

# Machine Learning, Spring 2021

## Homework 6

Daniil Bulat, Jonas Josef Huwyler, Haochen Li, Giovanni Magagnin

4/29/2021

### Exercise 1

Consider the WDBC data with the two features `perimeter_mean` and `concaveP_mean`. Load the *library(class)* and use the *knn()*-command for the K-nearest neighbor algorithm.

```
## Warning: package 'class' was built under R version 4.0.5
```

### Question 1

Split the data into training and testing sets in the ratio 70%-30%.

```
# Shuffle and Split Data (70-30)
set.seed(123)
Data = Data[sample(nrow(Data)),]

# Create data frame for data to be used
df = data.frame(id = Data$id,
                diagnosis = Data$diagnosis,
                perimeter_mean = Data$perimeter_mean,
                concaveP_mean = Data$concaveP_mean)
ntrain = floor(nrow(df)*0.7)
df.Train = df[1:ntrain,]
df.Test = df[(ntrain+1):nrow(Data),]
```

### Question 2

Run the K-NN algorithm for  $K = 1, 10, 25$  with *knn(..., prob=TRUE)*. What type of predictions do you get, what do they mean?

```
xvars = c("perimeter_mean", "concaveP_mean")
yvars = "diagnosis"

knn_T_1 = knn(train = df.Train[xvars],
              test = df.Test[xvars],
              cl = as.matrix(df.Train[yvars]),
              k = 1,
              prob = TRUE )

knn_T_10 = knn(train = df.Train[xvars],
               test = df.Test[xvars],
               cl = as.matrix(df.Train[yvars]),
               k = 10,
```

```

        prob = TRUE )

knn_T_25 = knn(train = df.Train[xvars],
               test = df.Test[xvars],
               cl = as.matrix(df.Train[yvars]),
               k = 25,
               prob = TRUE )

print(knn_T_1)

##      [1] B B B B M B M B M M M B B B M M B B M B B B M B B B M M B B B
##     [38] M B B B M M B M B B M B B B B B B B B B M M B B M M B B B B M M M
##     [75] B B B B M M M B B B B M B B B B M M B M M B B B M B B M B M B B
##    [112] B B B B B M M M M M B B B M B B M M M M M B M M B B B M M B B M B B M
##    [149] B M B M B B B B B M M M M B B B M M B B M B M B M
## attr(,"prob")
##      [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##     [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##     [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##    [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##    [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: B M

print(class(knn_T_1))

```

```
## [1] "factor"
```

We have a training data set on which we train the KNN algorithm. This means that the algorithm learns the location of every training data point and its type. Then the algorithm takes the location of every single test data points, looks at the type of its  $k$  (i.e. 1, 10 or 25) nearest neighbors and predicts the class of the test data point according to a majority vote.

The probability which is in this case displayed for the KNN algorithm shows us the percentage (or how close) the majority vote was. If our predicted class was M and the probability 0.75, this means that 75% of the  $k$  neighbors of that point “voted” for the M-class.

### Question 3

Run the K-NN algorithm for  $K = 1, 10, 25$  with `knn(..., prob=FALSE)` and compute the (test) error rate.

```

knn_F_1 = knn(train = df.Train[xvars],
               test = df.Test[xvars],
               cl = as.matrix(df.Train[yvars]),
               k = 1,
               prob = FALSE )

knn_F_10 = knn(train = df.Train[xvars],
                test = df.Test[xvars],
                cl = as.matrix(df.Train[yvars]),
                k = 10,
                prob = FALSE )

knn_F_25 = knn(train = df.Train[xvars],
                test = df.Test[xvars],
                cl = as.matrix(df.Train[yvars]),

```

```

        k = 25,
        prob = FALSE )

err_F_1 = round(sum(knn_F_1 != as.matrix(df.Test[yvars]))/length(knn_F_1)* 100,2)
err_F_10 = round(sum(knn_F_10 != as.matrix(df.Test[yvars]))/length(knn_F_10)* 100,2)
err_F_25 = round(sum(knn_F_25 != as.matrix(df.Test[yvars]))/length(knn_F_25)* 100,2)

K = c(1,10,25)
errors = c(err_F_1,err_F_10,err_F_25)

for (i in 1:3){
  cat("The error rate of K-NN algorithm when K = ", K[i], "is", errors[i], "%\n")
}

## The error rate of K-NN algorithm when K = 1 is 12.87 %
## The error rate of K-NN algorithm when K = 10 is 12.28 %
## The error rate of K-NN algorithm when K = 25 is 12.87 %

