0.1 K-nearest Neigbours

0.1.1 Description

1. Given training set of size M, N feature values, and 1 binary outcome (0 or 1), we have a table of feature values x_{ij} and a vector of outcome y_i for $1 \leq i \leq M,\ 1 \leq j \leq N$

2. Given any test point x^*, with N feature values x_j^*, for $1 \leq j \leq N$, calculate euclidean distance of x^* to each training point x_i, i.e. for $1 \leq i \leq M$, $\text{distance}_i = \sqrt{\sum\limits_{j=1}^{N}(x_{ij} - x_j^*)^2}$

3. Given chosen value k, the k-nearest neighbours/training points x_i to x^*, denoted $N_k(x^*)$, is the set of the first k x_i in the sequence of x_i sorted by increasing distance_i

4. $\hat{Y}(x^*) = \frac{1}{k} \sum\limits_{x_i \in N_k(x^*)} y_i$

5. The predicted outcome for $x^* = y^* = \begin{cases} 1 & \hat{Y} > \sigma \\ 0 & \hat{Y} < \sigma \end{cases}$ where σ is the threshold. $\sigma = 0.5$ in the majority rule.

0.1.2 Choice of σ

1. As σ increases,
   (a) TP, TPR, FP, and FPR decreases or stays the same
   (b) TN, TNR, FN, and FNR increases or stays the same

2. As σ decreases,
   (a) TP, TPR, FP, and FPR increases or stays the same
   (b) TN, TNR, FN, and FNR decreases or stays the same

3. See Diagnostics of Classifiers ROC Curve for more info

0.1.3 Choice of k

1. when k increases, the variance decreases, but bias increases

2. when k decreases, the variance increases, but bias decreases

3. See Diagnostics of Classifiers Bias-Variance Tradeoff for more info

0.1.4 Prediction Surface

1. Boundaries can be curvy
2. Can be not axis-aligned

0.1.5 Standardising Variables

1. Any variable with a larger scale than others will have a larger effect on the Euclidean distance
2. To prevent this problem, we can standardise our data so that all our variables will have mean of zero and standard deviation of one with the scale function in R

```r
data <- cbind(
    1:5,
    seq(100, 500, 100),
    sample(0:1, 5, replace = TRUE)
)
data

##      [,1] [,2] [,3]
## [1,]    1  100    1
## [2,]    2  200    0
## [3,]    3  300    1
## [4,]    4  400    0
## [5,]    5  500    1

data[, 1:2] = scale(data[, 1:2])
data

##            [,1]       [,2] [,3]
## [1,] -1.2649111 -1.2649111    1
## [2,] -0.6324555 -0.6324555    0
## [3,]  0.0000000  0.0000000    1
## [4,]  0.6324555  0.6324555    0
## [5,]  1.2649111  1.2649111    1
```

0.1.6 R Implementation

```r
library(class)
data <- matrix(
    c(1, 1, 0,
      1, 2, 0,
      2, 1, 0,
      2, 2, 0,
      2, 3, 0,
      8, 8, 1,
      9, 8, 1,
      8, 7, 1,
      9, 9, 1,
      9, 7, 1), nrow = 10, byrow = TRUE
)
train <- sample(1:10, 5, replace = FALSE)
train.x <- data[train, 1:2]
train.y <- data[train, 3]
cbind(train.x, train.y)

##          train.y
## [1,] 2 3       0
## [2,] 8 8       1
## [3,] 1 1       0
## [4,] 1 2       0
## [5,] 9 8       1

test.x <- data[-train, 1:2]
test.y <- data[-train, 3]
cbind(test.x, test.y)

##          test.y
## [1,] 2 1      0
## [2,] 2 2      0
## [3,] 8 7      1
## [4,] 9 9      1
## [5,] 9 7      1

knn.pred <- knn(train.x, test.x, train.y, k=3)
confusion.matrix <- table(test.y, knn.pred)
confusion.matrix
```

```
##         knn.pred
## test.y 0 1
##        0 2 0
##        1 0 3
```

0.1.7 Calculation Intensive Exam Questions & Solutions

Euclidean Distances, $\hat{Y}$, and Prediction for 1 Test Data Point
Adapted from Midterm Q17-18. Suppose we have a training set of 5 data points with binary value outcome = c(1,1,0,1,0), x_1 = c(1,2,1,3,3), and x_2 = c(3,2,1,3,1). Using the 3-nearest neighbors classifier and the majority, what is the **fitted outcome value** $\hat{Y}$ and the **predicted outcome value** of $(x_1^*, x_2^*) = (2,4)$?
Solution

