0   2 362   0   0   0   0   0   0   0 | c = 2
##   0   2   0 333   0   0   0   0   0   1 | d = 3
##   0   0   0   0 355   8   0   0   0   1 | e = 4
##   0   0   0   5   0 325   0   1   0   4 | f = 5
##   0   0   0   0   0   0 336   0   0   0 | g = 6
##   0  10   1   3   0   0   1 348   1   0 | h = 7
##   0   0   0   0   0   1   0   0 335   0 | i = 8
##   0   2   0   3   0   5   0   3   1 322 | j = 9
```

So running the algorithm for k=1 we have a percentage of accuracy of 97.74% of the cases. We also reduce the cases that 7 and 1 have gotten mislabeled.