Problem Solutions

Chapter 1

Pierre Paquay

Problem 1.1

Let B_1 and B_2 be the events *The first picked ball is black* and *The second picked ball is black*. We have

$$\mathbb{P}(B_2|B_1) = \frac{\mathbb{P}(B_2 \cap B_1)}{\mathbb{P}(B_1)},$$

with

$$\begin{aligned} \mathbb{P}(B_1) &= \mathbb{P}(B_1|\text{Bag 1})\mathbb{P}(\text{Bag 1}) + \mathbb{P}(B_1|\text{Bag 2})\mathbb{P}(\text{Bag 2}) \\ &= 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4} \end{aligned}$$

and

$$\begin{aligned} \mathbb{P}(B_1 \cap B_2) &= \mathbb{P}(B_1 \cap B_2|\text{Bag 1})\mathbb{P}(\text{Bag 1}) + \mathbb{P}(B_1 \cap B_2|\text{Bag 2})\mathbb{P}(\text{Bag 2}) \\ &= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}. \end{aligned}$$

In conclusion, we get

$$\mathbb{P}(B_2|B_1) = \frac{1/2}{3/4} = \frac{2}{3}.$$

Problem 1.2

We have $h(x) = \text{sign}(w^T x)$ with $w = (w_0, w_1, w_2)^T$ and $x = (1, x_1, x_2)^T$.

(a) If we have $h(x) = +1$ (resp. -1), it implies that $w^T x > 0$ (resp. < 0). So, we may conclude that the separation between these two regions is the line whose equation is $w^T x = 0$ or more explicitly

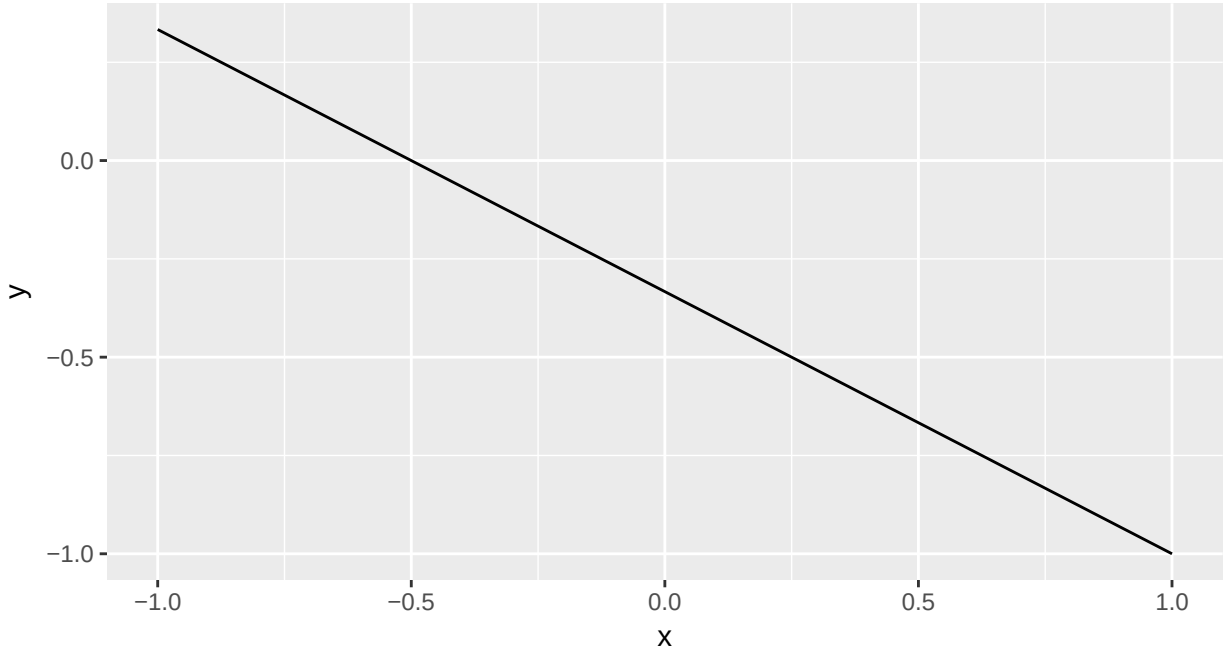
$$w_0 + w_1 x_1 + w_2 x_2 = 0.$$

We may also write this equation as

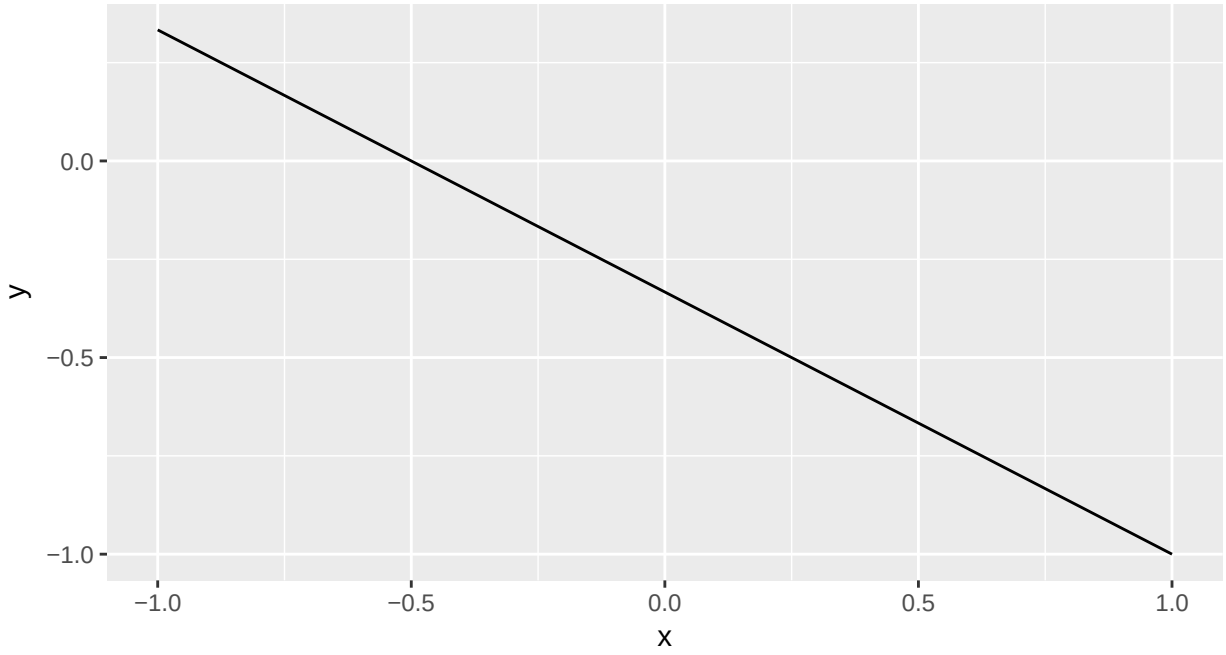
$$x_2 = a x_1 + b$$

where $a = -w_1/w_2$ and $b = -w_0/w_2$.

(b) For $w = (1, 2, 3)^T$, we have the graph below.



For $w = -(1, 2, 3)^T$, we have the graph below.



We may notice that the lines are identical in these two graphs (which is not that suprising considering that the coefficients are opposites). However, the regions where $h(x) = +1$ and $h(x) = -1$ are different; indeed in the first plot the positive region is the one above the line, and in the second plot the positive region is the one below the line.

Problem 1.3

(a) As every x_n is well classified by w^* for all $n = 1, \cdots, N$, we have that

$$y_n = \text{sign}(w^{*T} x_n),$$

which translates to

$$y_n(w^{*T}x_n) > 0$$

for all $n = 1, \cdots, N$. This implies that

$$\rho = \min_n y_n(w^{*T}x_n) > 0.$$

(b) We have that

$$\begin{aligned} w^T(t)w^* &= [w^T(t-1) + y(t-1)x^T(t-1)]w^* \\ &= w^T(t-1)w^* + y(t-1)w^{*T}x(t-1) \\ &\geq w^T(t-1)w^* + \rho. \end{aligned}$$

It remains to prove that $w^T(t)w^* \geq t\rho$, to do this we will proceed by induction. If $t = 0$, we obviously get that $0 \cdot w^* \geq 0$. If the thesis is true for $t-1$, let us prove it for t. If we use the first part of point (b), we have that

$$w^T(t)w^* \geq w^T(t-1)w^* + \rho \geq (t-1)\rho + \rho = t\rho.$$

(c) We may write that

$$\begin{aligned} ||w(t)||^2 &= ||w(t-1) + y(t-1)x(t-1)||^2 \\ &= ||w(t-1)||^2 + ||y(t-1)x(t-1)||^2 + 2y(t-1)w^T(t-1)x(t-1) \\ &\leq ||w(t-1)||^2 + ||y(t-1)x(t-1)||^2 \\ &\leq ||w(t-1)||^2 + ||x(t-1)||^2 \end{aligned}$$

as $x(t-1)$ is misclassified by $w(t-1)$ and $y(t-1)^2 = 1$.

