Problem Solutions

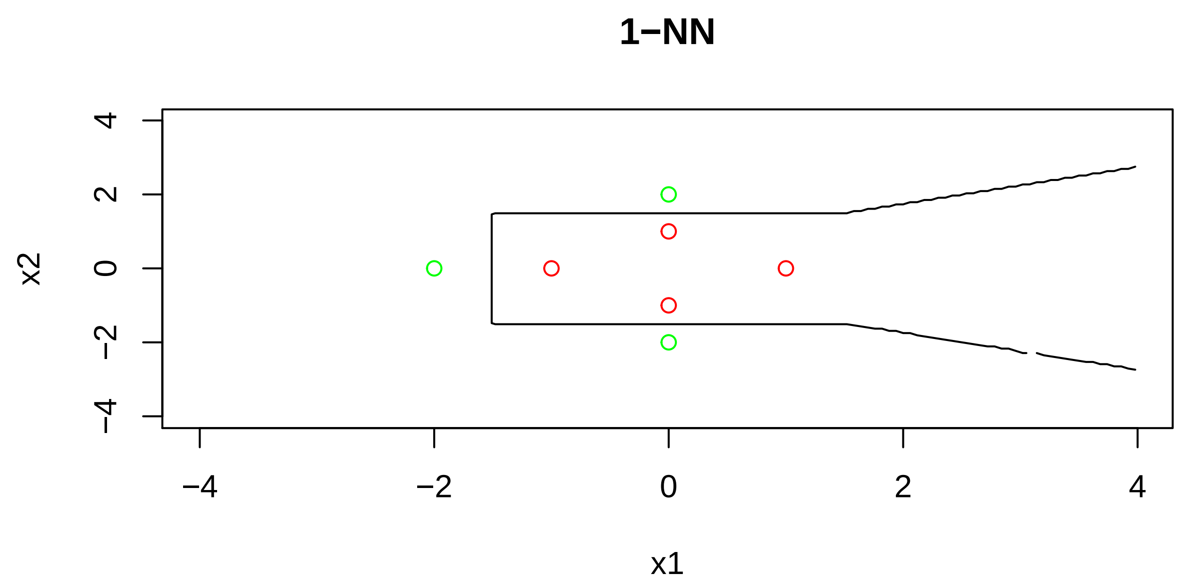
e-Chapter 6

Pierre Paquay

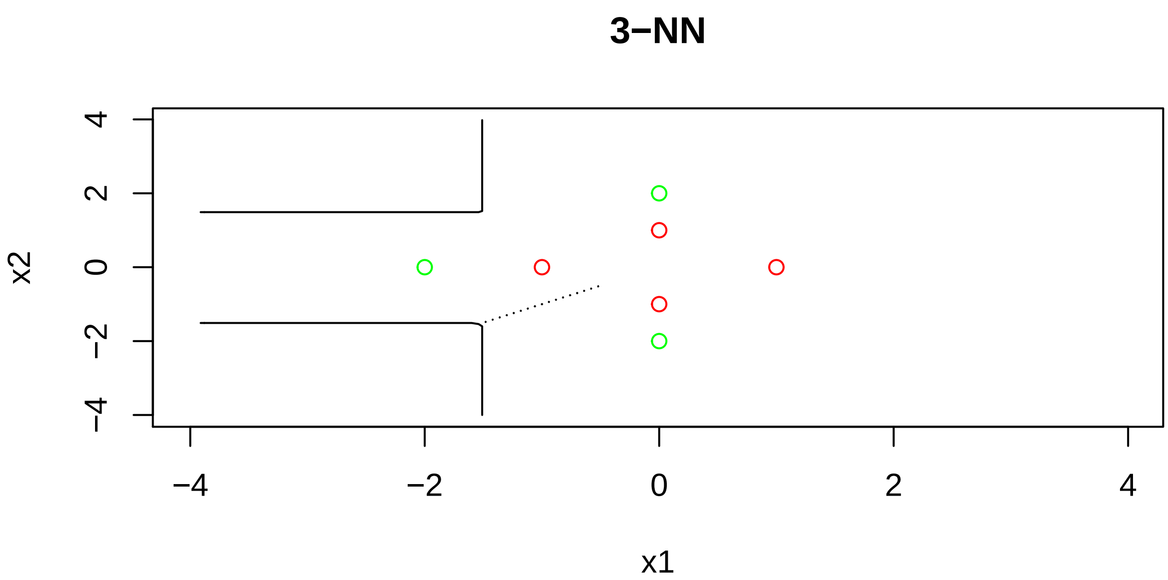
Problem 6.1

(a) Below we plot the decision regions for the 1-NN and 3-NN rules.

```
set.seed(10)
X <- matrix(c(1, 0, 0, -1, 0, 0, -2, 0, 1, -1, 0, 2, -2, 0), nrow = 7)
seq <- seq(-4, 4, by = 0.06)
Xnew <- expand.grid(seq, seq)
labels <- c(-1, -1, -1, -1, 1, 1, 1)

NN_1 <- knn(X, Xnew, labels, k = 1, prob = TRUE)
