perceptron

Joyce Robbins

8/13/2018

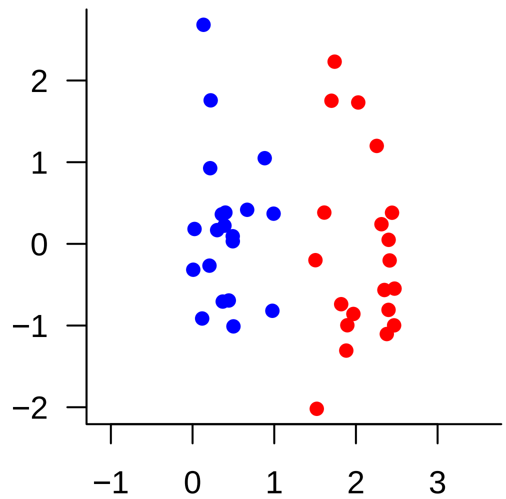
The perceptron is a simple algorithm that learns to classify inputs into two classes by adjusting the weights (w) in the equation $y_i = \text{sign}(w_i x_i)$ until all inputs in a training set are correctly classified. Here the steps of algorithm will be presented visually in two-dimensional space.

The basics

We start by plotting (x_1, x_2), coloring each point by class. Note that the points can be separated by a line; if this is not the case, the algorithm won't work.

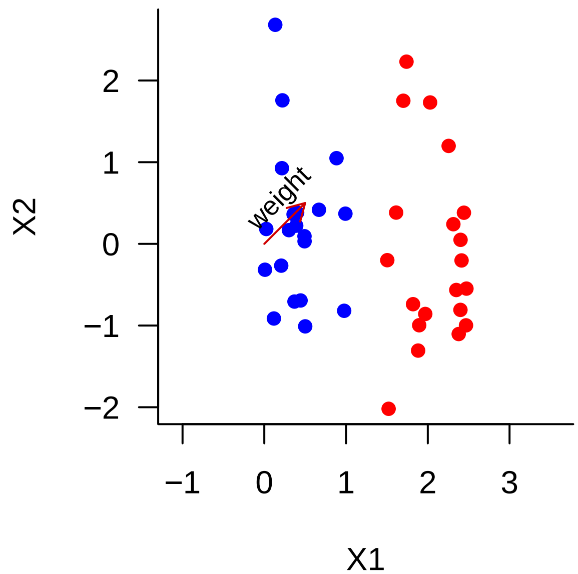
```
#devtools::install_github("jtr13/perceptron")
library(perceptron)
set.seed(9)
X <- matrix(c(runif(20), runif(20)+1.5, rnorm(40)), nrow = 40)
Y <- c(rep(-1, 20), rep(1, 20))
```

```
X <- cbind(rep(1, length(Y)), X)
W <- c(0, .5, .5)
weight_matrix <- matrix(W, 1)
i <- 1
converged <- FALSE
set.seed(1)
par(xpd = TRUE)
draw_points(X[,2], X[,3], Y, axes = TRUE)
```



We start with an arbitrary weight vector, $(w0, w1, w2)$. Often $(0, 0, 0)$ is used, but we'll start with (0, 0.5, 0.5) so we can visualize it:

```
s <- shift(W)
draw_points(X[,2], X[,3], Y, axes = TRUE)
mtext("X1", side = 1, line = 3)
mtext("X2", side = 2, line = 3)
draw_weight_vector(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], "weight")
```

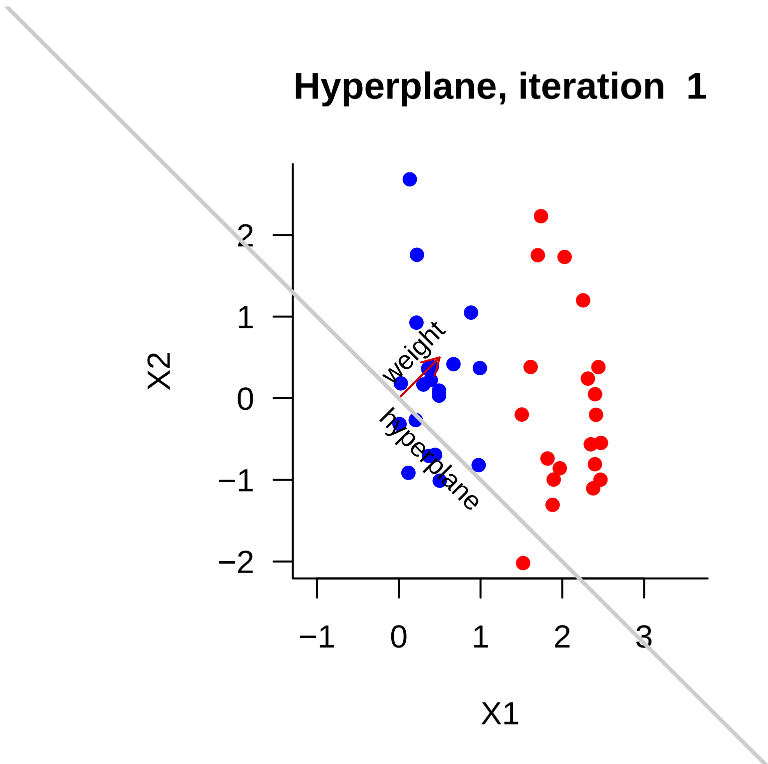


The decision boundary, or hyperplane, is the line orthogonal to the weight vector. For points on the line, the sign of $(w_i x_i)$ equals zero. On one side of the line, the sign of $(w_i x_i)$ is greater than zero whereas on the other side the sign of $(w_i x_i)$ is less than zero; hence the line serves to divide all points into two classes according to the perceptron logic.

```
all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
if (length(all.mis.points) == 0) converged <- TRUE
if (!converged) {
  all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
  index <- sample(nrow(all.mis.points), 1)
  mis.point <- all.mis.points[index,]
  mis.point.Y <- all.mis.points.Y[index]
}
```

```
s <- shift(W)
draw_points(X[,2], X[,3], Y, axes = TRUE)
mtext("X1", side = 1, line = 3)
mtext("X2", side = 2, line = 3)
draw_weight_vector(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], "weight")
draw_hyperplane(W)
e <- get_endpoints(W)
if (sum(is.na(e)) == 0) {
  label_vector(e[1], e[2], e[3], e[4],
               label = "hyperplane",
               "below")
}
all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
if (length(all.mis.points) == 0) converged <- TRUE
```

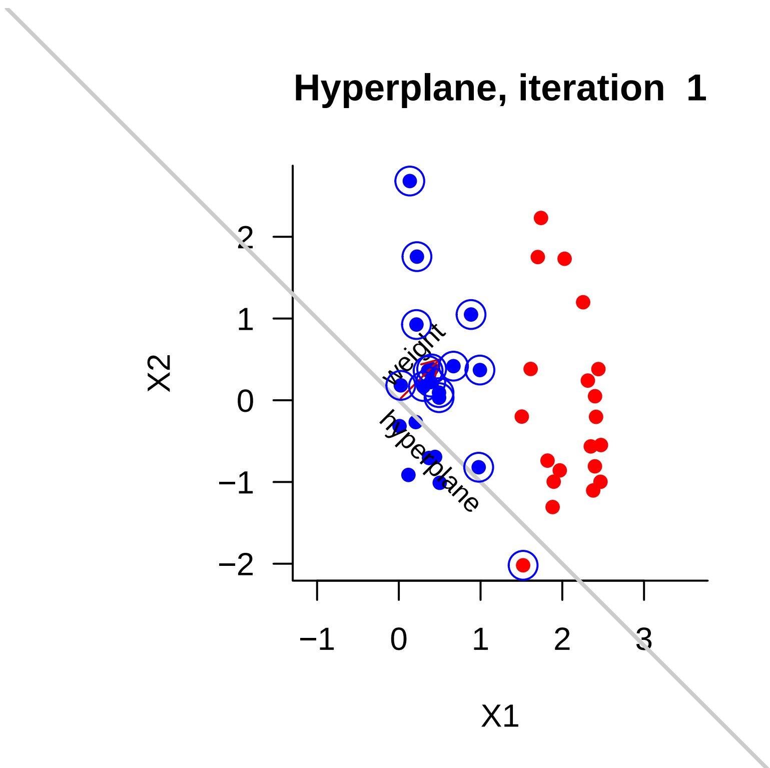
```
if (!converged) {
  all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
  index <- sample(nrow(all.mis.points), 1)
  mis.point <- all.mis.points[index,]
  mis.point.Y <- all.mis.points.Y[index]
}
if (!converged) {
  title(paste("Hyperplane, iteration ", i))
} else {
    title(paste("\nConverged! Iteration ", i))
}
```



Note the circled points – these are the misclassified points – the ones for which $y_i \neq \text{sign}(w_i x_i)$.

```
s <- shift(W)
draw_points(X[,2], X[,3], Y, axes = TRUE)
mtext("X1", side = 1, line = 3)
mtext("X2", side = 2, line = 3)
draw_weight_vector(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], "weight")
draw_hyperplane(W)
e <- get_endpoints(W)
if (sum(is.na(e)) == 0) {
  label_vector(e[1], e[2], e[3], e[4],
               label = "hyperplane",
               "below")
}
all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
```

```r
if (length(all.mis.points) == 0) converged <- TRUE
if (!converged) {
  all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
  index <- sample(nrow(all.mis.points), 1)
  mis.point <- all.mis.points[index,]
  mis.point.Y <- all.mis.points.Y[index]
}
if (!converged) {
  title(paste("Hyperplane, iteration ", i))
} else {
    title(paste("\nConverged! Iteration ", i))
}
points(all.mis.points[,2],
       all.mis.points[,3],
       col = "blue", cex = 2)
```



The Algorithm

The perceptron algorithm works by updating the weight vector based on a randomly selected misclassified point, calculating the new hyperplane, and repeating until the hyperplane separates all points into the two classes.