(d) Now we prove by induction that $||w(t)||^2 \leq tR^2$ where $R = \max_n ||x_n||$. If $t = 0$, we trivially have that $0 \leq 0 \cdot R^2$. If the thesis is true for $t-1$, let us prove it for t. Because of point (c), we may write that

$$||w(t)||^2 \leq ||w(t-1)||^2 + ||x(t-1)||^2 \leq (t-1)R^2 + R^2 = tR^2.$$

(e) By using points (b) and (d), we get that

$$\frac{w^T(t)w^*}{||w(t)||} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}.$$

By using the inequality above, we may write that

$$t \leq \frac{R^2}{\rho^2}\frac{(w^T(t)w^*)^2}{||w(t)||^2} = \frac{R^2}{\rho^2}\frac{(w^T(t)w^*)^2}{||w(t)||^2||w^*||^2}||w^*||^2.$$

However, we have that

$$\frac{(w^T(t)w^*)^2}{||w(t)||^2||w^*||^2} \leq 1$$

as

$$(w^T(t)w^*)^2 \leq ||w(t)||^2||w^*||^2$$

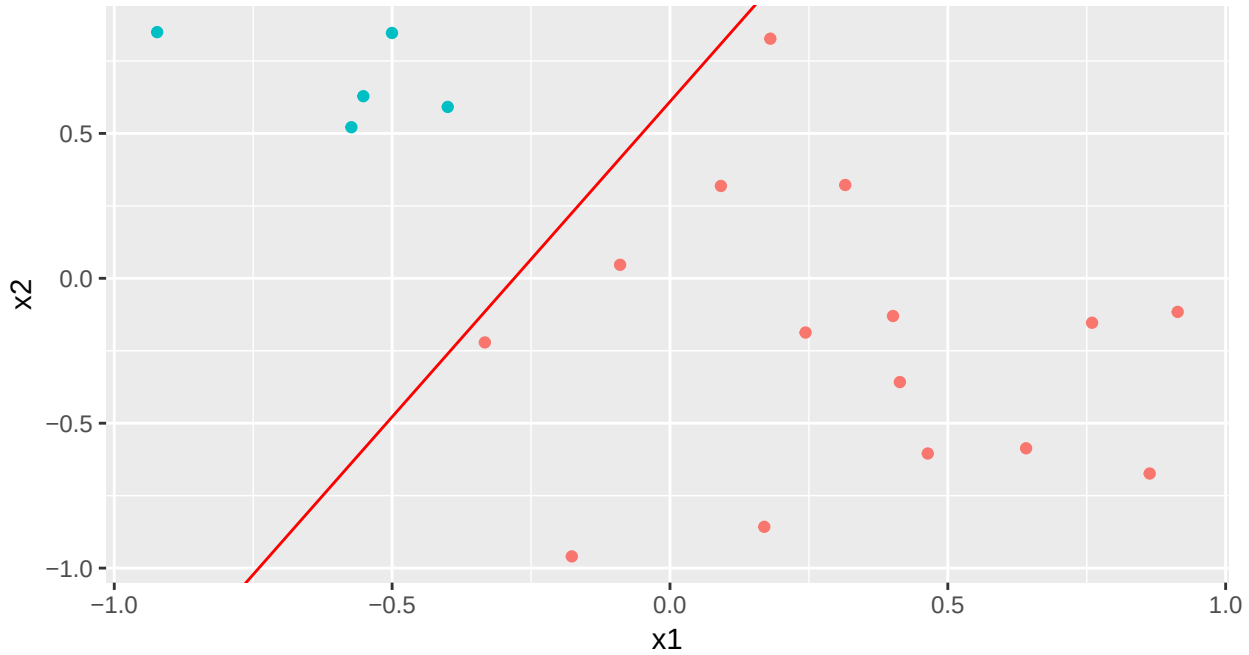
by the Cauchy-Schwarz inequality. In conclusion, we get that

$$t \leq \frac{R^2||w^*||^2}{\rho^2}.$$

Problem 1.4

(a) Below, we generate a linearly separable data set of size 20 and the target function f (in red).

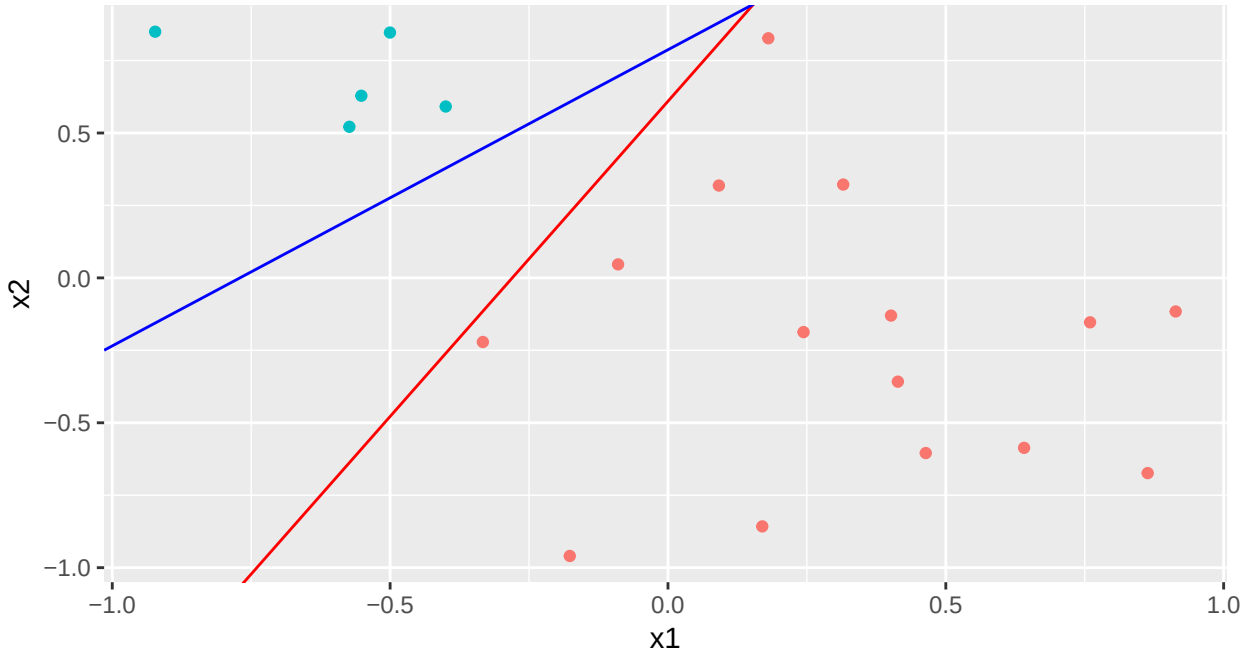
```
set.seed(101)

h <- function(x, w) {
  scalar_prod <- cbind(1, x$x1, x$x2) %*% w

  return(as.vector(sign(scalar_prod)))
}

w0 <- runif(1, min = -999, max = 999)
w1 <- runif(1, min = -999, max = 999)
w2 <- runif(1, min = -999, max = 999)

f <- function(x) {
  return(h(x, c(w0, w1, w2)))
}

D <- data.frame(x1 = runif(20, min = -1, max = 1), x2 = runif(20, min = -1, max = 1))
D <- cbind(D, y = f(D))

p <- ggplot(D, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none")

p_f <- p + geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red")
p_f
```



(b) Below, we plot the training data, the target function f (in red) and the final hypothesis g (in blue) generated by PLA.

```
iter <- 0
w <- c(0, 0, 0)
```

```
repeat {
  y_pred <- h(D, w)
  D_mis <- subset(D, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  x_t <- D_mis[1, ]
  w <- w + c(1, x_t$x1, x_t$x2) * x_t$y
  iter <- iter + 1
}

p_g <- p_f + geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
p_g
```

