PEC1: Algorithm K-NN

Laura Sudupe Medinilla

5 de abril, 2021

Contents

The kNN Algorithm

In this markdown we are going to learn about classification using k-NN. Unlike many classification algorithms, k-NN does not do any learning. It simply stores the training data verbatim. Unlabeled test examples are then matched to the most similar records in the training set usin a distance function, and the unlabeled example is assignaed the label of its neighbors.

The strengths and weaknesses of this algorithm are as follows:

Strengths	**Weaknesses**
* Simple and effective	* Does not produce a model, which limits the ability to find novel insights in relationships among features
* Makes no assumptions about the underlying data distribution	* Slow classification phase
* Fast training phase	* Requires a large amount of memory * Nominal features and missing data require additional processing

Step 1 - Collecting data

We will utilize the "Protein Secondary Structure" from the Brookhaven National Laboratory (USA).This data includes the protein secondary structures from 101 representative proteins.

Step 2 - Exploring and preparing the data

We are going to take a look to our data with `str(amino)`.

```
'data.frame':   10000 obs. of  18 variables:
 $ V1 : chr  "S" "L" "L" "L" ...
 $ V2 : chr  "S" "L" "L" "L" ...
 $ V3 : chr  "P" "Q" "N" "N" ...
 $ V4 : chr  "F" "Y" "P" "A" ...
 $ V5 : chr  "S" "Y" "K" "K" ...
 $ V6 : chr  "Q" "G" "K" "G" ...
  [list output truncated]
```

The data is structured with 10000 sequences with 18 amino acids. We want to see which is the central amino acid structure type. The `table()` output indicates that 5557 aminos has **coil** structure, 1935 has **beta-sheet** structure and 2508 has **alpha-helix** structure.

```
table(amino$V18)
```

```
   _    e    h
5557 1935 2508
```

We can label the data and check the structure porcentages with `prop.table()`

```
amino$V18 <- factor(amino$V18, levels = c("_", "e", "h"),
              labels=c("coil","beta-sheet","alpha-sheet"))
round(prop.table(table(amino$V18)) * 100, digits = 1)
```

```
       coil  beta-sheet alpha-sheet
       55.6        19.4        25.1
```

Data preparation

Creating training and test datasets

We are going to divide our data into two portions: a training dataset that will be used to build the KNN model and a test dataset that will be used to estimate the predictive accuracy of the model.

```
data <- cbind(amino_one_hot, amino[,ncol(amino)])
colnames(data)[ncol(data)] <- 'V341'

set.seed(123)

train_indx <- sample(x= 1:nrow(data),
```

```
                    size= 0.67*nrow(data),
                    replace=FALSE)

train_data <- data[train_indx,-ncol(data)]
test_data <- data[-train_indx,-ncol(data)]

train_label <- data[train_indx, ncol(data)]
test_label <- data[-train_indx, ncol(data)]
```

Step 3 - Training a model on the data

For the classification, we will use a kNN implementation from the `class` package. We use the `knn()` function to classify the test data

```
amino_test_pred <- knn(train= train_data, test= test_data,
                       cl=train_label, k=21)
```

The `knn()` function returns a factor vector of predicted labels for each of the examples in the test dataset, which we have assigned to `amino_test_pred`.

Step 4 - Evaluating model performance

We need to evaluate how well the predicted classes in the `amino_test_pred` vector match up with the known values in the `test_label` vector. To these purpose, we can use the `CrossTable()` function. We will create a cross tabulation indicating the agreement between two vectors.

```
CrossTable(x=test_label, y=amino_test_pred, prop.chisq=FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3300


             | amino_test_pred
  test_label |        coil |  beta-sheet | alpha-sheet |   Row Total |
-------------|-------------|-------------|-------------|-------------|
        coil |        1739 |          29 |          53 |        1821 |
             |       0.955 |       0.016 |       0.029 |       0.552 |
             |       0.589 |       0.271 |       0.218 |             |
             |       0.527 |       0.009 |       0.016 |             |
-------------|-------------|-------------|-------------|-------------|
```

```
  beta-sheet |         544 |          64 |          38 |         646 |
             |       0.842 |       0.099 |       0.059 |       0.196 |
             |       0.184 |       0.598 |       0.156 |             |
             |       0.165 |       0.019 |       0.012 |             |
-------------|-------------|-------------|-------------|-------------|
 alpha-sheet |         667 |          14 |         152 |         833 |
             |       0.801 |       0.017 |       0.182 |       0.252 |
             |       0.226 |       0.131 |       0.626 |             |
             |       0.202 |       0.004 |       0.046 |             |
-------------|-------------|-------------|-------------|-------------|
Column Total |        2950 |         107 |         243 |        3300 |
             |       0.894 |       0.032 |       0.074 |             |
-------------|-------------|-------------|-------------|-------------|
```

The cell percentage in the table indicate the portion of values that fall into four categories. In the top-left, these of , cell we have the number of values the KNN algorithm correctly identified as coil. In the diagonal we have the true posiive values, for example, these of , shows values the KNN correctly identified as alpha-sheet. In the case of examples are false negatives, the predicted value were coil but the structure were beta-sheet. The same happens with of the predicted values were coil but the structure were alpha-sheet.