The formula for the new weight vector is:

$w_{t+1} = w_t + \eta y_i x_i$, where

x_i = the misclassified point

y_i = the true label of the misclassified point (-1 or 1)

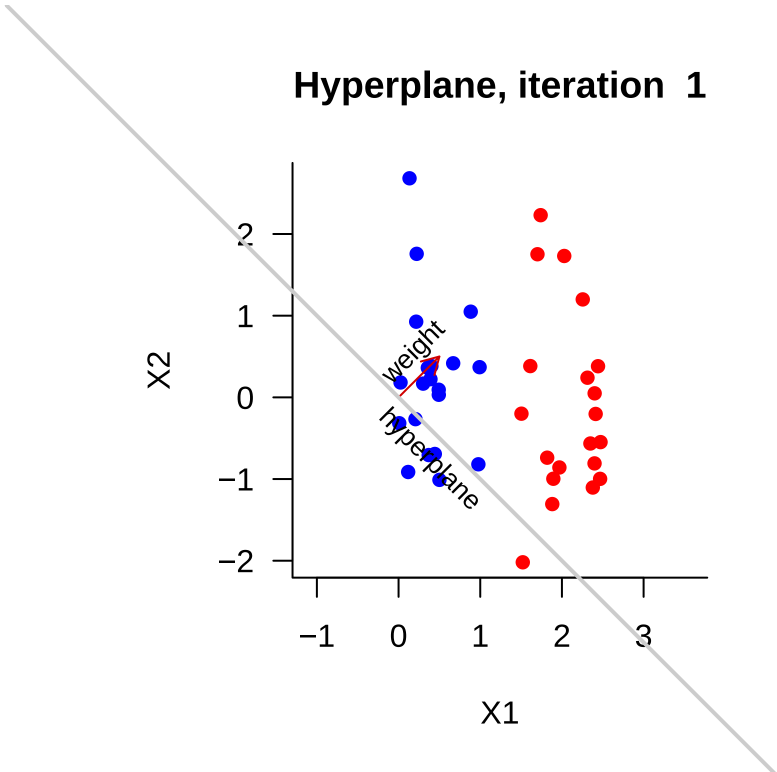
η = the learning rate, which we'll set to 1 for the sake of simplicity

Visually, the new weight vector, w_{t+1}, is determined by adding $y_i x_i$ to w_t and then shifting by the offset $w_0/||w||_2$.

We'll go through the algorithm one step at a time.

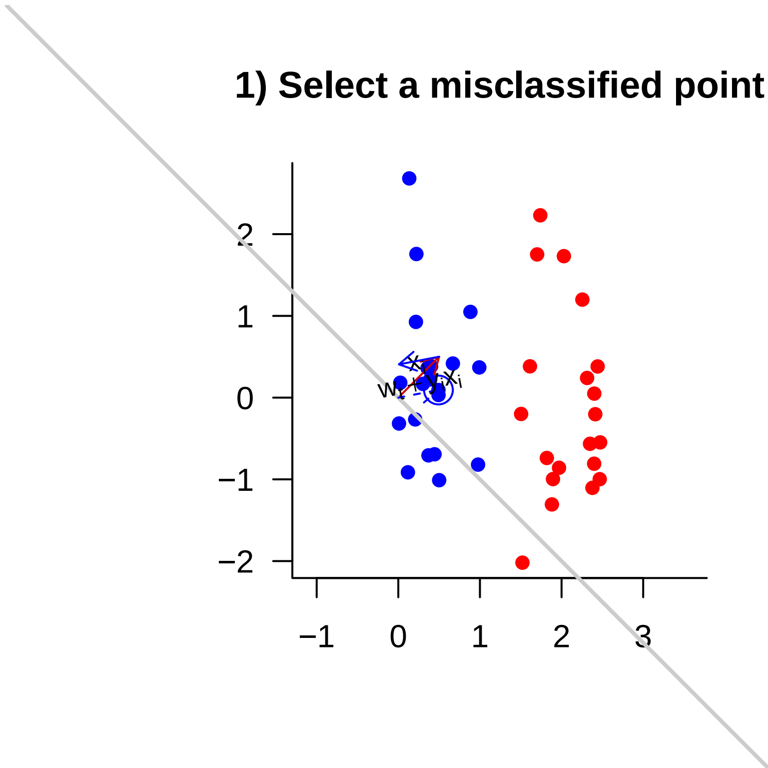
We begin with our original weight vector and hyperplane:

```
s <- shift(W)
draw_points(X[,2], X[,3], Y, axes = TRUE)
mtext("X1", side = 1, line = 3)
mtext("X2", side = 2, line = 3)
draw_weight_vector(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], "weight")
draw_hyperplane(W)
e <- get_endpoints(W)
if (sum(is.na(e)) == 0) {
  label_vector(e[1], e[2], e[3], e[4],
               label = "hyperplane",
               "below")
}
all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
if (length(all.mis.points) == 0) converged <- TRUE
if (!converged) {
  all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
  index <- sample(nrow(all.mis.points), 1)
  mis.point <- all.mis.points[index,]
  mis.point.Y <- all.mis.points.Y[index]
}
if (!converged) {
  title(paste("Hyperplane, iteration ", i))
} else {
    title(paste("\nConverged! Iteration ", i))
}
```



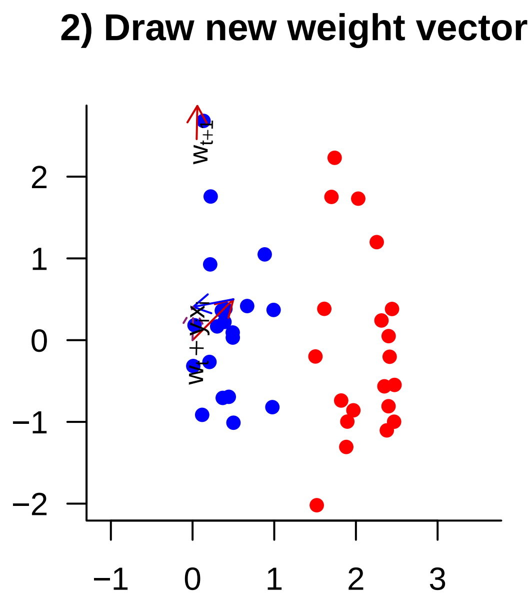
Next we randomly select a misclassified point. In the diagram below, x_i is shown as a **dashed blue arrow**, and $y_i x_i$ added to w_t as a **solid blue arrow**:

```
draw_points(X[,2], X[,3], Y, axes = TRUE)
title("1) Select a misclassified point")
draw_weight_vector(W)
draw_hyperplane(W)
circle_point(mis.point)
draw_mis_vector(mis.point)
label_vector(0, 0, mis.point[2],
             mis.point[3],
             expression(x[i]))
draw_mis_vector_added(W, mis.point,
                      mis.point.Y)
s <- shift(W)
label_vector(W[2] + s["x"],
             W[3] + s["y"],
             W[2] + s["x"] + mis.point.Y *
               mis.point[2],
             W[3] + s["y"] + mis.point.Y *
               mis.point[3],
             expression(w[t] + y[i]*x[i]),
             "below")
```