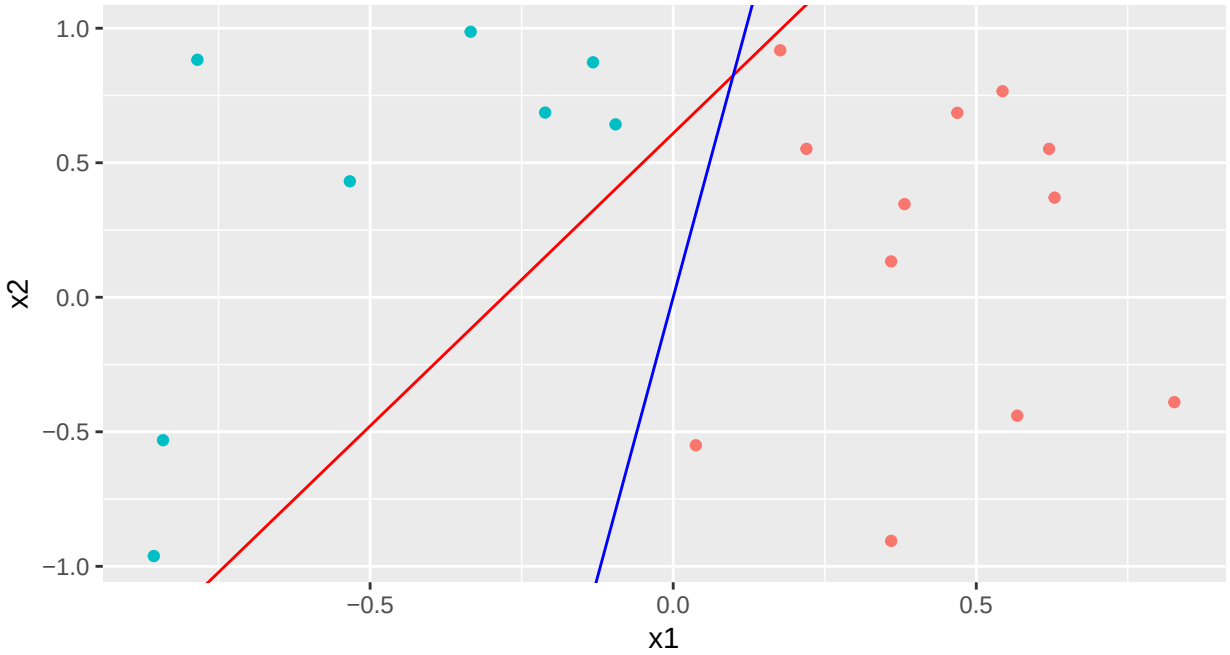


Here, the PLA took 5 iterations before converging. We may notice that although g is pretty close to f, they are not quite identical.

(c) Below, we repeat what we did in point (b) with another randomly generated data set of size 20.

```
D1 <- data.frame(x1 = runif(20, min = -1, max = 1), x2 = runif(20, min = -1, max = 1))
D1 <- cbind(D1, y = f(D1))

iter <- 0
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D1, w)
  D_mis <- subset(D1, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  x_t <- D_mis[1, ]
  w <- w + c(1, x_t$x1, x_t$x2) * x_t$y
  iter <- iter + 1
}

ggplot(D1, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
```

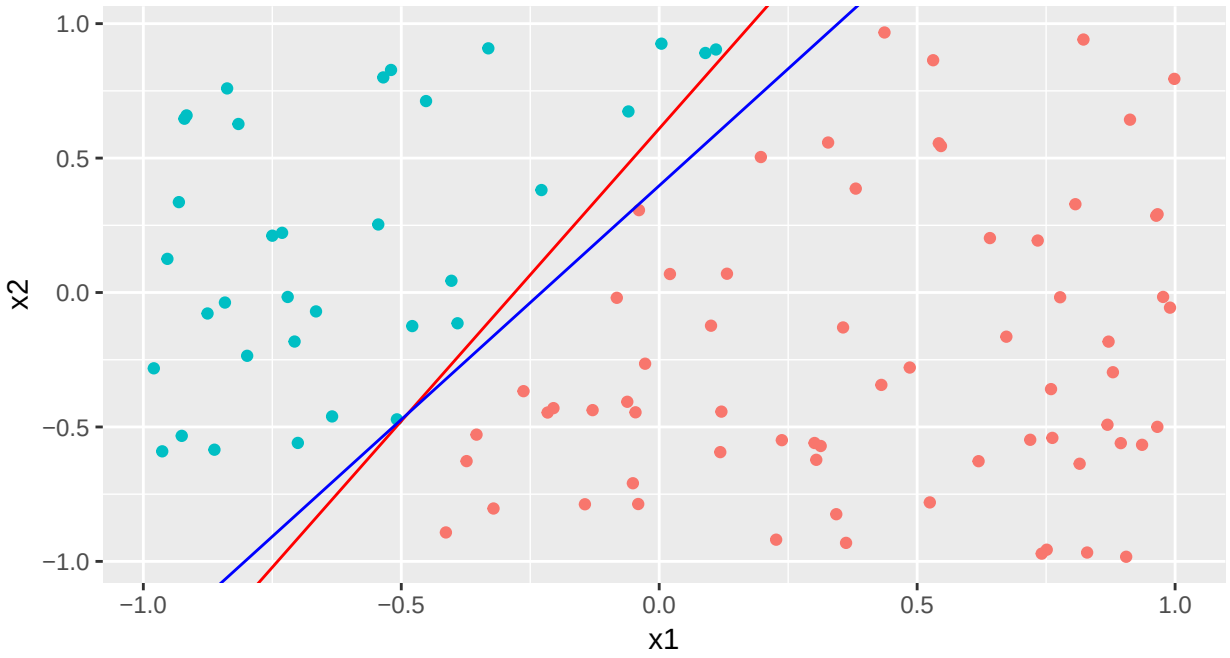
```
  theme(legend.position = "none") +
  geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
```



In this case, the PLA took 12 iterations (which is greater than in (b)) before converging. We may notice that, as in point (b), although g is pretty close to f, they are not quite identical.

(d) Below, we repeat what we did in point (b) with another randomly generated data set of size 100.

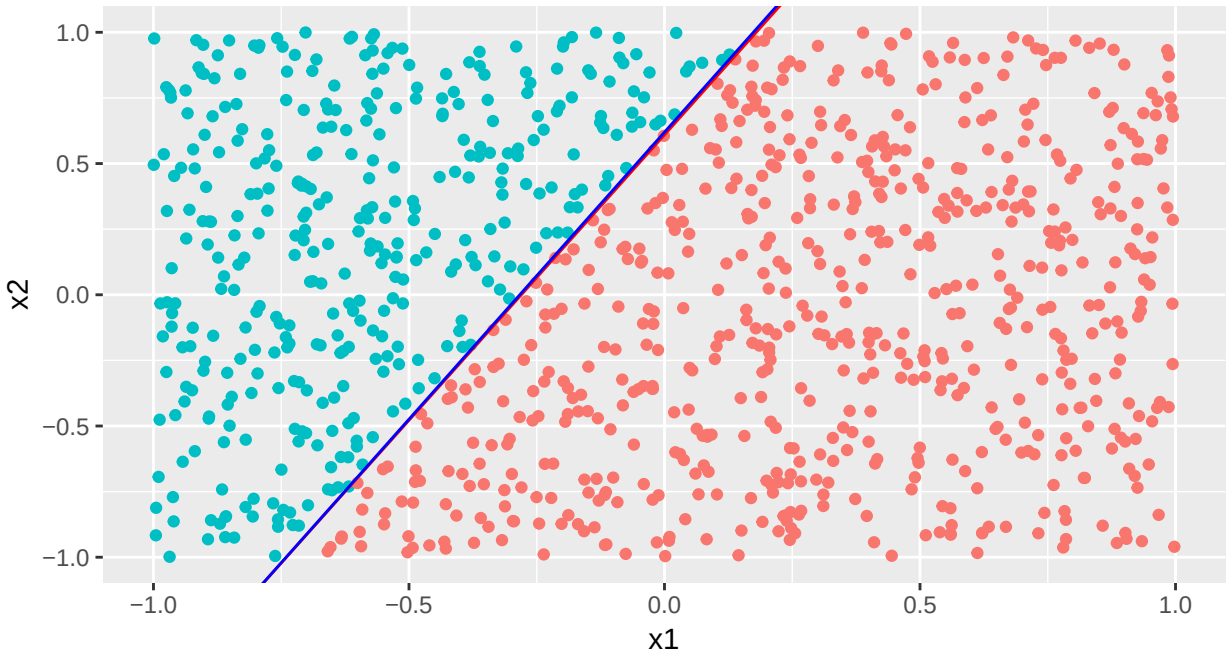
```
D1 <- data.frame(x1 = runif(100, min = -1, max = 1), x2 = runif(100, min = -1, max = 1))
D1 <- cbind(D1, y = f(D1))

iter <- 0
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D1, w)
  D_mis <- subset(D1, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  x_t <- D_mis[1, ]
  w <- w + c(1, x_t$x1, x_t$x2) * x_t$y
  iter <- iter + 1
}

ggplot(D1, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none") +
  geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
```



In this case, the PLA took 33 iterations (which is greater than in (b) and (c)) before converging. We may notice that, here f and g are very close to each other.