```

## Question 4

Run logistic regression on the same data, and predict “malign” when the probability is at least 0.5 and predict “benign” otherwise. Compute the (test) error rate.

```

glm.fit = glm(diagnosis~perimeter_mean + concaveP_mean,data = df.Train,family = binomial)
summary(glm.fit)

##
## Call:
## glm(formula = diagnosis ~ perimeter_mean + concaveP_mean, family = binomial,
##      data = df.Train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.45821  -0.26302  -0.10627   0.06101   2.59877
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -13.28007    1.77748  -7.471 7.94e-14 ***
## perimeter_mean   0.09716    0.01927   5.042 4.62e-07 ***
## concaveP_mean   77.06571   11.42843   6.743 1.55e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 515.00  on 397  degrees of freedom
## Residual deviance: 149.32  on 395  degrees of freedom
## AIC: 155.32
##
## Number of Fisher Scoring iterations: 7

yhat = predict(glm.fit, type = "response",newdata = df.Test)
yhat = ifelse(yhat<0.5,"B","M")

err_logit = round(sum(yhat != as.matrix(df.Test[yvars]))/length(yhat) * 100,2)

```

```
cat("The error rate of logistic algorithm is", err_logit, "%\n")
```

```
## The error rate of logistic algorithm is 12.28 %
```

## Question 5

Compare the four results (three times K-NN and logistic regression).

```
errors = data.frame(KNN1 = err_F_1,
                    KNN10 = err_F_10,
                    KNN25 = err_F_25,
                    Logistic = err_logit,
                    row.names = "Error Rate")
print(errors)
```

```
##           KNN1 KNN10 KNN25 Logistic
## Error Rate 12.87 12.28 12.87    12.28
```

First, let's take a look at the three times K-NN algorithms. When  $K = 1$  and  $K = 25$ , the error rates are equal to 12.87. When  $K = 10$ , the error rate is smaller than the former two models, which suggests that the error rates of KNN regressions perform a U-shape, and the algorithm with  $K = 10$  performs better than the other two.

Second, the logistic regression's error rate is identical to the KNN algorithm with  $K = 10$ . As we know, if we have a linear relationship, the linear model would perform better than the non-parametric model. However, in this case, we are not sure whether the true model is linear or not. The condition that the two models share same error rate could also suggest that they define similar separating hyperplanes and have same predictions of classification.

## Exercise 2

Consider an observation  $x = (1, 1)$  and the following 7 nearest neighbors:

$$\mathcal{N}_x = \{(0, 2), (0, -1), (3, -1), (2, 3), (-3, 1), (0, 5), (5, 4)\}$$

with the corresponding class labels (in the same order)

$$\{1, 1, 2, 3, 3, 2, 2\}$$

First compute the distances from the neighbors to  $x$ . Then compute the probabilities for  $x$  to be in class 1,2,3 (respectively) with weighted 7-NN for the following weights:

## Question 1

$$w = \left(\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}\right)$$

**This corresponds to normal 7-NN.** First, let's calculate the euclidean distances (with the same order) of  $\mathcal{N}_x$  and  $x$ ,

$$d = \{\sqrt{2}, \sqrt{10}, 2\sqrt{2}, \sqrt{5}, 4, \sqrt{17}, 5\}$$

Then the probabilities for  $x$  to be in class 1,2,3 will be

$$P(y = 1|x) = \frac{2}{7}$$

$$P(y = 2|x) = \frac{3}{7}$$

$$P(y = 3|x) = \frac{2}{7}$$

## Question 2

$$w = (\frac{9}{34}, \frac{9}{34}, \frac{6}{34}, \frac{4}{34}, \frac{3}{34}, \frac{2}{34}, \frac{1}{34})$$

**This is a weighted 7-NN. (The weighted vector follows the order of distance from nearest to farthest)** First, let's calculate the euclidean distances (with the same order) of  $\mathcal{N}_x$  and  $x$ ,

$$d = \{\sqrt{2}, \sqrt{10}, 2\sqrt{2}, \sqrt{5}, 4, \sqrt{17}, 5\}$$

Then the probabilities for  $x$  to be in class 1,2,3 will be

$$P(y = 1|x) = \frac{9+4}{34} = \frac{13}{34}$$

$$P(y = 2|x) = \frac{6+2+1}{34} = \frac{9}{34}$$

$$P(y = 3|x) = \frac{9+3}{34} = \frac{12}{34} = \frac{6}{17}$$

## Exercise 3

### Question 1

Write your own 2-NN regression algorithm for the Olympics data.

### Question 2

Use the data for the years 1906,1932,1968,2000 as testing data (the rest are training data) and compute the test-RSS for your 2-NN regression algorithm.