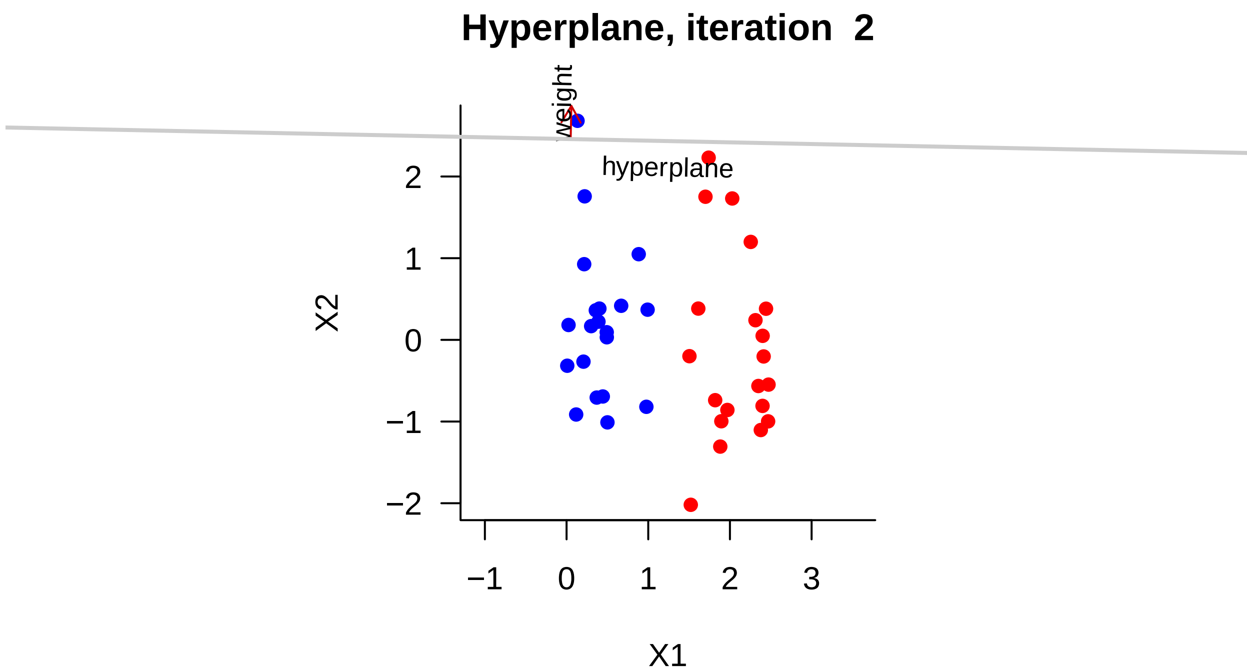
Next we determine the new weight vector by shifting the vector sum by $w_0/||w||_2$:

```
draw_points(X[,2], X[,3], Y, axes = TRUE)
title("2) Draw new weight vector")
draw_weight_vector(W)
draw_mis_vector_added(W, mis.point, mis.point.Y)
draw_new_weight_vector(W, mis.point, mis.point.Y)
label_vector(s["x"],
             s["y"],
             W[2] + s["x"] + mis.point.Y * mis.point[2],
             W[3] + s["y"] + mis.point.Y * mis.point[3], expression(w[t] + y[i]*x[i]), "below")

# calculate new weight vector
W <- W + mis.point.Y %*% mis.point
weight_matrix <- rbind(weight_matrix, matrix(W, 1))
i <- i + 1
draw_weight_vector(W)
s <- shift(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], expression(w[t+1]),
             "below")
```



Finally, we draw the new hyperplane:

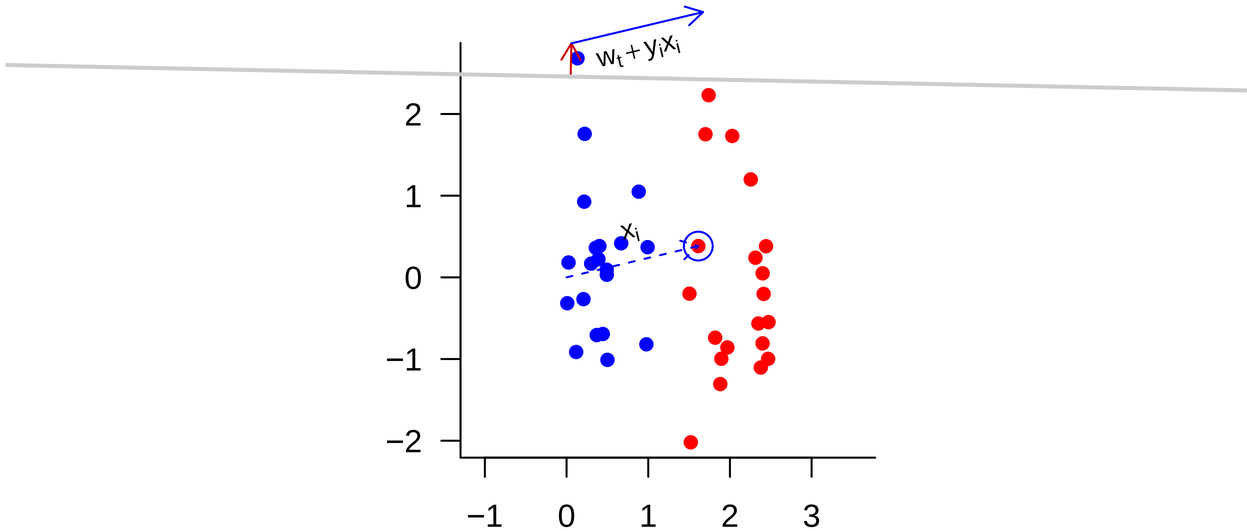
```
s <- shift(W)
draw_points(X[,2], X[,3], Y, axes = TRUE)
mtext("X1", side = 1, line = 3)
mtext("X2", side = 2, line = 3)
draw_weight_vector(W)
label_vector(s["x"], s["y"], W[2] + s["x"],
             W[3] + s["y"], "weight")
draw_hyperplane(W)
e <- get_endpoints(W)
if (sum(is.na(e)) == 0) {
  label_vector(e[1], e[2], e[3], e[4],
               label = "hyperplane",
               "below")
}
all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
if (length(all.mis.points) == 0) converged <- TRUE
if (!converged) {
  all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
  index <- sample(nrow(all.mis.points), 1)
  mis.point <- all.mis.points[index,]
  mis.point.Y <- all.mis.points.Y[index]
}
if (!converged) {
  title(paste("Hyperplane, iteration ", i))
} else {
    title(paste("\nConverged! Iteration ", i))
}
```



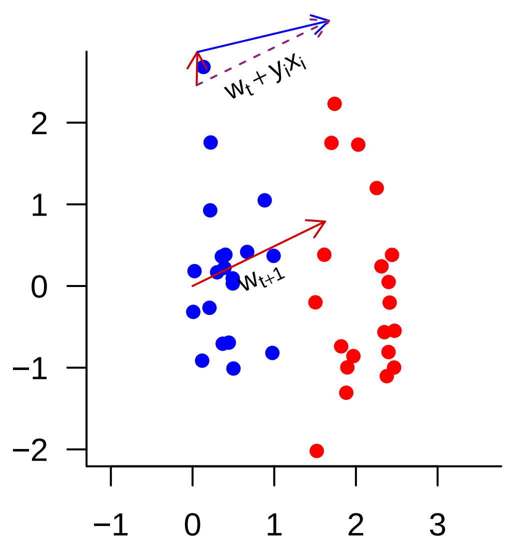

```
while (!converged) {
  draw_points(X[,2], X[,3], Y, axes = TRUE)
  title("1) Select a misclassified point")
  draw_weight_vector(W)
  draw_hyperplane(W)
  circle_point(mis.point)
  draw_mis_vector(mis.point)
  label_vector(0, 0, mis.point[2],
               mis.point[3],
               expression(x[i]))
  draw_mis_vector_added(W, mis.point,
                        mis.point.Y)
  s <- shift(W)
  label_vector(W[2] + s["x"],
               W[3] + s["y"],
               W[2] + s["x"] + mis.point.Y *
                 mis.point[2],
               W[3] + s["y"] + mis.point.Y *
                 mis.point[3],
               expression(w[t] + y[i]*x[i]),
               "below")
  draw_points(X[,2], X[,3], Y, axes = TRUE)
  title("2) Draw new weight vector")
  draw_weight_vector(W)
  draw_mis_vector_added(W, mis.point, mis.point.Y)
  draw_new_weight_vector(W, mis.point, mis.point.Y)
  label_vector(s["x"],
               s["y"],
               W[2] + s["x"] + mis.point.Y * mis.point[2],
```