(e) Below, we repeat what we did in point (b) with another randomly generated data set of size 1000.

```
D1 <- data.frame(x1 = runif(1000, min = -1, max = 1), x2 = runif(1000, min = -1, max = 1))
D1 <- cbind(D1, y = f(D1))

iter <- 0
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D1, w)
  D_mis <- subset(D1, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  x_t <- D_mis[1, ]
  w <- w + c(1, x_t$x1, x_t$x2) * x_t$y
  iter <- iter + 1
}

ggplot(D1, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none") +
  geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
```



In this case, the PLA took 511 iterations (which is greater than in (b), (c) and (d)) before converging. We may notice that, here f and g are nearly undistinguishable.

(f) Here, we randomly generate a linearly separable data set of size 1000 with $x_n \in \mathbb{R}^{10}$.

```
N <- 10

h <- function(x, w) {
  scalar_prod <- cbind(1, x) %*% w

  return(as.vector(sign(scalar_prod)))
}

w <- runif(N + 1)

f <- function(x) {
  return(h(x, w))
}

D2 <- matrix(runif(10000, min = -1, max = 1), ncol = N)
D2 <- cbind(D2, y = f(D2))
D2 <- data.frame(D2)

iter <- 0
w0 <- rep(0, N + 1)
repeat {
  y_pred <- h(as.matrix(D2[, 1:N]), as.numeric(w0))
  D_mis <- subset(D2, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  x_t <- D_mis[1, ]
  w0 <- w0 + cbind(1, x_t[, 1:N]) * x_t$y
  iter <- iter + 1
}
```

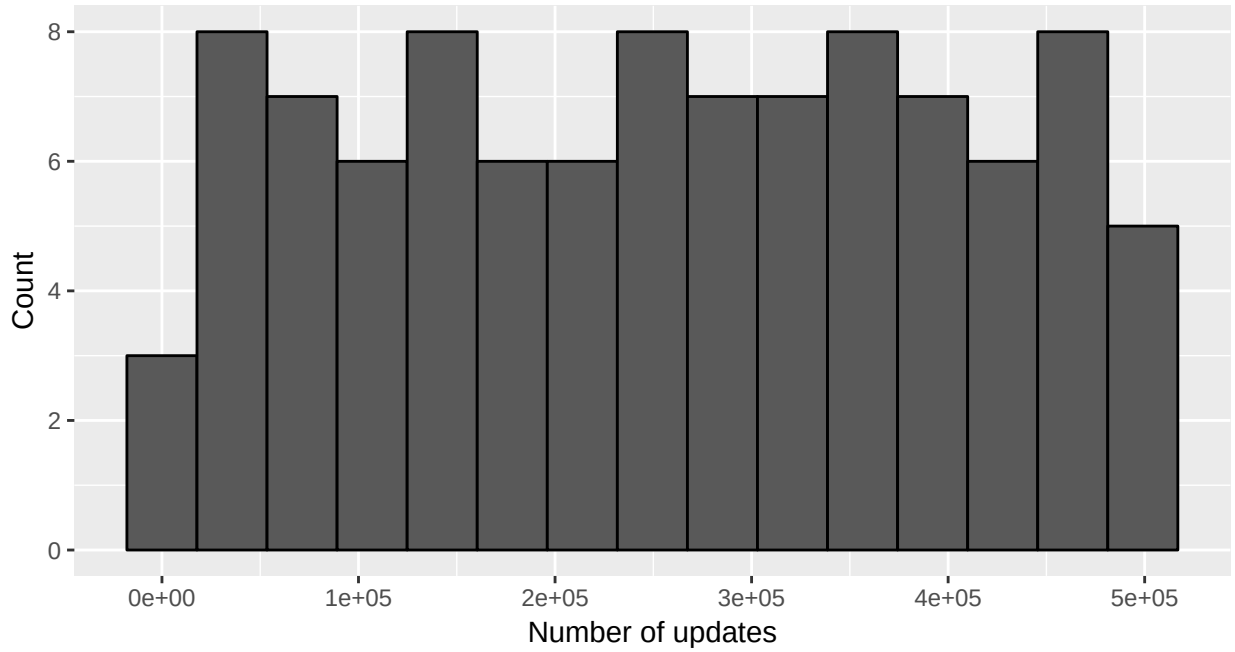
In this case, we may see that the number of iterations is 4326 which is very big, this is a direct consequence of the increase in dimensions from 2 to 10.

(g) Below, we repeat the algorithm on the same data set as (f) for 100 experiments, and we pick $x(t)$ randomly instead of deterministically.

```
updates <- rep(0, 100)
iter <- 0
for (i in 1:100) {
  w0 <- rep(0, N + 1)
  repeat {
    y_pred <- h(as.matrix(D2[, 1:N]), as.numeric(w0))
    D_mis <- subset(D2, y != y_pred)
    if (nrow(D_mis) == 0)
      break
    x_t <- D_mis[sample(nrow(D_mis), 1), ]
    w0 <- w0 + cbind(1, x_t[, 1:N]) * x_t$y
    iter <- iter + 1
  }
  updates[i] <- iter
}
```

Now, we plot a histogram of the number of updates that the PLA takes to converge.

```
ggplot(data.frame(updates), aes(x = updates)) + geom_histogram(bins = 15, col = "black") +
  labs(x = "Number of updates", y = "Count")
```



We may see that the distribution of the number of updates seems pretty uniform and varies from 0 to 5.0437×10^5.

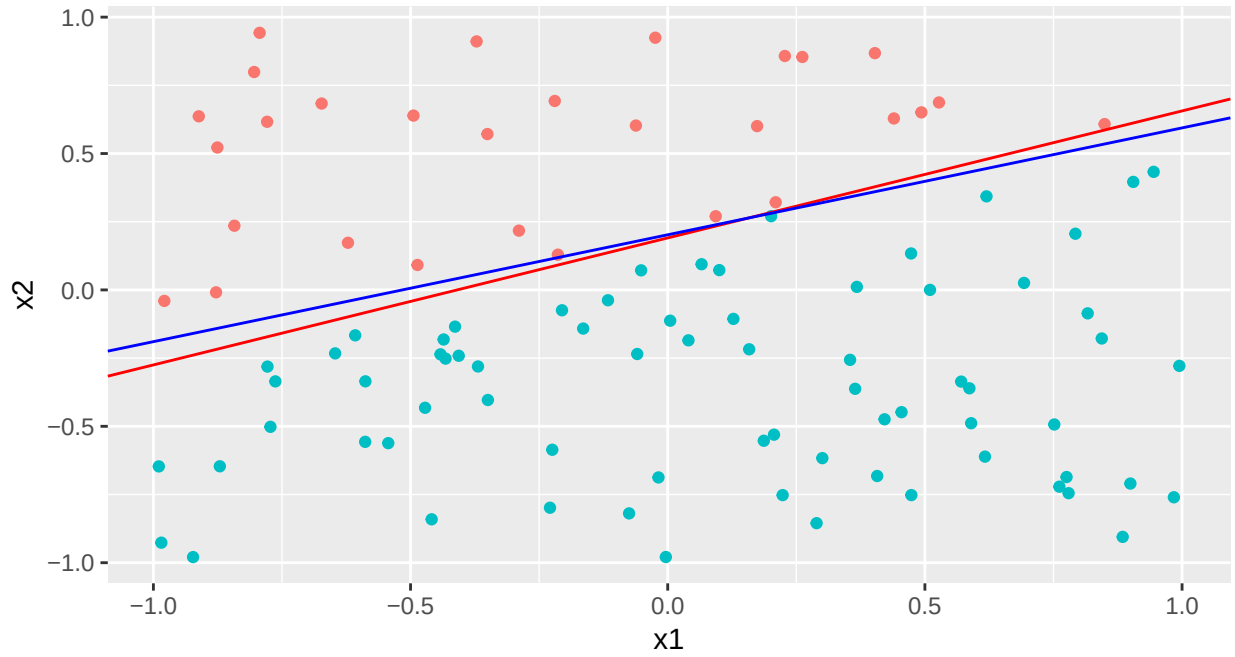
(h) As we saw above, the more data points we have (N), the more accurate g becomes in approximating f and the greater the running time gets. Moreover, the greater d becomes, the greater the running time gets also.