On the other side, a total of out of ‘ were incorrectly classified by the KNN algorithm like alpha-sheet.

Step 5 - Improving the model performance

We will try several different values for **k**, (k = 1, 3, 5, 7, 11), to examine the performance. Using the previous test and training datasets, the same records were classified. The number of false negatives and false positives are shown for each iteration.

k value	# false negatives	# false positives	% classified Incorrectly
1	169	115	8.606061
3	292	135	12.939394
5	349	131	14.545454
7	395	110	15.303030
11	449	85	16.181818

The 11NN aproach was able yo avoid a lot of false positives at the expense of adding false negatives. In these cases, the 1NN aproach has the higher porcentage os classified incorrectly.

Let’s see some other values

```
confusionMatrix(amino_test_pred, test_label, positive = "coil" )
```

```
Confusion Matrix and Statistics

             Reference
Prediction    coil beta-sheet alpha-sheet
  coil        1630        449         556
  beta-sheet    85        143          43
  alpha-sheet  106         54         234
```

```
Overall Statistics

               Accuracy : 0.6082
                 95% CI : (0.5913, 0.6249)
    No Information Rate : 0.5518
    P-Value [Acc > NIR] : 3.392e-11

                  Kappa : 0.2365

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: coil Class: beta-sheet Class: alpha-sheet
Sensitivity               0.8951           0.22136            0.28091
Specificity               0.3205           0.95177            0.93514
Pos Pred Value            0.6186           0.52768            0.59391
Neg Pred Value            0.7128           0.83394            0.79387
Prevalence                0.5518           0.19576            0.25242
Detection Rate            0.4939           0.04333            0.07091
Detection Prevalence      0.7985           0.08212            0.11939
Balanced Accuracy         0.6078           0.58657            0.60803
```

The kappa value adjusts accuracy by acounting for the possibility of a correct prediction by chance alone. We have a fair agreement kappa value, and big error rate, these indicated us the proportion of the incorrectly classified examples.

Step 7 - Prove our KNN classifier with different labels

We know the alpha-helix and beta-sheet are both non-coil structures. We are going to performe the same algorithm for these labels.

```
data_coil <- cbind(amino_one_hot, amino[,ncol(amino)])
colnames(data_coil)[ncol(data_coil)] <- 'V341'

data_coil$V341 <- factor(data_coil$V341, levels = c( "beta-sheet", "alpha-sheet", "coil"),
              labels=c("no-coil","no-coil","coil"))
```

```
table(data_coil$V341)
```

```
no-coil    coil
   4443    5557
```

Creating training and test datasets

We are going to divide our data into two portions: a training dataset that will be used to build the KNN model and a test dataset that will be used to estimate the predictive accuracy of the model.

```
train_indx_coil <- sample(x= 1:nrow(data_coil),
                      size= 0.67*nrow(data_coil),
                      replace=FALSE)

train_data_coil <- data_coil[train_indx_coil,-ncol(data_coil)]
test_data_coil <- data_coil[-train_indx_coil,-ncol(data_coil)]


train_label_coil <- data_coil[train_indx_coil, ncol(data_coil)]
test_label_coil <- data_coil[-train_indx_coil, ncol(data_coil)]
```

The Receiver Operating Characteristic curve is used to examine the trade-off between the detection of true positives, whiile avoiding the false positives. we are goint to calculate de AUC from all the k-s

```
# ROC and AUC for k=1
aminocoil_test_pred1 <- knn(train = train_data_coil, test = test_data_coil, cl = train_label_coil, k=1,p
prob1 <- attr(aminocoil_test_pred1, "prob")
prob1 <- ifelse(aminocoil_test_pred1 == "coil", 1-prob1, prob1) - 1
labels1 <- factor(test_label_coil, labels = c(0,1))

roc1 <- pROC::roc(labels1, prob1, auc= TRUE, ci=TRUE)
```

```
Setting levels: control = 0, case = 1
```

```
Setting direction: controls > cases
```

```
# ROC and AUC for k=3
aminocoil_test_pred3 <- knn(train = train_data_coil, test = test_data_coil, cl = train_label_coil, k=3,p
prob3 <- attr(aminocoil_test_pred3, "prob")
prob3 <- ifelse(aminocoil_test_pred3 == "coil", 1-prob3, prob3) - 1
labels3 <- factor(test_label_coil, labels = c(0,1))

roc3 <- pROC::roc(labels3, prob3, auc= TRUE, ci=TRUE)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls > cases
```

```
# ROC and AUC for k=5
aminocoil_test_pred5 <- knn(train = train_data_coil, test = test_data_coil, cl = train_label_coil, k=5,p
prob5 <- attr(aminocoil_test_pred5, "prob")
prob5 <- ifelse(aminocoil_test_pred5 == "coil", 1-prob5, prob5) - 1
labels5 <- factor(test_label_coil, labels = c(0,1))

roc5 <- pROC::roc(labels5, prob5, auc= TRUE, ci=TRUE)
```