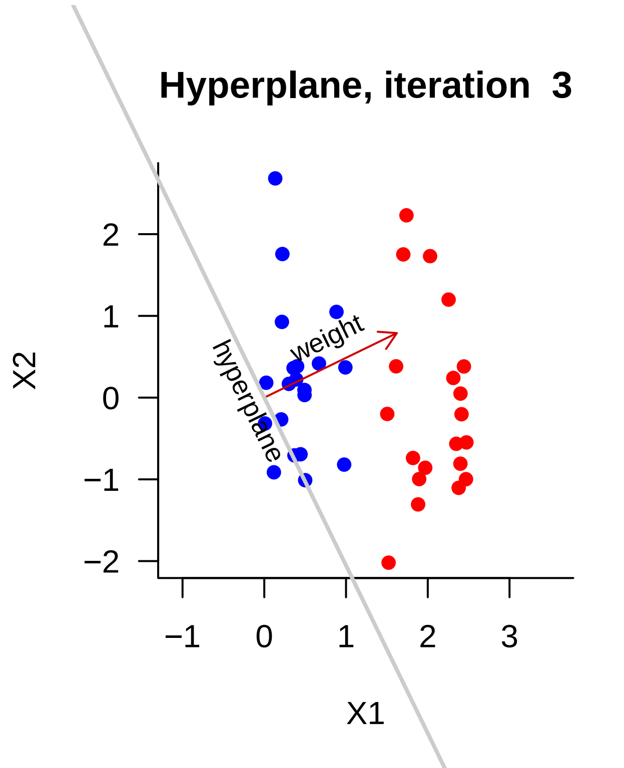
```
               W[3] + s["y"] + mis.point.Y * mis.point[3], expression(w[t] + y[i]*x[i]), "below")

  # calculate new weight vector
  W <- W + mis.point.Y %*% mis.point
  weight_matrix <- rbind(weight_matrix, matrix(W, 1))
  i <- i + 1
  draw_weight_vector(W)
  s <- shift(W)
  label_vector(s["x"], s["y"], W[2] + s["x"],
               W[3] + s["y"], expression(w[t+1]),
               "below")
  s <- shift(W)
  draw_points(X[,2], X[,3], Y, axes = TRUE)
  mtext("X1", side = 1, line = 3)
  mtext("X2", side = 2, line = 3)
  draw_weight_vector(W)
  label_vector(s["x"], s["y"], W[2] + s["x"],
               W[3] + s["y"], "weight")
  draw_hyperplane(W)
  e <- get_endpoints(W)
  if (sum(is.na(e)) == 0) {
    label_vector(e[1], e[2], e[3], e[4],
                 label = "hyperplane",
                 "below")
  }
  all.mis.points <- X[sign(W %*% t(X)) != Y, , drop = F]
  if (length(all.mis.points) == 0) converged <- TRUE
  if (!converged) {
    all.mis.points.Y <- Y[sign(W %*% t(X)) != Y]
    index <- sample(nrow(all.mis.points), 1)
    mis.point <- all.mis.points[index,]
    mis.point.Y <- all.mis.points.Y[index]
  }
  if (!converged) {
    title(paste("Hyperplane, iteration ", i))
  } else {
      title(paste("\nConverged! Iteration ", i))
  }
}
```