Problem 1.5

(a) Below, we generate a training data set of size 100 and a test data set of size 10000. We also plot the target function f (in red) and the final hypothesis g (in blue) generated by Adaline *[We use a η value of 5 instead of 100 to simplify the values computed].*

```
set.seed(1975)

h <- function(x, w) {
  scalar_prod <- cbind(1, x$x1, x$x2) %*% w

  return(as.vector(sign(scalar_prod)))
}

w0 <- runif(1, min = -999, max = 999)
w1 <- runif(1, min = -999, max = 999)
w2 <- runif(1, min = -999, max = 999)

f <- function(x) {
  return(h(x, c(w0, w1, w2)))
}

D_train <- data.frame(x1 = runif(100, min = -1, max = 1), x2 = runif(100, min = -1, max = 1))
D_train <- cbind(D_train, y = f(D_train))

D_test <- data.frame(x1 = runif(10000, min = -1, max = 1), x2 = runif(10000, min = -1, max = 1))
D_test <- cbind(D_test, y = f(D_test))

iter <- 0
eta <- 5
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D_train, w)
  D_mis <- subset(D_train, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  obs_t <- D_mis[sample(nrow(D_mis), 1), ]
  x_t <- c(1, as.numeric(obs_t[1:2]))
  y_t <- as.numeric(obs_t[3])
  s_t <- sum(w * x_t)
  if (y_t * s_t <= 1)
    w <- w + eta * (y_t - s_t) * x_t
  iter <- iter + 1
  if (iter == 1000)
    break
}

test_error <- mean(h(D_test, w) != D_test$y)

p <- ggplot(D_train, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none")

p_g <- p + geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
```

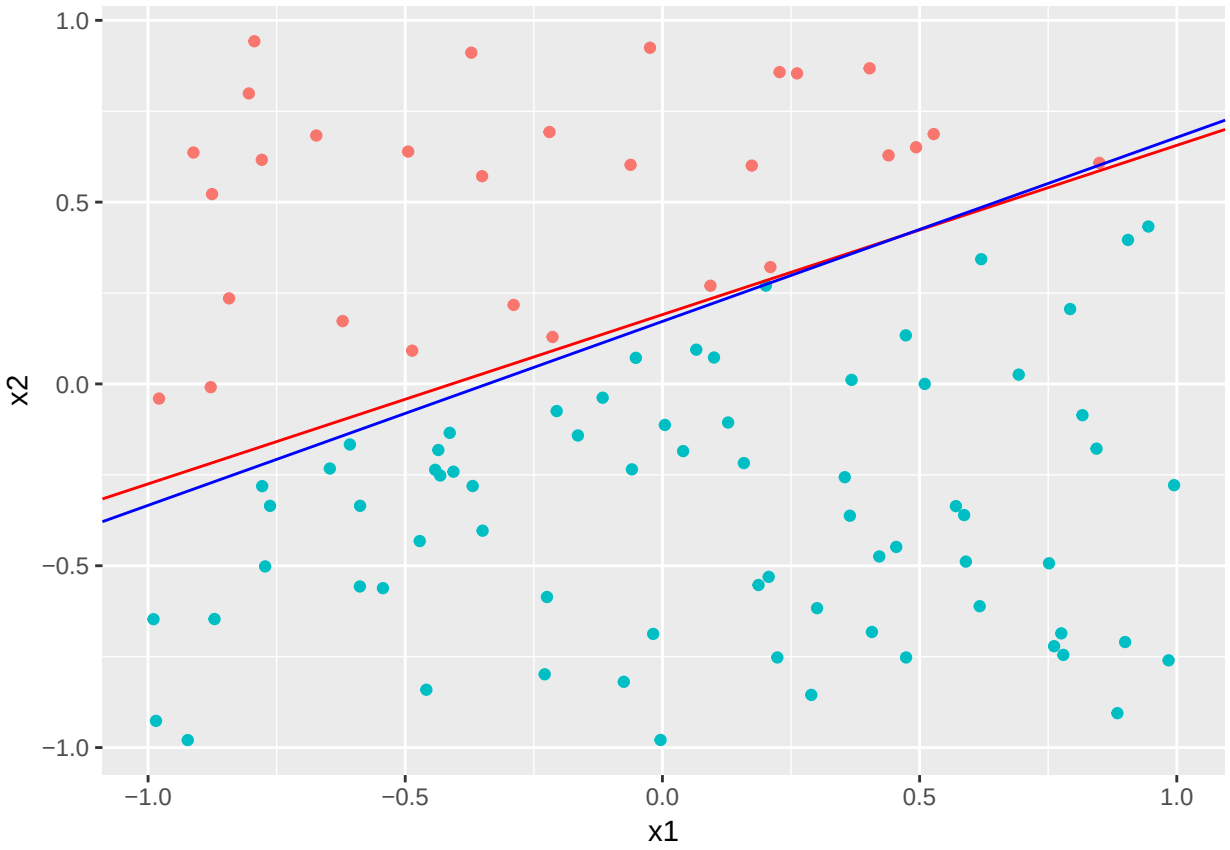
```
p_g
```



We have a classification error rate of 2.23% on the test set.

(b) Now we repeat everything we did in (a) with $\eta = 1$.

```
iter <- 0
eta <- 1
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D_train, w)
  D_mis <- subset(D_train, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  obs_t <- D_mis[sample(nrow(D_mis), 1), ]
  x_t <- c(1, as.numeric(obs_t[1:2]))
  y_t <- as.numeric(obs_t[3])
  s_t <- sum(w * x_t)
  if (y_t * s_t <= 1)
    w <- w + eta * (y_t - s_t) * x_t
  iter <- iter + 1
  if (iter == 1000)
    break
}

test_error <- mean(h(D_test, w) != D_test$y)

p <- ggplot(D_train, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none")

p_g <- p + geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
p_g
```

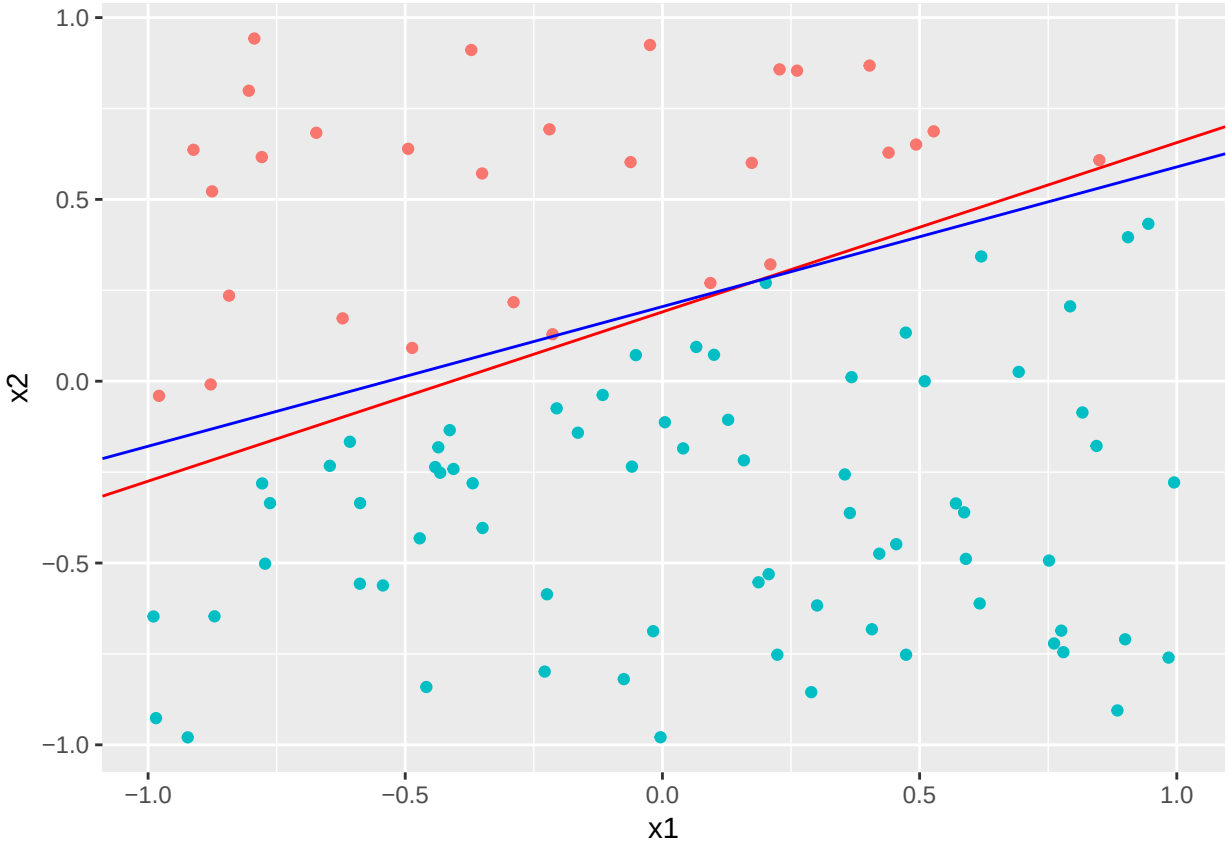


We may see that the classification error rate has now decreased to 1.23% on the test set.