prob <- attr(NN_1, "prob")
prob <- ifelse(NN_1 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "1-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```

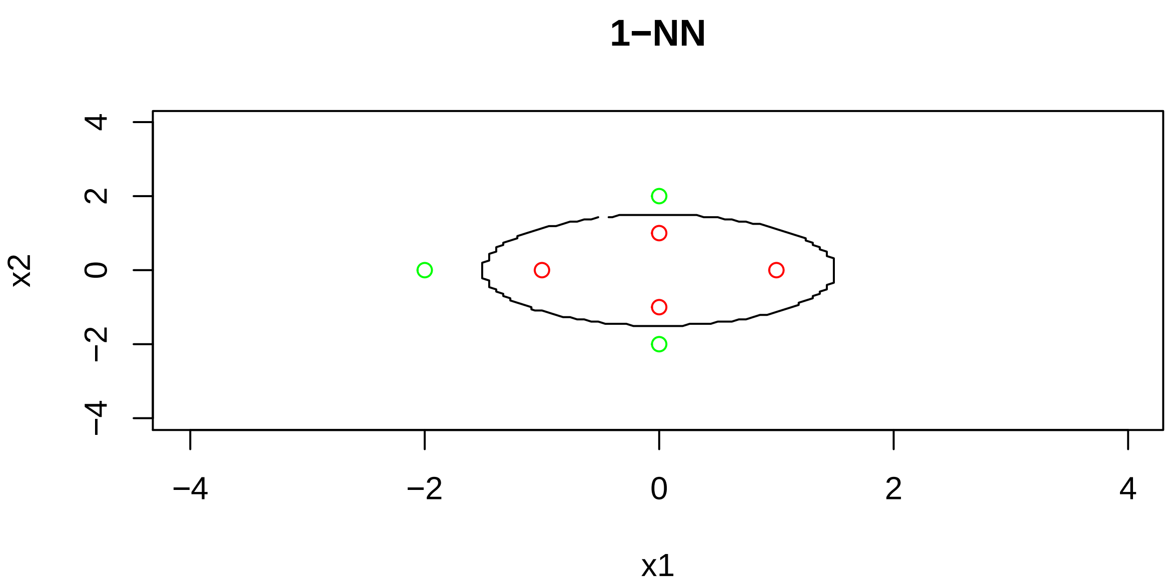



```
NN_3 <- knn(X, Xnew, labels, k = 3, prob = TRUE)
prob <- attr(NN_3, "prob")
prob <- ifelse(NN_3 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "3-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```