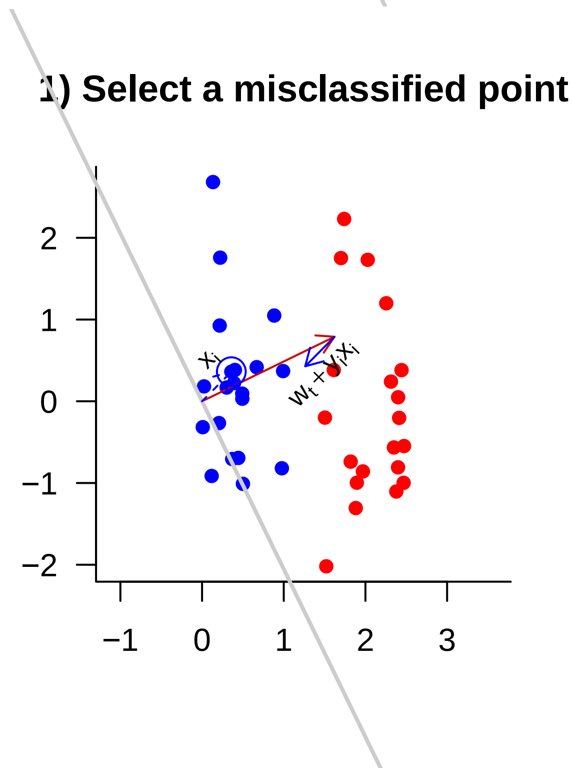
1) Select a misclassified point

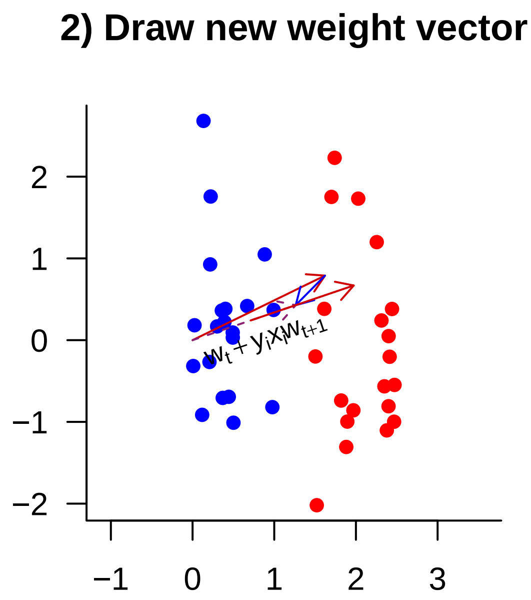


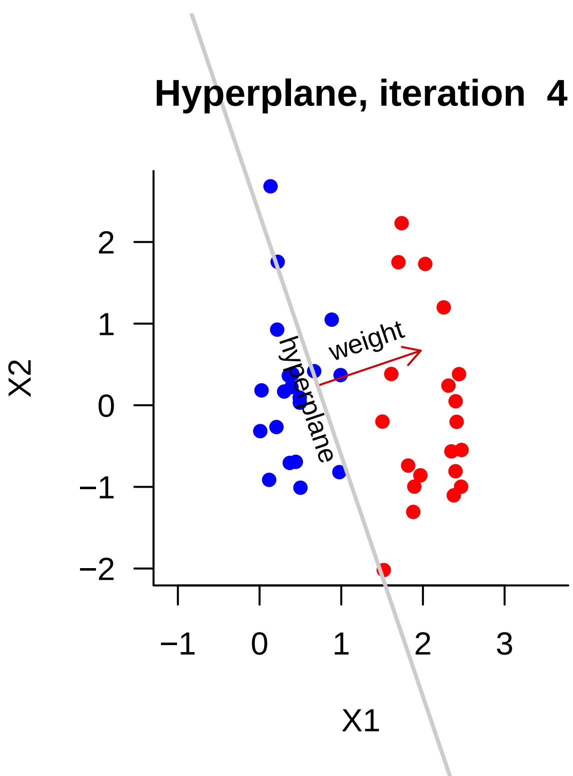
2) Draw new weight vector



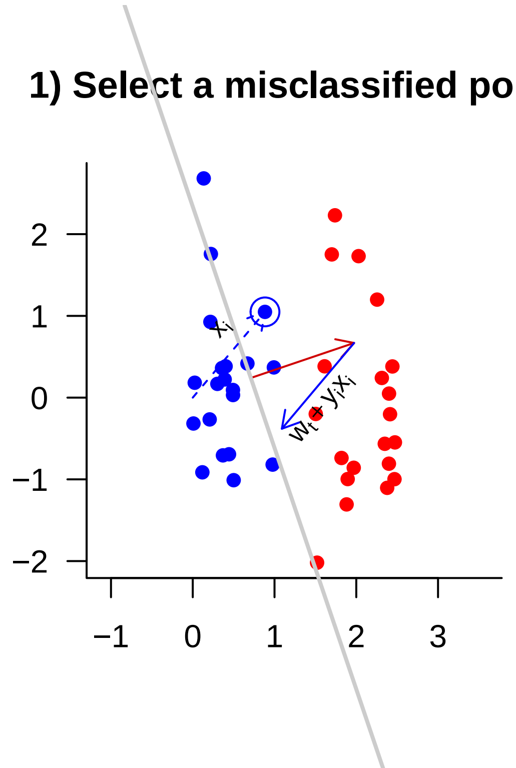

Hyperplane, iteration 3
weight
hyperplane
X2
X1
2
1
0
−1
−2
−1
0
1
2
3


1) Select a misclassified point
xi
wt+yixi
2
1
0
−1
−2
−1
0
1
2
3

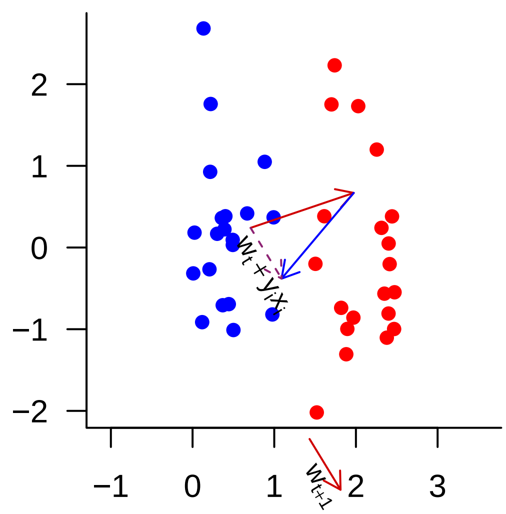

2) Draw new weight vector
w_t + y_i x_i
w_{t+1}


Hyperplane, iteration 4
weight
hyperplane
X1
X2

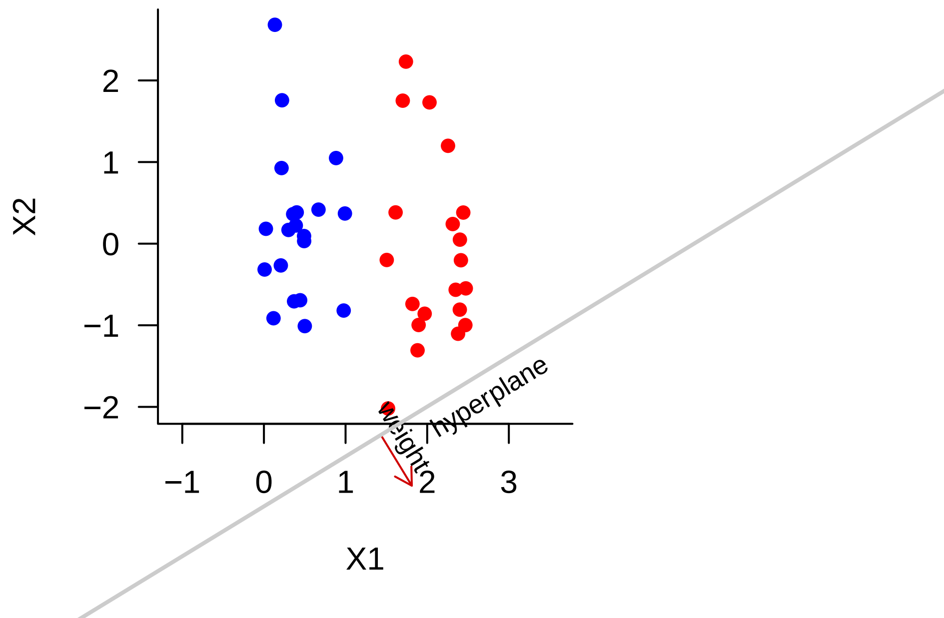
1) Select a misclassified point



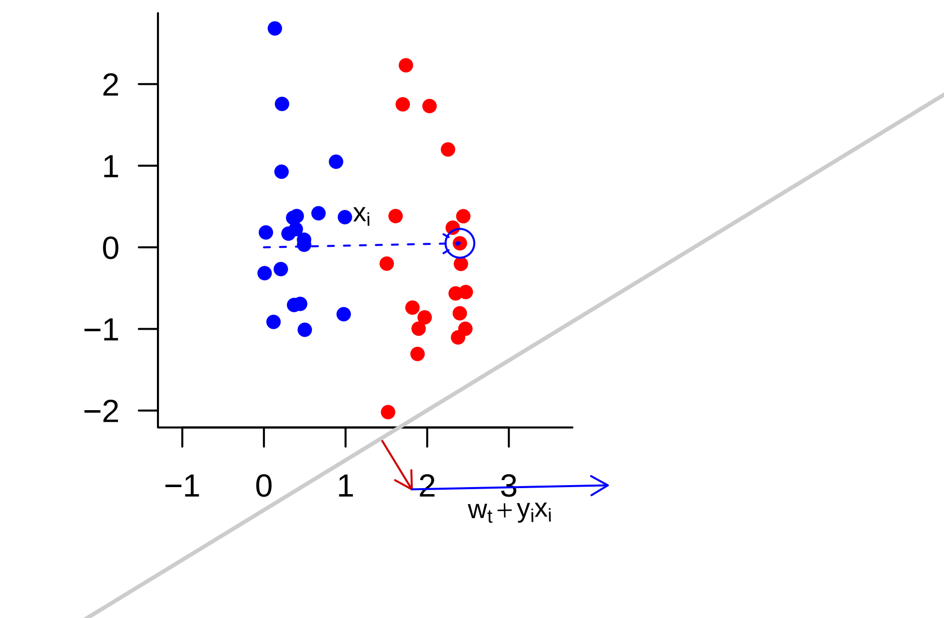
2) Draw new weight vector