(c) Now we repeat everything we did in (a) with $\eta = 0.01$.

```
iter <- 0
eta <- 0.01
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D_train, w)
  D_mis <- subset(D_train, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  obs_t <- D_mis[sample(nrow(D_mis), 1), ]
  x_t <- c(1, as.numeric(obs_t[1:2]))
  y_t <- as.numeric(obs_t[3])
  s_t <- sum(w * x_t)
  if (y_t * s_t <= 1)
    w <- w + eta * (y_t - s_t) * x_t
  iter <- iter + 1
  if (iter == 1000)
    break
}

test_error <- mean(h(D_test, w) != D_test$y)

p <- ggplot(D_train, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none")
```

```
p_g <- p + geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
p_g
```

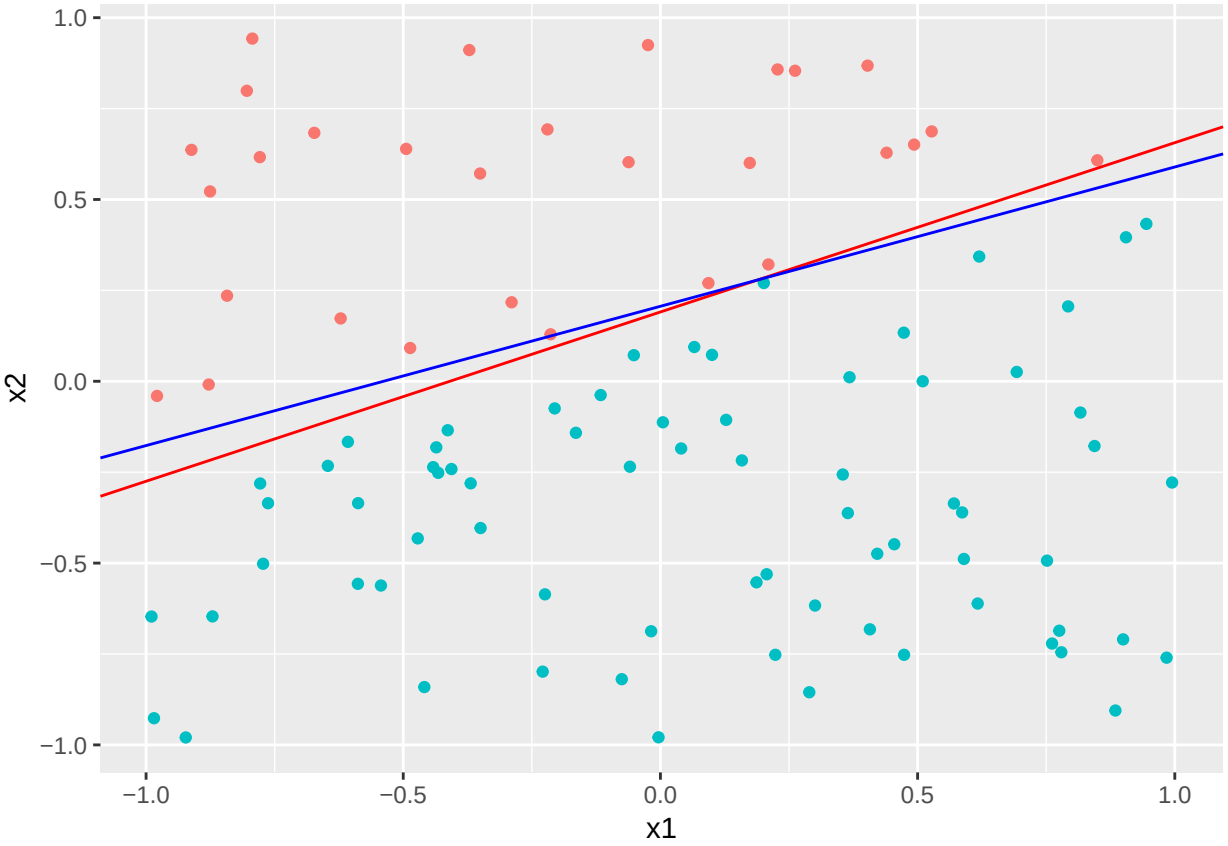


We may see that the classification error rate has now increased to 2.43% on the test set.

(d) Now we repeat everything we did in (a) with $\eta = 0.0001$.

```
iter <- 0
eta <- 0.0001
w <- c(0, 0, 0)
repeat {
  y_pred <- h(D_train, w)
  D_mis <- subset(D_train, y != y_pred)
  if (nrow(D_mis) == 0)
    break
  obs_t <- D_mis[sample(nrow(D_mis), 1), ]
  x_t <- c(1, as.numeric(obs_t[1:2]))
  y_t <- as.numeric(obs_t[3])
  s_t <- sum(w * x_t)
  if (y_t * s_t <= 1)
    w <- w + eta * (y_t - s_t) * x_t
  iter <- iter + 1
  if (iter == 1000)
    break
}

test_error <- mean(h(D_test, w) != D_test$y)
```

```r
p <- ggplot(D_train, aes(x = x1, y = x2, col = as.factor(y + 3))) + geom_point() +
  theme(legend.position = "none")

p_g <- p + geom_abline(slope = -w1 / w2, intercept = -w0 / w2, colour = "red") +
  geom_abline(slope = -w[2] / w[3], intercept = -w[1] / w[3], colour = "blue")
p_g
```



We may see that the classification error rate has now increased to 2.5% on the test set.

(*e*) We may conclude that the η value that results in the minimum classification error rate on the test set is actually 1.

Problem 1.6

(*a*) For one sample we have that

$$\mathbb{P}(\nu = 0) = (1 - \mu)^{10}.$$

So for $\mu = 0.05$, we get $\mathbb{P}(\nu = 0) = 0.5987369$, for $\mu = 0.5$, we get $\mathbb{P}(\nu = 0) = 9.765625 \times 10^{-4}$, and for $\mu = 0.8$, we get $\mathbb{P}(\nu = 0) = 1.024 \times 10^{-7}$.

(*b*) Now, for 1000 independant samples, we have that

$$\begin{aligned}
\mathbb{P}(\text{At least one sample has } \nu = 0) &= 1 - \mathbb{P}(\nu_i > 0 \ \forall i) \\
&= 1 - \prod_{i=1}^{1000} \mathbb{P}(\nu_i > 0) \\
&= 1 - \prod_{i=1}^{1000} [1 - \mathbb{P}(\nu_i = 0)] \\
&= 1 - \prod_{i=1}^{1000} [1 - (1-\mu)^{10}] \\
&= 1 - [1 - (1-\mu)^{10}]^{1000}.
\end{aligned}$$

Which gives us 1 when $\mu = 0.05$, 0.6235762 when $\mu = 0.5$, and 1.0239476×10^{-4} when $\mu = 0.8$.

(c) Here, we repeat (b) for 1000000 independant samples. In that case, we obtain

$$\mathbb{P}(\text{At least one sample has } \nu = 0) = 1 - [1 - (1-\mu)^{10}]^{1000000}.$$

Which gives us 1 when $\mu = 0.05$, 1 when $\mu = 0.5$, and 0.0973316 when $\mu = 0.8$.