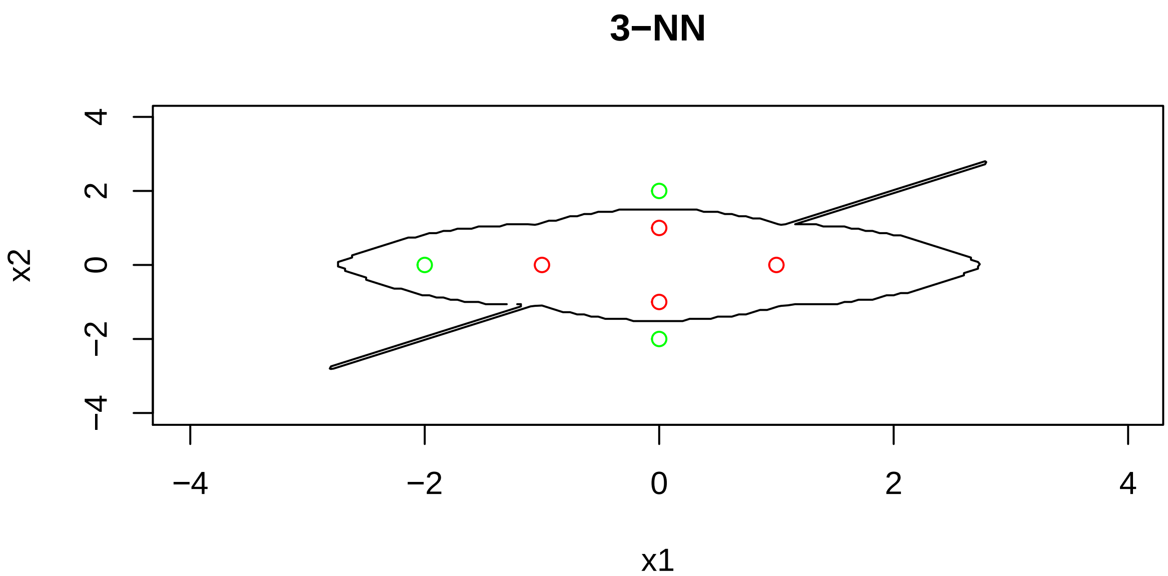


(*b*) Below we plot the classification regions in the x-space for the 1-NN and 3-NN rules implemented on the data in the z-space.

```
Z1 <- apply(X, 1, function(x) sqrt(x[1]^2 + x[2]^2))
Z2 <- apply(X, 1, function(x) atan(x[2]/ x[1]))
Z <- matrix(c(Z1, Z2), byrow = FALSE, ncol = 2)
Znew1 <- apply(Xnew, 1, function(x) sqrt(x[1]^2 + x[2]^2))
Znew2 <- apply(Xnew, 1, function(x) atan(x[2]/ x[1]))
Znew <- matrix(c(Znew1, Znew2), byrow = FALSE, ncol = 2)

NN_1 <- knn(Z, Znew, labels, k = 1, prob = TRUE)
prob <- attr(NN_1, "prob")
prob <- ifelse(NN_1 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "1-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```



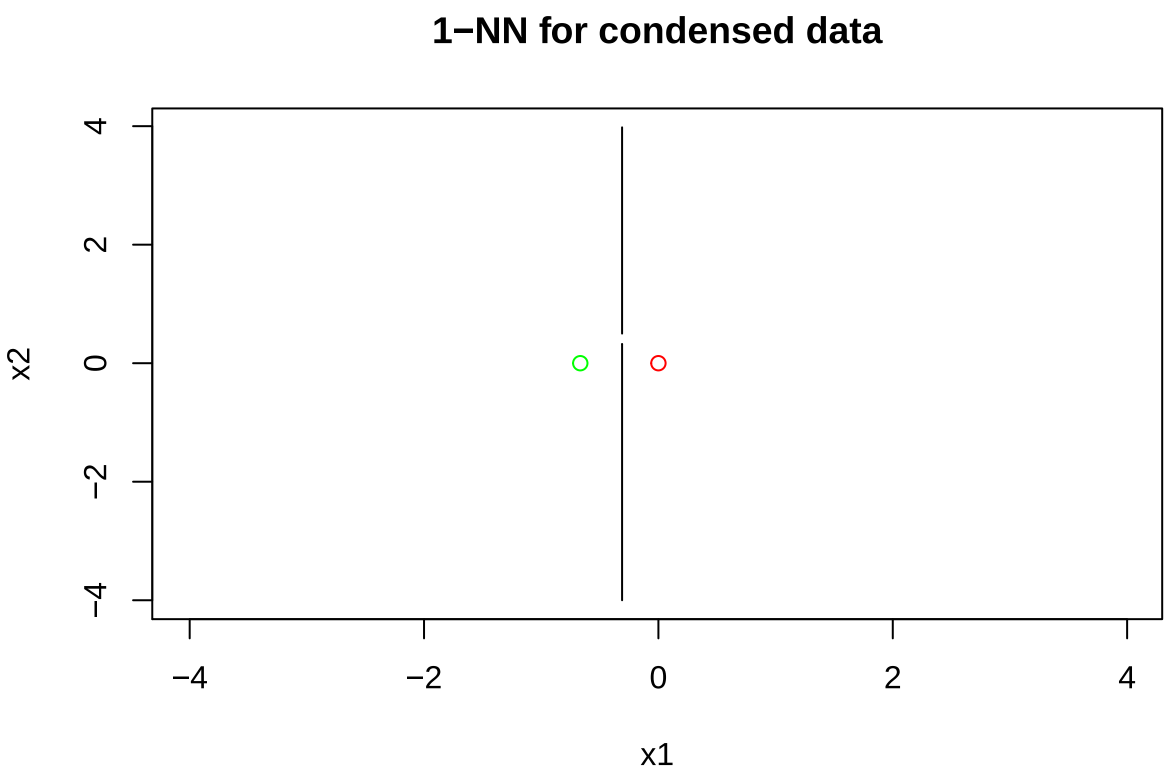
```
NN_3 <- knn(Z, Znew, labels, k = 3, prob = TRUE)
prob <- attr(NN_3, "prob")
prob <- ifelse(NN_3 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "3-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```