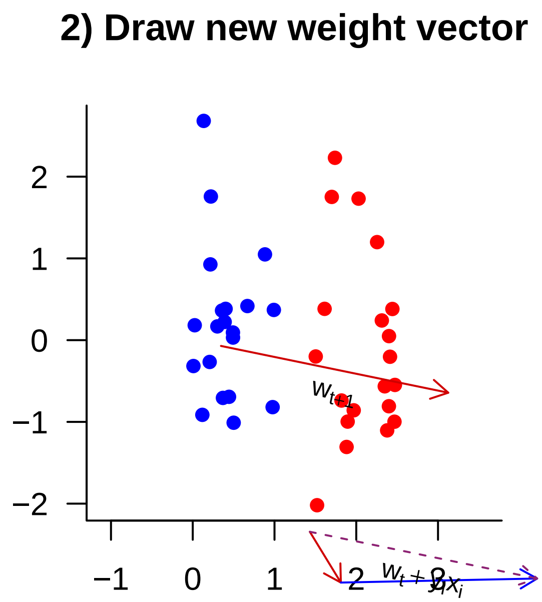


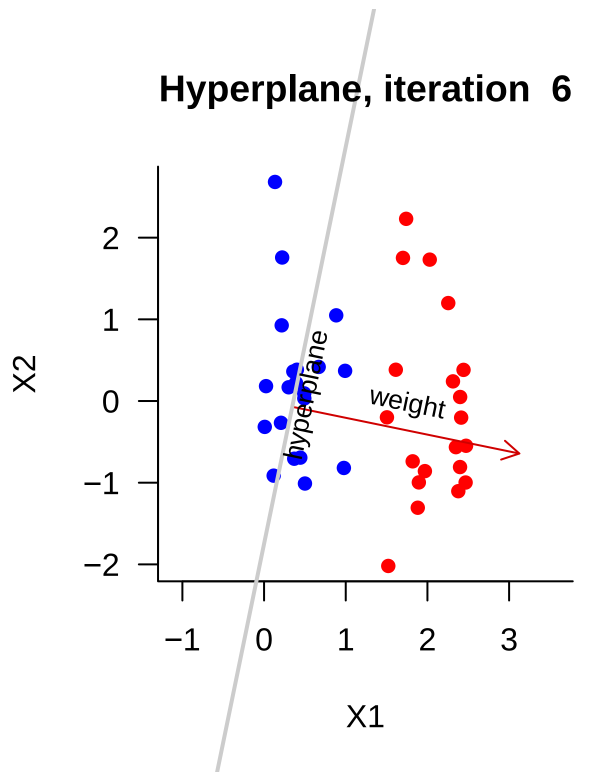
Hyperplane, iteration 5

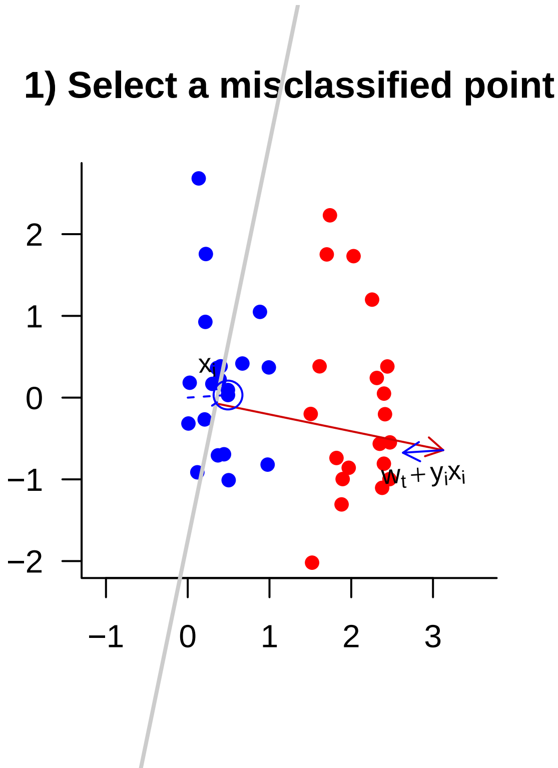


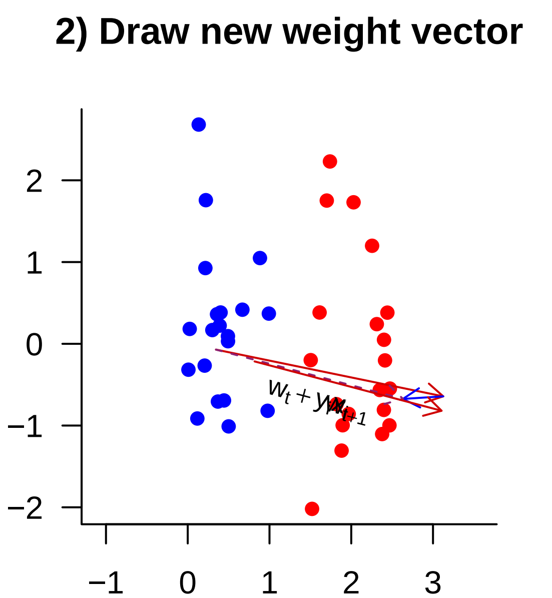
1) Select a misclassified point

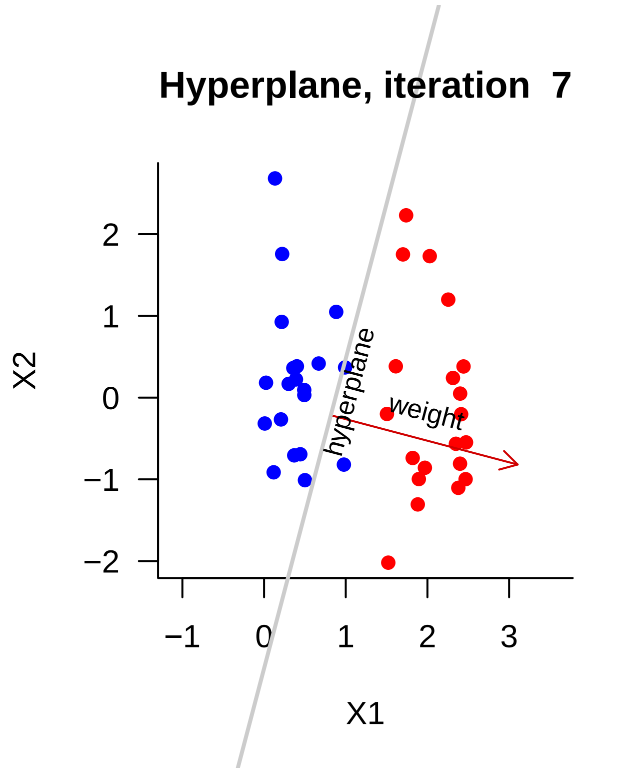


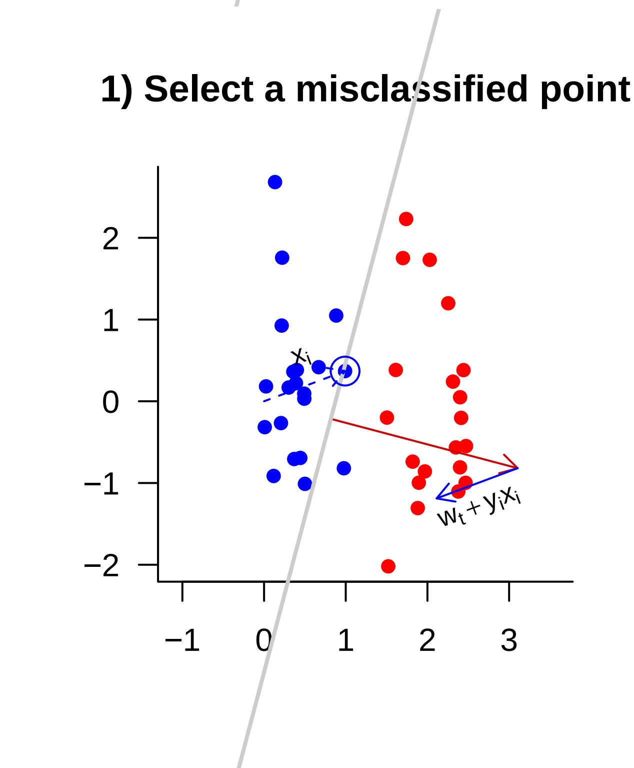

2) Draw new weight vector
w_{t+1}
$w_t + y_i x_i$

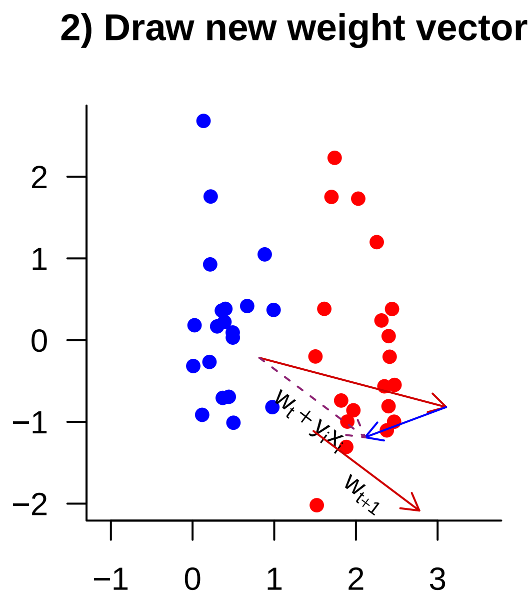

Hyperplane, iteration 6
hyperplane
weight
X1
X2

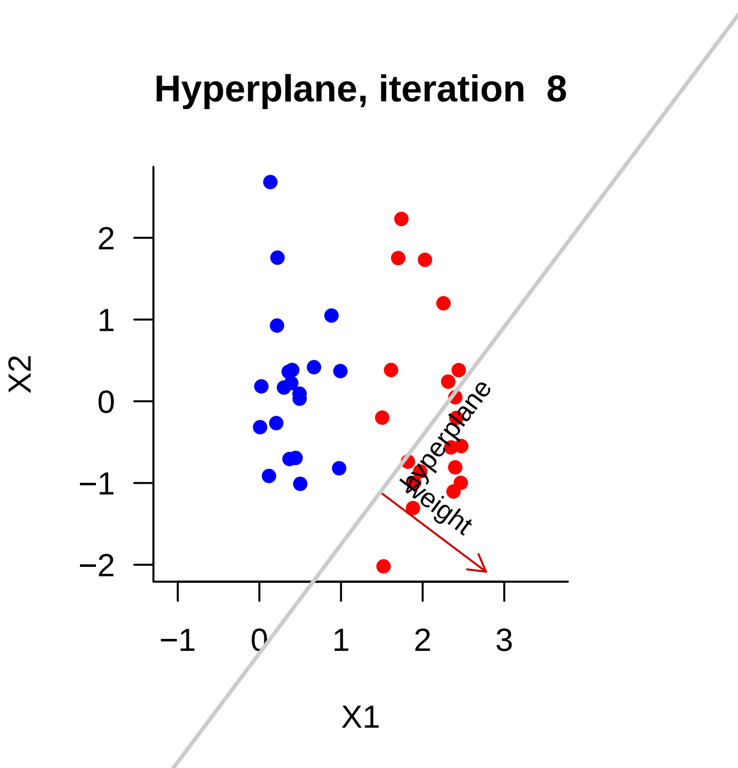

1) Select a misclassified point
2
1
0
−1
−2
−1
0
1
2
3
x_i
$w_t + y_i x_i$

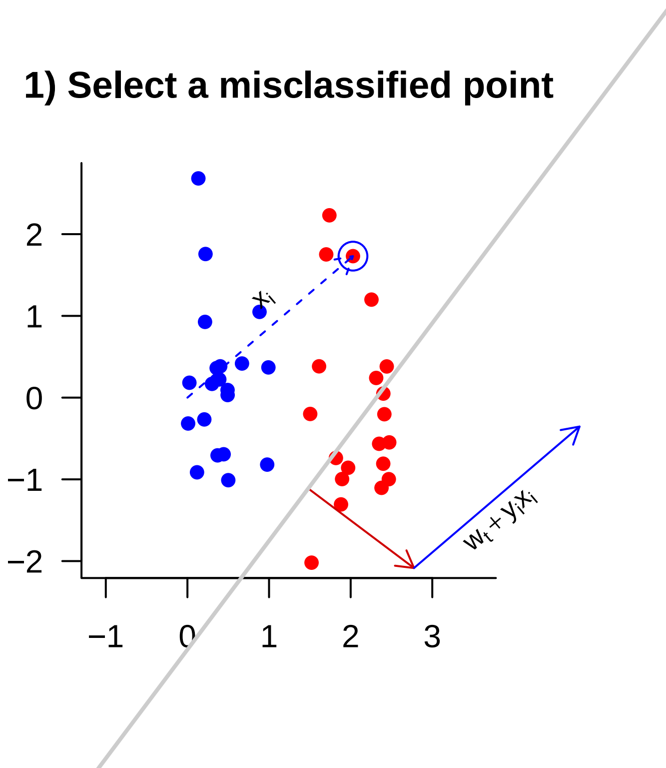

2) Draw new weight vector
2
1
0
−1
−2
−1
0
1
2
3