Problem 1.7

(a) First we treat the case where $\mu = 0.05$, for one coin, we have that

$$\mathbb{P}(\text{At least one coin has } \nu = 0) = (1 - 0.05)^{10} = 0.5987369;$$

for 1000 coins, we have

$$\mathbb{P}(\text{At least one coin has } \nu = 0) = 1 - [1 - (1 - 0.05)^{10}]^{1000} = 1;$$

and finally for 1000000 coins we get

$$\mathbb{P}(\text{At least one coin has } \nu = 0) = 1 - [1 - (1 - 0.05)^{10}]^{1000000} = 1.$$

We repeat the same reasoning for $\mu = 0.8$, for one coin, we have that

$$\mathbb{P}(\text{At least one coin has } \nu = 0) = (1 - 0.8)^{10} = 1.024 \times 10^{-7};$$

for 1000 coins, we have

$$\mathbb{P}(\text{At least one coin has } \nu = 0) = 1 - [1 - (1 - 0.8)^{10}]^{1000} = 1.0239476 \times 10^{-4};$$

and finally for 1000000 coins we get

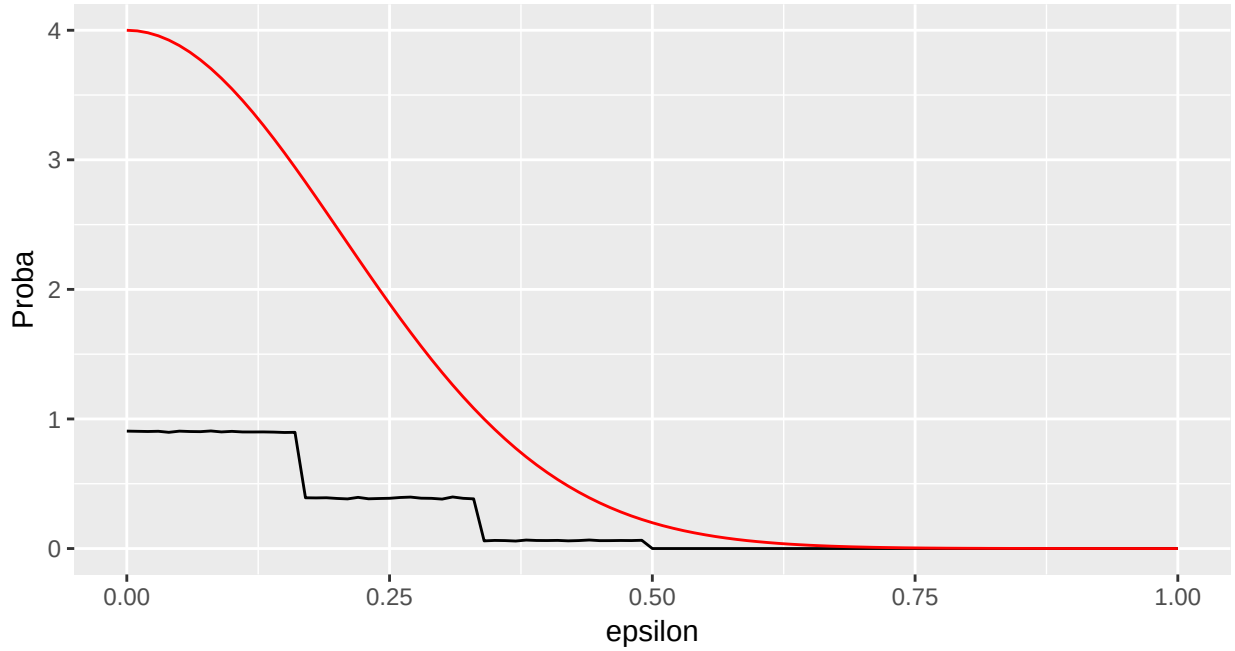
$$\mathbb{P}(\text{At least one coin has } \nu = 0) = 1 - [1 - (1 - 0.8)^{10}]^{1000000} = 0.0973316.$$

(b) Here, we consider $N = 6$, two coins, and $\mu = 0.5$. If we use the Hoeffding inequality bound, we obtain that

$$\begin{aligned}
\mathbb{P}(\max_i |\nu_i - \mu_i| > \epsilon) &= \mathbb{P}(|\nu_1 - \mu_1| > \epsilon \text{ or } |\nu_2 - \mu_2| > \epsilon) \\
&= \mathbb{P}(|\nu_1 - \mu_1| > \epsilon) + \mathbb{P}(|\nu_2 - \mu_2| > \epsilon) - \mathbb{P}(|\nu_1 - \mu_1| > \epsilon \text{ and } |\nu_2 - \mu_2| > \epsilon) \\
&= \mathbb{P}(|\nu_1 - \mu_1| > \epsilon) + \mathbb{P}(|\nu_2 - \mu_2| > \epsilon) - \mathbb{P}(|\nu_1 - \mu_1| > \epsilon)\mathbb{P}(|\nu_2 - \mu_2| > \epsilon) \\
&\leq 4e^{-12\epsilon^2}.
\end{aligned}$$

Below, we plot the above probability with its Hoeffding inequality bound for ϵ in the range $[0, 1]$.

```r
N <- 6
mu <- 0.5
max_abs <- function(epsilon) {
  c1 <- sample(c(0, 1), N, replace = TRUE)
  nu1 <- mean(c1)
  c2 <- sample(c(0, 1), N, replace = TRUE)
  nu2 <- mean(c2)
  m <- max(abs(nu1 - mu), abs(nu2 - mu))

  return(m > epsilon)
}
proba <- function(epsilon) mean(replicate(10000, max_abs(epsilon = epsilon)))
proba <- Vectorize(proba)
ggplot(data.frame(epsilon = c(0, 1)), aes(x = epsilon)) +
  stat_function(fun = proba, geom = "line") +
  stat_function(fun = function(epsilon) 4 * exp(-12 * epsilon^2), geom = "line", col = "red") +
  labs(y = "Proba")
```



Problem 1.8

(a) If t is a non-negative random variable and $\alpha > 0$, we may write that

$$\alpha I_{\{t \geq \alpha\}} \leq t$$

as in the case where $t \geq \alpha$, we have $\alpha \cdot 1 \leq t$, and in the case where $t < \alpha$, we have $\alpha \cdot 0 \leq t$. As the expectation is a non-decreasing function, we get

$$\mathbb{E}(\alpha I_{\{t \geq \alpha\}}) \leq \mathbb{E}(t).$$

As we also have that

$$\mathbb{E}(\alpha I_{\{t \geq \alpha\}}) = \alpha \mathbb{P}(t \geq \alpha) + 0 \cdot \mathbb{P}(t < \alpha) = \alpha \mathbb{P}(t \geq \alpha),$$

we finally get

$$\mathbb{P}(t \geq \alpha) \leq \frac{\mathbb{E}(t)}{\alpha}$$

which proves the *Markov Inequality*.

(b) Here u is a random variable with mean μ and variance σ^2 and $\alpha > 0$. If we consider the random variable $(u-\mu)^2 \geq 0$, the Markov Inequality tells us that

$$\mathbb{P}[(u-\mu)^2 \geq \alpha] \leq \frac{\mathbb{E}[(u-\mu)^2]}{\alpha} = \frac{\sigma^2}{\alpha}$$

which proves the *Chebyshev Inequality*.

(c) Now $u_1, \cdots, u_N$ are iid random variables each with mean μ and variance σ^2, $u = \frac{1}{N}\sum_{n=1}^{N} u_n$ and $\alpha > 0$. We consider the random variable u, its expectation is

$$\mathbb{E}(u) = \frac{1}{N}\sum_{n=1}^{N} \mu = \mu$$

and its variance is

$$\mathrm{Var}(u) = \frac{1}{N^2}\sum_{n=1}^{N} \sigma^2 = \frac{\sigma^2}{N}.$$

It suffices now to apply the Chebyshev Inequality to u to obtain that

$$\mathbb{P}[(u-\mu)^2 \geq \alpha] \leq \frac{\sigma^2}{N\alpha}.$$

Problem 1.9

(a) Let t be a finite random variable, $\alpha > 0$ and $s > 0$, we may write that

$$\begin{aligned} \mathbb{P}(t \geq \alpha) &= \mathbb{P}(st \geq s\alpha) \\ &= \mathbb{P}(e^{st} \geq e^{s\alpha}) \\ &\leq \frac{\mathbb{E}(e^{st})}{e^{s\alpha}} = e^{-s\alpha}T(s) \end{aligned}$$

because of the *Markov Inequality*.

(b) Here $u_1, \cdots, u_N$ are iid random variables and $u = \frac{1}{N}\sum_{n=1}^{N} u_n$. We have successively that

$$\begin{aligned} \mathbb{P}(u \geq \alpha) &= \mathbb{P}(Nu \geq N\alpha) \\ &\leq e^{-sN\alpha}\mathbb{E}(e^{sNu}) \\ &= e^{-sN\alpha}\prod_{n=1}^{N}\mathbb{E}(e^{su_n}) \\ &= (e^{-sN\alpha}U(s))^N. \end{aligned}$$

(c) In this case, we may write that

$$U(s) = \mathbb{E}(e^{su_n}) = e^{s\cdot 0}\mathbb{P}(u_n = 0) + e^{s\cdot 1}\mathbb{P}(u_n = 1) = 1\cdot\frac{1}{2} + e^s\cdot\frac{1}{2} = \frac{1}{2}(1+e^s)$$

Let

$$f(s) = e^{-s\alpha}\frac{1}{2}(1+e^s),$$

we get immediately

$$\frac{df}{ds} = \frac{e^{-s\alpha}}{2}[(1-\alpha)e^s - \alpha],$$

which has a root for $s = \ln(\alpha/(1-\alpha))$, and

$$\frac{d^2 f}{ds^2} = \frac{e^{-s\alpha}}{2}[e^s(\alpha-1)^2 + \alpha] \geq 0.$$

