k-Nearest Neighbors

Exercise 01

Research some real-world applications of Machine Learning and describe applications for each of the problems below. For each of these problems, include a description of feature vectors, labels, or responses when appropriate:

a) Classification problem.

b) Regression problem.

c) Clustering problem.

a) Classification Problem

One of the applications for the classification problem is email spam filters. Various methods can be employed for this type of problem, such as NLP-based classification, Bayesian classification, and complex neural networks. These methods rely on a large training dataset, which is provided whenever a user manually classifies an email as spam or non-spam, and extract features to determine whether the email is spam or not (0-1 classification).

In this case, feature vectors include all parameters that might influence the email's classification as spam, such as content, sender, keywords in the email body, attachments, structure, among others. The label would be a binary or Boolean value indicating whether the email is spam or not, such as "spam" or "non-spam," true for spam and false for non-spam.

b) Regression Problem

An example of a regression problem is temperature prediction. Temperature is considered a continuous variable, meaning it can take any value within a given range. This distinguishes regression from classification, where the possible outcomes are predefined values. For example, attempting to associate "hot" or "cold" with a given temperature would be better suited for a classification approach.

Thus, the feature vector would have entries like precipitation, current temperature, historical series, atmospheric pressure, wind intensity and direction. The response (equivalent to the label in classification) would be the predicted temperature. This way, it is possible to train a model to predict the temperature in the upcoming hours using regression.

c) Clustering Problem

The clustering problem involves forming groups (clusters) based on similarities or common characteristics among data points. Using these characteristics, the distance or similarity between data points is calculated. Data points that are closer in proximity are grouped together. Unlike classification and regression problems, data points do not have pre-defined labels or functional values.

An example is a recommendation algorithm within a music platform. Based on the user's listening history, distance (or similarity) measures can be used to identify other songs with similar characteristics to the user's preferences. Recommendations for other titles of potential interest can be provided based on this grouping.

In this case, the input includes the songs the user has listened to, which will be associated with a cluster of songs with similar characteristics such as genre, decade, rhythm, and artist. It is worth noting that the clustering method is commonly used in suggestions, but refining is necessary when extrapolating behaviors of a group, as differences can exist between data points associated with the same cluster.

Exercise 02

In your own words, explain the ”curse of dimensionality”.

As we have seen, each Machine Learning problem requires a feature vector, and the dimension of a problem is defined by the length of this feature vector. The curse of dimensionality arises from the fact that as the dimension of a problem increases, the distance between neighboring points also increases. Consequently, it becomes necessary to traverse increasingly longer paths to find the nearest neighbors, which may result in the neighborhood no longer accurately representing the original point's characteristics.

While increasing the number of data points can help alleviate this issue, it comes at a performance cost. Even then, for high dimensions, a substantial number of data points is required to minimize distances.

The kNN method performs well when dealing with problems that have few dimensions and a large number of data points, as the distance between a point and its neighbors remains small, ensuring the neighborhood remains representative of the point's characteristics.

Exercise 03

Implement the k-nearest neighbors (kNN) method for a classification problem. Your function should accept the following arguments:

1. An integer ’k’ representing the number of neighbors to use in kNN.
2. A vector $x = (x1, x2)$ with two components.
3. A DataFrame ’D’ with columns $x1, x2$, and y, where $x1$ and $x2$ are numerical values representing two components of the feature vector x (i.e., $x = (x1, x2)$), and y is a categorical variable representing the labels of the points.

Here is an example of such a DataFrame with two classes:

```r
library(tidyverse)
D <- tibble( x_1 = rnorm(100,1,1),
          x_2 = rnorm(100,-1,2),
          y = factor(sample(c("one","two","three"),100,replace = T)))
head(D)
```

```
## # A tibble: 6 x 3
##     x_1    x_2 y
##   <dbl>  <dbl> <fct>
## 1 2.82  -2.21  three
## 2 1.65   1.52  two
## 3 1.88   0.268 one
## 4 0.405 -3.83  three
## 5 2.04  -4.91  one
## 6 1.42  -3.15  two
```

The function should have the signature function(k, x, D) and should return the most probable class associated with point x.

Tip: You can perform kNN using a sequence of commands chained by the pipe operator %>%. For example, test the following sequence of commands with the previous DataFrame ‘D’:

```
x = c(1,2)
k = 10
D2 <- D %>%
  mutate( dist = (x[1] - x_1)**2 + (x[2] - x_2)**2 ) %>%
  arrange( dist ) %>% head(k) %>% count(y)
```

Solution:

```
library(tidyverse)
dataframe <- tibble(x_1 = rnorm(100, 1, 1),
x_2 = rnorm(100, -1, 2),
y = factor(sample(c("one", "two", "three"), 100, replace = TRUE)))

knn <- function(k, x, df) {
   df$distance <- sqrt((df$x_1 - x[1])^2 + (df$x_2 - x[2])^2)
   df <- df %>% arrange(distance)
   selected_values <- df %>%
   slice(1:k) %>% pull(y)
   most_common <- names(table(selected_values))[which.max(table(selected_values))]
   return(most_common)
}

t <- c(3.30, -4.37)
print(knn(10, t, dataframe))
```

```
## [1] "two"
```

```
print(knn(1, t, dataframe))
```

```
## [1] "two"
```

Exercise 04

Using the iris database and your previous kNN implementation, calculate the number of correctly classified points according to the Species label using the Petal.length and Sepal.length columns. Compare the results from k = 10 and k = 1.

```
library(tidyverse)
data("iris") # Load the dataset
iris <- as_tibble(iris) %>% # Convert to a tibble
  select(Petal.Length, Sepal.Length, Species) %>% # Select the columns
  rename(x_1 = Petal.Length, x_2 = Sepal.Length, y = Species)

knn <- function(k, x, df) {
  df$distance <- sqrt((df$x_1 - x[1]) ^ 2 + (df$x_2 - x[2]) ^ 2)
  df <- df %>% arrange(distance)
  selected_values <- df %>%
  slice(1:k) %>% pull(y)
  most_common <- names(table(selected_values))[which.max(table(selected_values))]
  return(most_common)
}
```

```r
predicted_values <- function(k) {
  prediction <- map_lgl(1:nrow(iris), function(i) {
  row_parameters <- iris[i, c("x_1", "x_2")]
  x_1 <- as.double(row_parameters$x_1)
  x_2 <- as.double(row_parameters$x_2)
  x <- c(x_1, x_2)

  row_response <- iris[i, c("y")]
  y <- row_response$y

  response <- knn(k, x, iris)

  as.character(response) == y
})

  return(prediction)
}

correct_1 <- sum(predicted_values(1))
correct_10 <- sum(predicted_values(10))

print(paste("Number of correctly classified observations with k = 1:", correct_1))
```

```
## [1] "Number of correctly classified observations with k = 1: 149"
```

```r
print(paste("Number of correctly classified observations with k = 10:", correct_10))
```

```
## [1] "Number of correctly classified observations with k = 10: 143"
```