$w_t + y_i x_i$
w_{t+1}

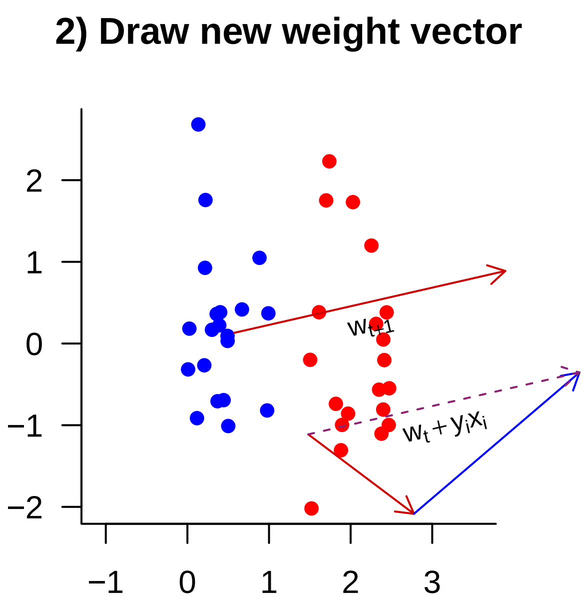

Hyperplane, iteration 7
X2
X1
hyperplane
weight
2
1
0
−1
−2
−1
0
1
2
3

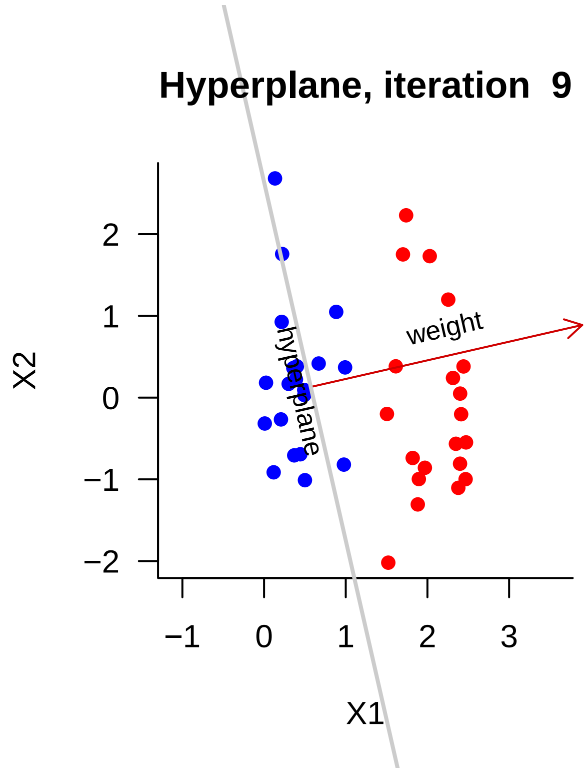

1) Select a misclassified point
x_i
$w_t + y_i x_i$
2
1
0
−1
−2
−1
0
1
2
3

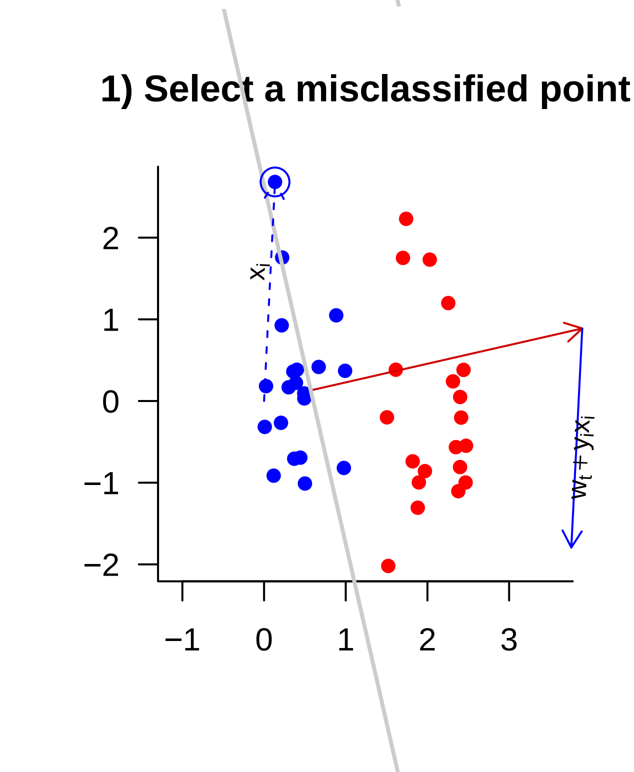

2) Draw new weight vector
2
1
0
−1
−2
−1
0
1
2
3
$w_t + y_i X_i$
w_{t+1}

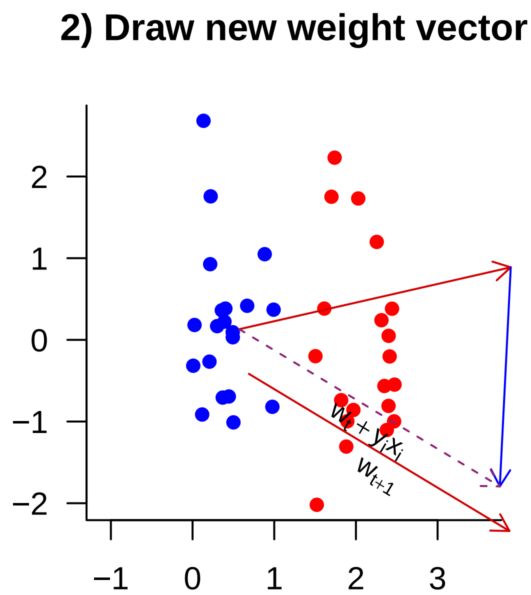

Hyperplane, iteration 8
X2
2
1
0
−1
−2
−1
0
1
2
3
X1
hyperplane
weight

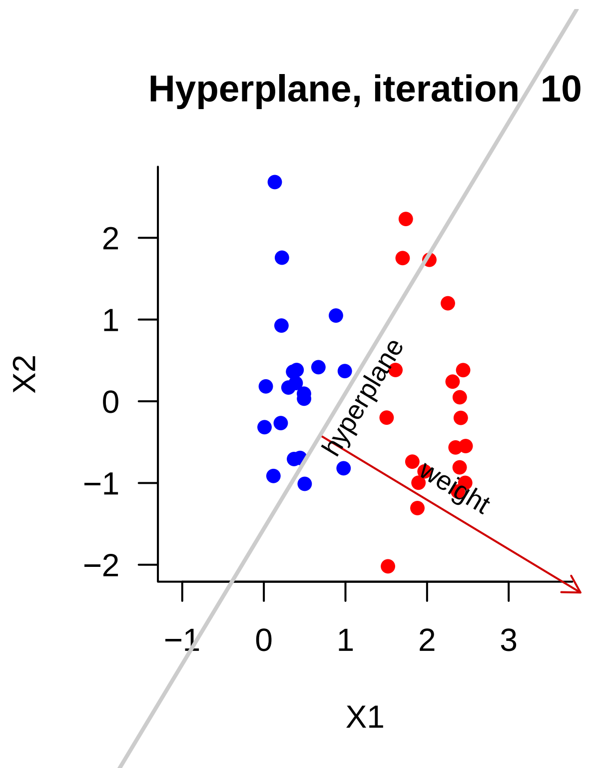

1) Select a misclassified point
x_i
$w_t + y_i x_i$
2
1
0
−1
−2
−1
0
1
2
3


2) Draw new weight vector
w_{t+1}
$w_t + y_i x_i$
2
1
0
−1
−2