So, we conclude that $s = \ln(\alpha/(1-\alpha))$ is a minimum of $f(s)$.

$$-\ln\frac{1/2+\epsilon}{1/2-\epsilon}(1/2+\epsilon)U(\ln\frac{1/2+\epsilon}{1/2-\epsilon})$$

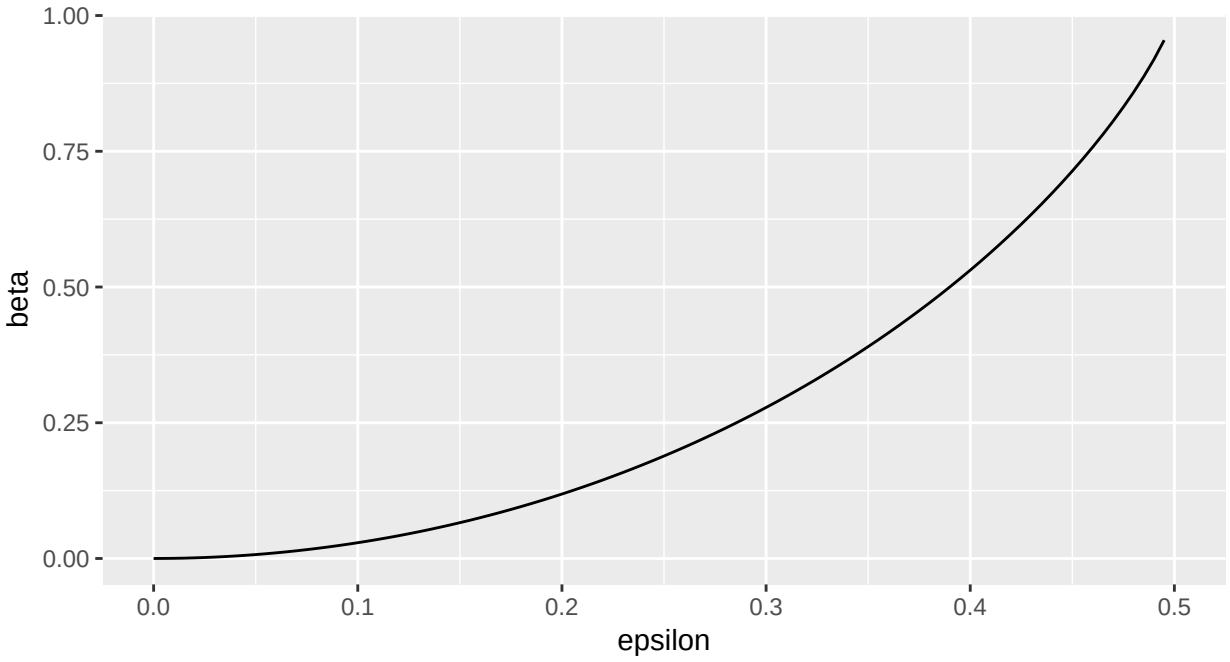
(d) First, we notice that

$$\mathbb{E}(u) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}(u_n) = \frac{1}{N}\sum_{n=1}^{N}(0\cdot\frac{1}{2} + 1\cdot\frac{1}{2}) = \frac{1}{2}.$$

Now, let $0 < \epsilon < \frac{1}{2}$, we may write that

$$\begin{aligned}
\mathbb{P}\left(u \geq \frac{1}{2}+\epsilon\right) &\leq (e^{-s(\frac{1}{2}+\epsilon)}U(s))^N \\
&\leq \min_s (e^{-s(\frac{1}{2}+\epsilon)}U(s))^N \\
&= \left(e^{-\ln\frac{1/2+\epsilon}{1/2-\epsilon}(1/2+\epsilon)}U\left(\ln\frac{1/2+\epsilon}{1/2-\epsilon}\right)\right)^N \\
&= (e^{-\ln\frac{1/2+\epsilon}{1/2-\epsilon}(1/2+\epsilon)}\frac{1}{2}(1+e^{\ln\frac{1/2+\epsilon}{1/2-\epsilon}}))^N \\
&= \left[\frac{1}{2}\left(\frac{1/2-\epsilon}{1/2+\epsilon}\right)^{1/2+\epsilon}\left(1+\frac{1/2+\epsilon}{1/2-\epsilon}\right)\right]^N \\
&= \left[\frac{1}{2}\left(\frac{1/2-\epsilon}{1/2+\epsilon}\right)^{1/2+\epsilon}\left(\frac{1}{2}-\epsilon\right)^{-1}\right]^N \\
&= \left[2^{-1}\frac{1}{(1/2+\epsilon)^{1/2+\epsilon}}\frac{1}{(1/2-\epsilon)^{1/2-\epsilon}}\right]^N \\
&= \left[2^{-1-\log_2(1/2+\epsilon)^{1/2+\epsilon}-\log_2(1/2-\epsilon)^{1/2-\epsilon}}\right]^N \\
&= 2^{-N[1+(1/2+\epsilon)\log_2(1/2+\epsilon)+(1/2-\epsilon)\log_2(1/2-\epsilon)]} = 2^{-\beta N}.
\end{aligned}$$

It remains to see that $\beta > 0$, to do that, we plot the graph of β for $0 < \epsilon < 1/2$ below.



Problem 1.10

(a) In this case, we may write that

$$E_{off}(h,f) = \frac{1}{M}\sum_{m=1}^{M}[[h(x_{N+m}) \neq f(x_{N+m})]] = \frac{1}{M}[\text{Number of odd } m \text{ between 1 and } M],$$

which is equal to $1/2$ if M is even, and to $1/2 + 1/2M$ if M is odd.

(b) If $\mathcal{D}$ is fixed of size N, there are 2^N target functions f that can generate $\mathcal{D}$ in a noiseless setting.

(c) We must have that

$$\sum_{m=1}^{M}[[h(x_{N+m}) \neq f(x_{N+m})]] = k,$$

which is equivalent to the fact that the number of $h(x_{N+m}) \neq f(x_{N+m})$ must be equal to k. And the number of ways we have k errors in M trials is C_M^k (the binomial coefficient).

(d) We may write that

$$\begin{aligned}\mathbb{E}_f[E_{off}(h,f)] &= \frac{1}{M}\mathbb{E}_f[\text{Number of } h(x_{N+m}) \neq f(x_{N+m})] \\ &= \frac{1}{M}\cdot M \cdot \frac{1}{2^M} = \frac{1}{2^M}\end{aligned}$$

as we have here a binomial distribution.

Problem 1.11

For the supermarket, we have

$$\begin{aligned} E_{in}^{(S)}(h) &= \frac{1}{N}\sum_{n=1}^{N} e(h(x_n), f(x_n)) \\ &= \frac{1}{N}[\sum_{y_n=1} e(h(x_n), 1) + \sum_{y_n=-1} e(h(x_n), -1)] \\ &= \frac{1}{N}[\sum_{y_n=1} 10 \cdot [[h(x_n) \neq 1]] + \sum_{y_n=-1} [[h(x_n) \neq -1]]. \end{aligned}$$

And for the CIA, we have

$$\begin{aligned} E_{in}^{(C)}(h) &= \frac{1}{N}\sum_{n=1}^{N} e(h(x_n), f(x_n)) \\ &= \frac{1}{N}[\sum_{y_n=1} e(h(x_n), 1) + \sum_{y_n=-1} e(h(x_n), -1)] \\ &= \frac{1}{N}[\sum_{y_n=1} [[h(x_n) \neq 1]] + \sum_{y_n=-1} 1000 \cdot [[h(x_n) \neq -1]]. \end{aligned}$$

Problem 1.12

(a) To minimize $E_{in}(h) = \sum_{n=1}^{N}(h - y_n)^2$, we have to find the stationary points of this function. We immediately find that

$$\frac{dE_{in}(h)}{dh} = 2\sum_{n=1}^{N}(h - y_n) = 0$$

implies $h = \frac{1}{N}\sum_{n=1}^{N} y_n = h_{mean}$. It is actually a minimum as we have that

$$\frac{d^2E_{in}(h)}{dh^2} = 2N > 0.$$

(b) We proceed in the same fashion to minimize $E_{in}(h) = \sum_{n=1}^{N}|h - y_n|$, so we get that

$$\frac{dE_{in}(h)}{dh} = \sum_{n=1}^{N}(h - y_n) = 0$$

implies $h = \text{median}\{y_1, \cdots, y_n\} = h_{med}$ as the derivative is equal to zero only if the number of positive terms is equal to the number of negative terms.

(c) In the case where y_N becomes an outlier, we get that $h_{mean} \to \infty$ and h_{med} remains unchanged.