Problem 6.2

(a) Below we plot the classification regions for the 1-NN rule using the condensed data.

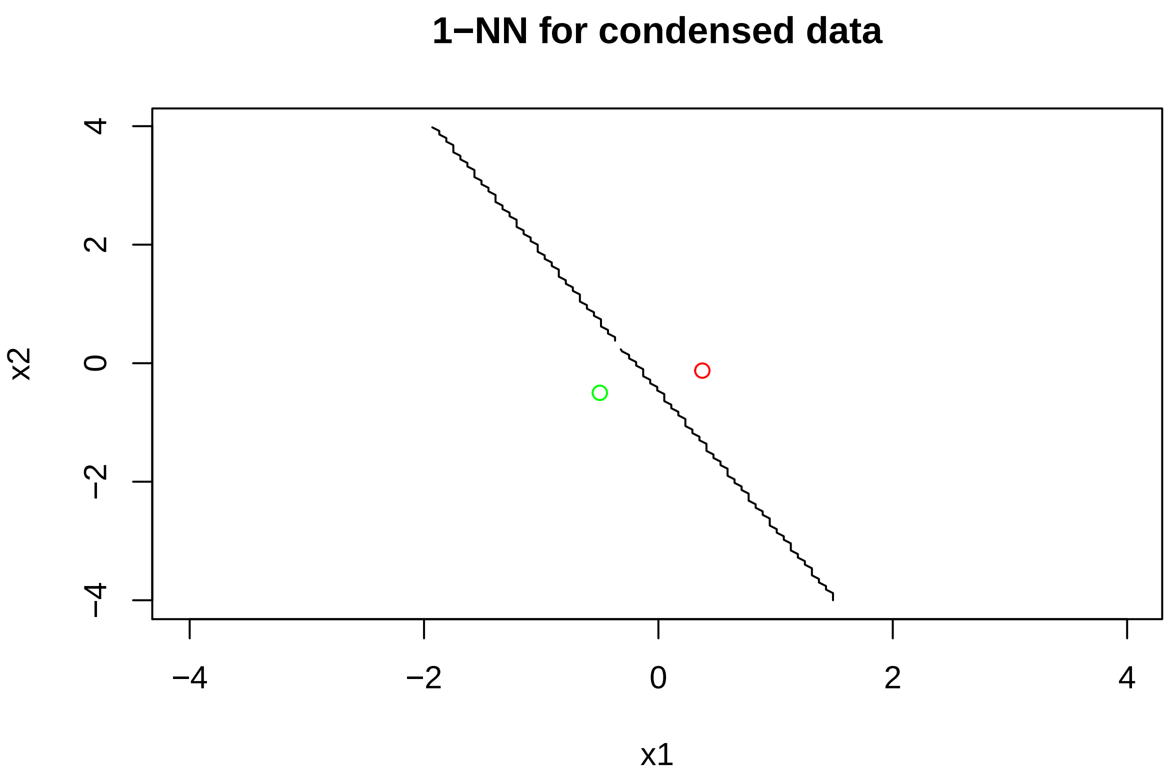
```
mu_1 <- apply(X[labels == 1, ], 2, mean)
mu_min1 <- apply(X[labels == -1, ], 2, mean)
Xcond <- rbind(mu_1, mu_min1)
labels2 <- c(1, -1)
knn_pred <- knn(Xcond, X, labels2, k = 1, prob = TRUE)

NN_1 <- knn(Xcond, Xnew, labels2, k = 1, prob = TRUE)
prob <- attr(NN_1, "prob")
prob <- ifelse(NN_1 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "1-NN for condensed data")
points(Xcond, col = ifelse(labels2 == -1, "red", "green"))
```



Here, we have an in-sample error equal to 0.4285714.