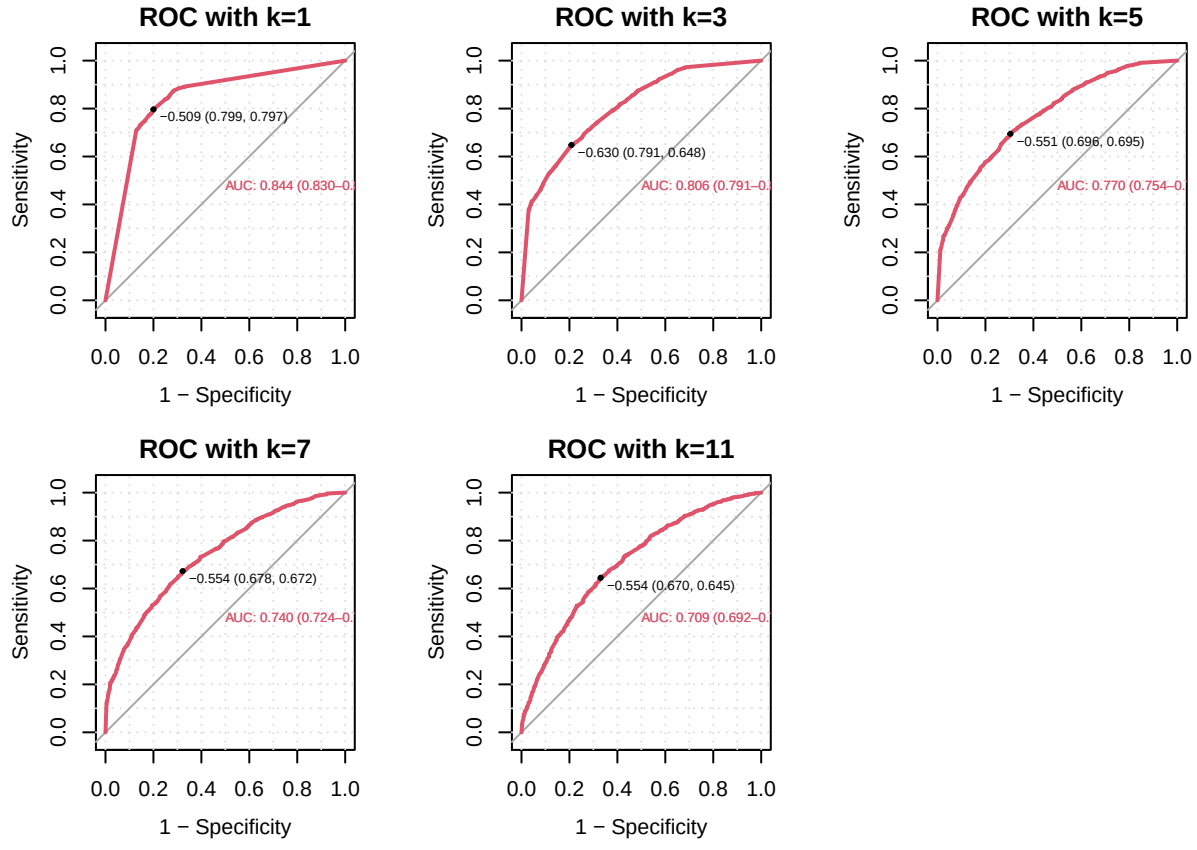
```
Setting levels: control = 0, case = 1
Setting direction: controls > cases
```

```r
# ROC and AUC for k=7
aminocoil_test_pred7 <- knn(train = train_data_coil, test = test_data_coil, cl = train_label_coil, k=7,p
prob7 <- attr(aminocoil_test_pred7, "prob")
prob7 <- ifelse(aminocoil_test_pred7 == "coil", 1-prob7, prob7) - 1
labels7 <- factor(test_label_coil, labels = c(0,1))

roc7 <- pROC::roc(labels7, prob7, auc= TRUE, ci=TRUE)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls > cases
```

```r
# ROC and AUC for k=11
aminocoil_test_pred11 <- knn(train = train_data_coil, test = test_data_coil, cl = train_label_coil, k=11
prob11 <- attr(aminocoil_test_pred11, "prob")
prob11 <- ifelse(aminocoil_test_pred11 == "coil", 1-prob11, prob11) - 1
labels11 <- factor(test_label_coil, labels = c(0,1))

roc11 <- pROC::roc(labels11, prob11, auc= TRUE, ci=TRUE)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls > cases
```



The perfect classifier has a curve that passes through the point at a 100 percent true positive rate and 0 percent false positive rate. The closer the curve is to the perfect classifier, the better it is at identifying

positive values. This measured using a statistic known as the *area under the ROC curve* (abbreviated AUC). The best AUC value it was 1NN classifier, next 3NN and the others are similar between them.

k value	# false negatives	# false positives	% classified Incorrectly
1	338	333	20.33333
3	433	513	28.66667
5	459	559	30.84848
7	456	624	32.72727
11	447	689	34.42424