1. Copy paste the following code:

```
distance <- function(m, t) {
  # returns numbered table of euclidean distance of t to each
  # training point in m
  # each column in m is feature variable except last column is
  # outcome y
  if (length(t) != ncol(m)-1) {
    print("test data does not match number of feature variables")
    return
  }
  cd <- function(p1, p2) {
    return (sqrt(sum((p1-p2)^2)))
  }

  table <- data.frame(id = 1:nrow(m))
  colnames(m) <- c(paste("x_", 1:(ncol(m)-1), sep = ""), "y")
  table <- cbind(table, m)
  dist <- rep(1, times<-nrow(m))
  for (r in 1:nrow(m)) {
    dist[r] <- cd(m[r, 1:(ncol(m)-1)], t)
  }
  table <- cbind(table, dist)
  return (table)
}
```

```
sorts <- function(d) {
  # sorts the table output of distance function in increasing
  # euclidean distance
  return (d[order(d[, ncol(d)]), ])
}

y_hat <- function(s, k) {
 # calculate y-hat from table output of sorts function given value of k
 return (sum(s[1:k, ncol(s)-1])/k)
}

predict <- function(y, s) {
 # return predicted class given y-hat y and threshold value s (sigma)
 # assumes y !<- s, i.e. no tie
 return (if (y > s) 1 else 0)
}
```

2. Calculate Euclidean Distances to (2, 4)

```
dist_matrix <- distance(matrix(
  c(1,2,1,3,3,
    3,2,1,3,1,
    1,1,0,1,0), nrow = 5, byrow = FALSE
), c(2, 4))
dist_matrix

##   id x_1 x_2 y     dist
## 1  1   1   3 1 1.414214
## 2  2   2   2 1 2.000000
## 3  3   1   1 0 3.162278
## 4  4   3   3 1 1.414214
## 5  5   3   1 0 3.162278
```

3. Sort By Increasing Euclidean Distances

```
sorted_dist_matrix <- sorts(dist_matrix)
sorted_dist_matrix
```

```
##   id x_1 x_2 y     dist
## 1  1   1   3 1 1.414214
## 4  4   3   3 1 1.414214
## 2  2   2   2 1 2.000000
## 3  3   1   1 0 3.162278
## 5  5   3   1 0 3.162278
```

4. Calculate **fitted outcome value** $\hat{Y}$ based on value of k=3

```
y_hat_value <- y_hat(sorted_dist_matrix, k=3)
y_hat_value

## [1] 1
```

5. Calculate **predicted outcome value** based on majority rule $\sigma = 0.5$

```
predicted_value <- predict(y_hat_value, s=0.5)
predicted_value

## [1] 1
```

$\hat{Y}$ to Confusion Matrix for n Test Data Points

Adapted from Midterm Q14. Suppose we have k-nearest neighbours classifier for binary outcome Y. The table below shows the actual and fitted outcome for $n = 10$ test data points.

Actual Y	1	1	0	1	1	0	0	0	1	0
$\hat{Y}$	0.9	0.8	0.8	0.6	0.5	0.5	0.5	0.3	0.2	0.1

What is the True Positive Rate (TPR) when we predict $Y = 1$ if $\sigma > 0.7$

Solution

1. Copy paste the following code

```
predict <- function(y, s) {
  # return predicted class given y-hat y and threshold value s (sigma)
  # assumes y !<- s, i.e. no tie
  return (ifelse(y > s, 1, 0))
}
```

2. Obtain predictions for $\sigma = 0.7$

```r
predictions <- predict(
  c(0.9, 0.8, 0.8, 0.6, 0.5, 0.5, 0.5, 0.3, 0.2, 0.1), s=0.7
)
predictions

##  [1] 1 1 1 0 0 0 0 0 0 0
```

3. Generate Confusion Matrix

```r
table(c(1, 1, 0, 1, 1, 0, 0, 0, 1, 0), predictions)

##    predictions
##      0 1
##    0 4 1
##    1 3 2
```

4. Refer to Section 3.11 for metric calculations