(b) It is easy to see that the method of condensing gives us two points of coordinates $(-1/2, -1/2)$ and $(3/8, -1/8)$ with labels of $+1$ and -1 respectively.

```
cond_1 <- c(-1/2, -1/2)
cond_min1 <- c(3/8, -1/8)
Xcond <- rbind(cond_1, cond_min1)
labels2 <- c(1, -1)
knn_pred <- knn(Xcond, X, labels2, k = 1, prob = TRUE)

NN_1 <- knn(Xcond, Xnew, labels2, k = 1, prob = TRUE)
prob <- attr(NN_1, "prob")
prob <- ifelse(NN_1 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(seq), length(seq))
contour(seq, seq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "1-NN for condensed data")
points(Xcond, col = ifelse(labels2 == -1, "red", "green"))
```



In this case, we have an in-sample error equal to 0.4285714 which is exactly equal to the in-sample error of point (a).

Problem 6.3

First, we begin by using a Cholesky factorization on the symmetric positive semi-definite matrix Q, we get $Q = T^T T$ where T is an upper triangular matrix with $r = \text{rank}(Q)$ positive diagonal elements and $n - r$ null rows. Our chosen transformation is then $\Phi(x) = Tx$ since in this case we have

$$\begin{aligned} d(x, x') &= (x - x')^T Q (x - x') \\ &= (x - x')^T T^T T (x - x') \\ &= (Tx - Tx')^T (Tx - Tx') \\ &= (\Phi(x) - \Phi(x'))^T (\Phi(x) - \Phi(x')) \end{aligned}$$

which is actually the euclidean distance in the z-space. In this context, our z-space is $\{Tx : x \in \mathbb{R}^d\} = \text{span}(T)$ the column space of T, and it is well-known that

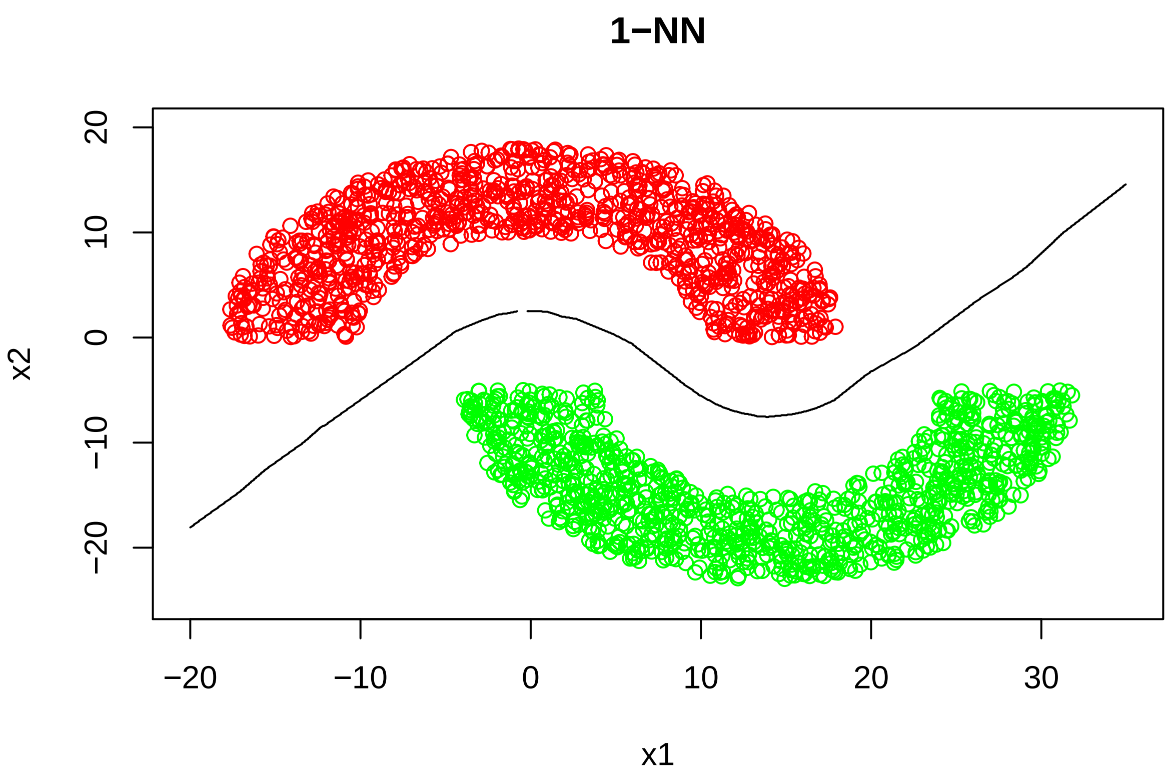
$$\dim(\text{span}(T)) = \text{rank}(T) = \text{rank}(T^T T) = \text{rank}(Q) = r.$$

Problem 6.4

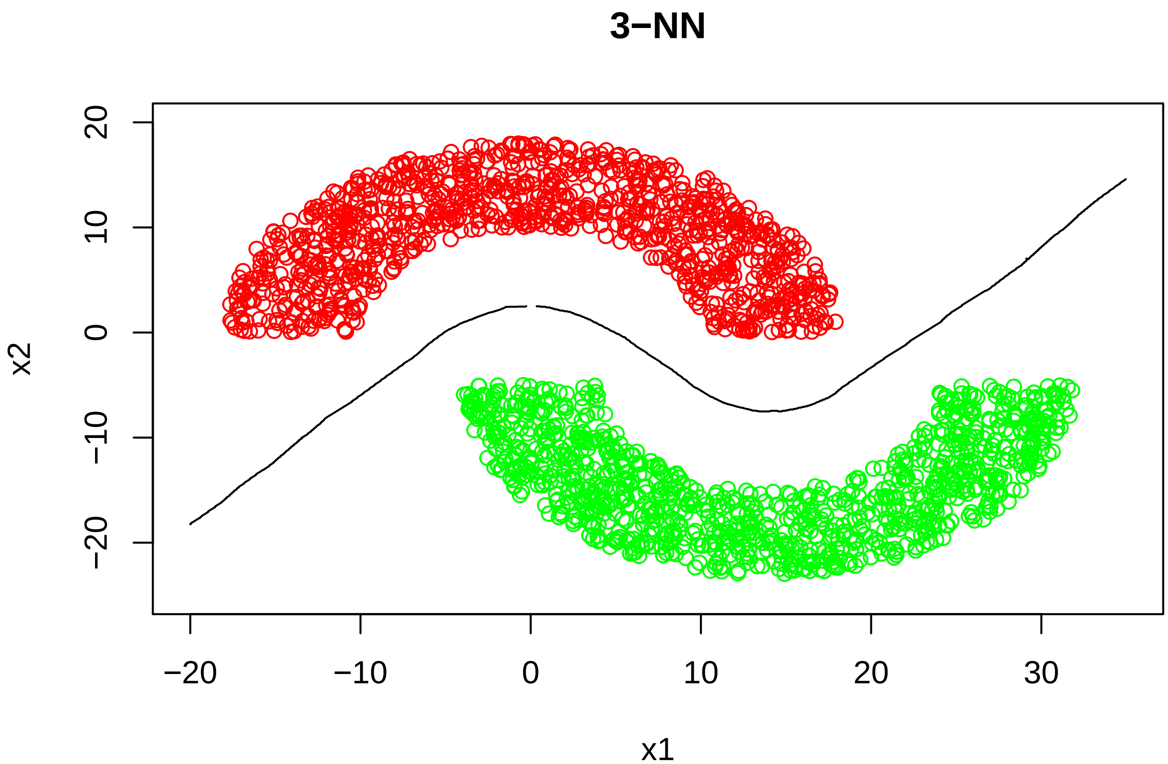
Below we plot the decision regions for the 1-NN and 3-NN rules in the case of the double semi-circle in Problem 3.1.

```
set.seed(101)
```

```r
init_data <- function(N, rad, thk, sep) {
  D <- data.frame(x = numeric(), y = numeric())
  y <- numeric()
  repeat {
    x1 <- runif(1, min = -25, max = 40)
    x2 <- runif(1, min = -30, max = 20)
    if ((x2 >= 0) && (rad^2 <= x1^2 + x2^2) && (x1^2 + x2^2 <= (rad + thk)^2)) {
      D <- rbind(D, c(x1, x2))
      y <- c(y, -1)
    }
    else if ((x2 < -sep) && (rad^2 <= (x1 - rad - thk / 2)^2 + (x2 + sep)^2) &&
             ((x1 - rad - thk / 2)^2 + (x2 + sep)^2 <= (rad + thk)^2)) {
      D <- rbind(D, c(x1, x2))
      y <- c(y, +1)
    }
    if (nrow(D) >= N)
      break
  }
  colnames(D) <- c("x1", "x2")

  return(cbind(D, y))
}

rad <- 10
thk <- 8
sep <- 5
D <- init_data(2000, rad, thk, sep)
X <- as.matrix(D[, 1:2])
xseq <- seq(-20, 35, by = 0.06)
yseq <- seq(-25, 20, by = 0.06)
Xnew <- expand.grid(xseq, yseq)
labels <- D$y

NN_1 <- knn(X, Xnew, labels, k = 1, prob = TRUE)
prob <- attr(NN_1, "prob")
prob <- ifelse(NN_1 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(xseq), length(yseq))
contour(xseq, yseq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "1-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```




```r
NN_3 <- knn(X, Xnew, labels, k = 3, prob = TRUE)
prob <- attr(NN_3, "prob")
prob <- ifelse(NN_3 == "1", prob, 1-prob)
prob1 <- matrix(prob, length(xseq), length(yseq))
contour(xseq, yseq, prob1, levels = 0.5, labels = "", xlab = "x1", ylab = "x2",
        main = "3-NN")
points(X, col = ifelse(labels == -1, "red", "green"))
```



Problem 6.5

We begin by defining the Voronoi region relative to x_n by

$$\mathcal{C}_n = \{x : d(x, x_n) \leq d(x, x_m)\ ,\ m \neq n\}.$$

We also define the *bisector of* x' *and* x'' (the set of points at equal distance of x' and x'') by

$$B(x', x'') = \{x : d(x', x) = d(x'', x)\},$$

this bisector separates the closed halfspace containing x' namely

$$D(x', x'') = \{x : d(x', x) \leq d(x'', x)\}$$

from the closed halfspace containing x'' namely $D(x'', x')$. Now we may see that

$$\cap_{m \neq n} D(x_n, x_m) = \mathcal{C}_n$$

since

$$\begin{aligned} x \in \mathcal{C}_n \quad &\Leftrightarrow \quad d(x, x_n) \leq d(x, x_m)\ \forall m \neq n \\ &\Leftrightarrow \quad x \in \cap_{m \neq n} D(x_n, x_m). \end{aligned}$$

Now, as each Voronoi region $\mathcal{C}_n$ is the intersection of a finite number of halfspaces it is also convex.

Problem 6.6

We know from Chapter 4 that for linear regression with weight decay, we have

$$w_{reg} = (X^T X + \lambda \Gamma^T \Gamma)^{-1} X^T y.$$

Consequently, we may write that

$$\begin{aligned} g(x) &= x^T(X^TX + \lambda\Gamma^T\Gamma)^{-1}X^Ty \\ &= x^T(X^TX + \lambda\Gamma^T\Gamma)^{-1}\sum_{n=1}^{N} x_n y_n \\ &= \sum_{n=1}^{N} x^T(X^TX + \lambda\Gamma^T\Gamma)^{-1}x_n y_n \\ &= \sum_{n=1}^{N} K(x, x_n) y_n \end{aligned}$$

where $K(x, x') = x^T(X^TX + \lambda\Gamma^T\Gamma)^{-1}x'$.

Problem 6.7

Since $\mathcal{H}$ is the set which contains all labeled Voronoi tesselations on K points, it is obvious that for any $N < K$ points, we are able to generate all the dichotomies on these N points with our hypothesis set $\mathcal{H}$, which means that $m_{\mathcal{H}}(N) = 2^N$ for any $N < K$. The same can be said for K points, so we get $m_{\mathcal{H}}(K) = 2^K$. However, we have no guarantee that $m_{\mathcal{H}}(N) = 2^N$ for any $N > K$. In conclusion, we have $d_{VC}(\mathcal{H}) = K$.