−1
0
1
2
3

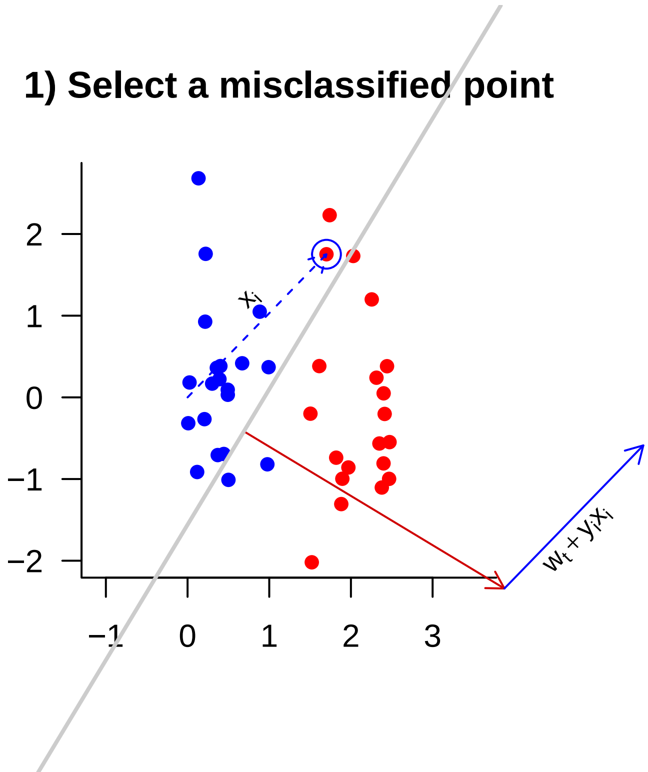

Hyperplane, iteration 9
X2
X1
weight
hyperplane
2
1
0
−1
−2
−1
0
1
2
3


1) Select a misclassified point
x_i
$w_t + y_i x_i$
2
1
0
−1
−2
−1
0
1
2
3

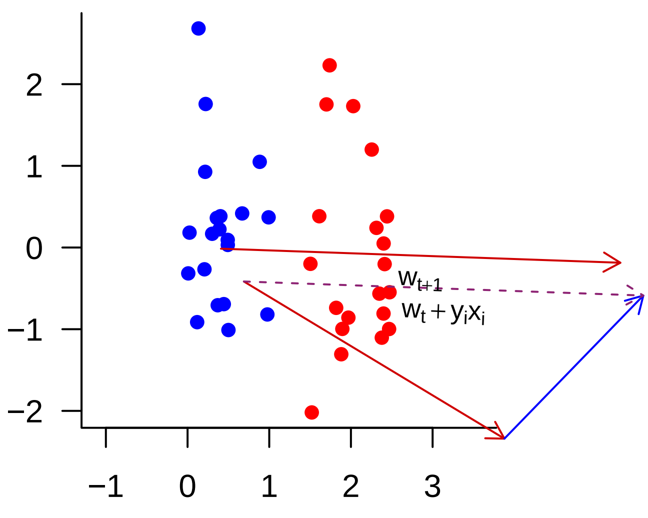

2) Draw new weight vector
$w_t + y_i x_i$
w_{t+1}
2
1
0
−1
−2
−1
0
1
2
3

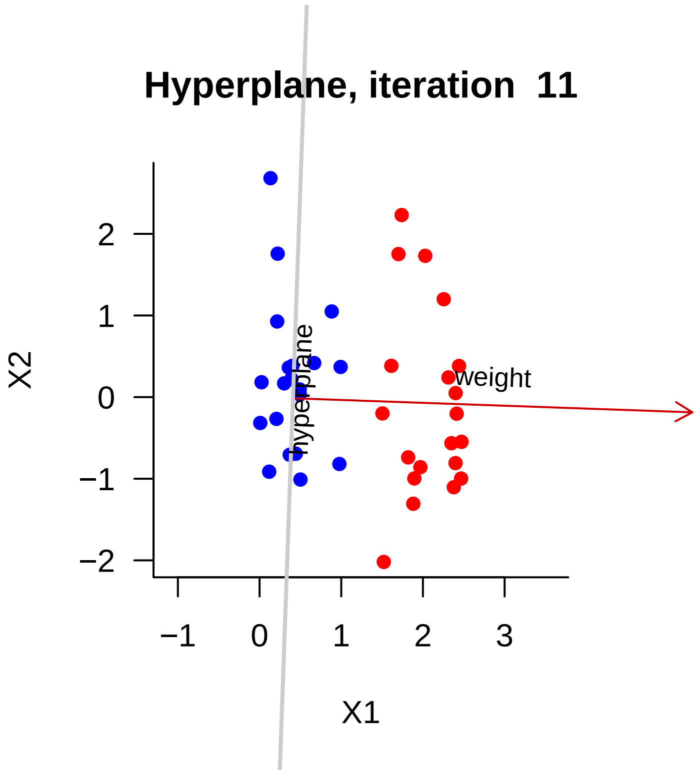

Hyperplane, iteration 10
hyperplane
weight
X2
X1
2
1
0
−1
−2
−1
0
1
2
3

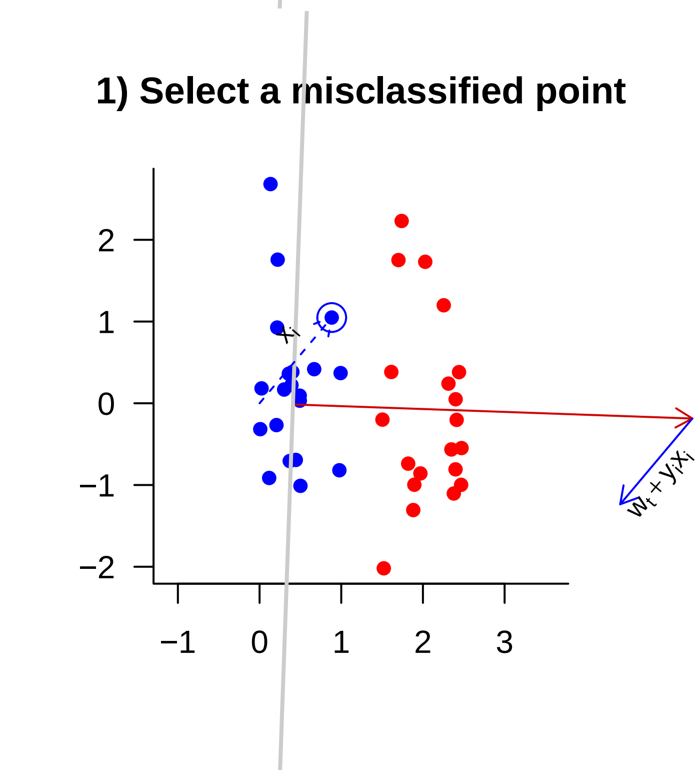
1) Select a misclassified point

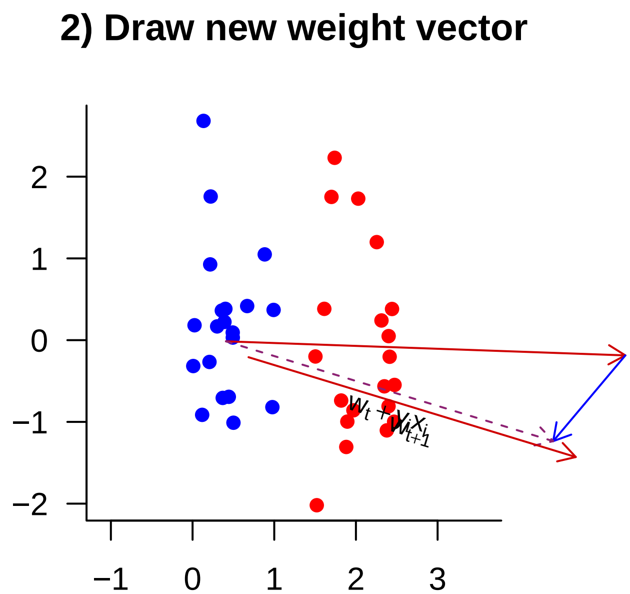


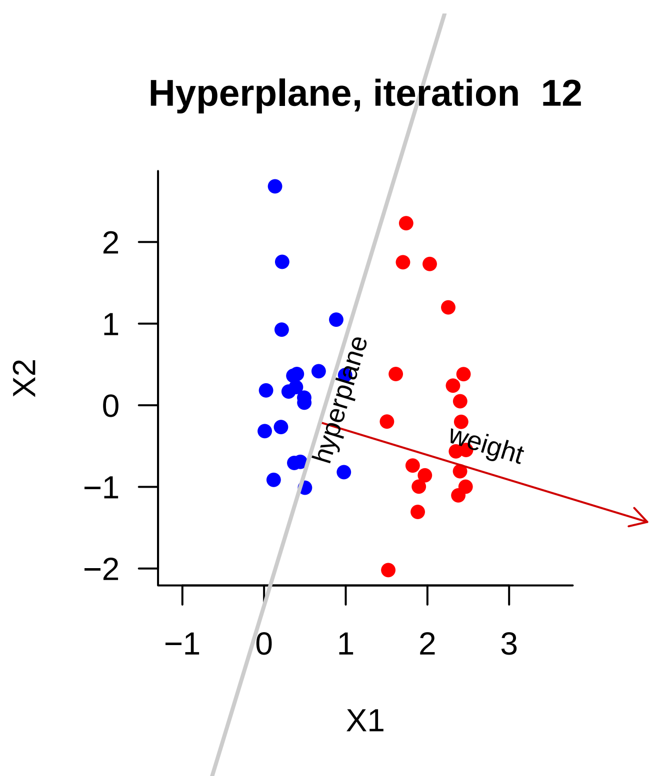
2) Draw new weight vector

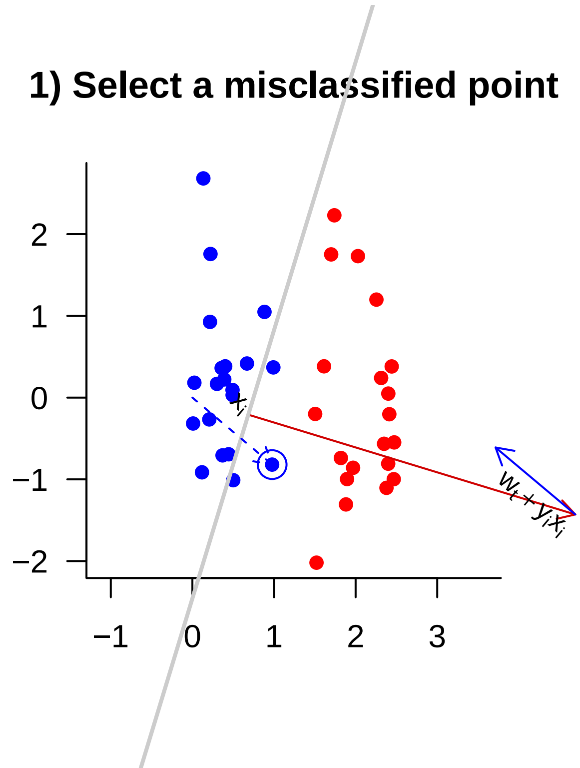


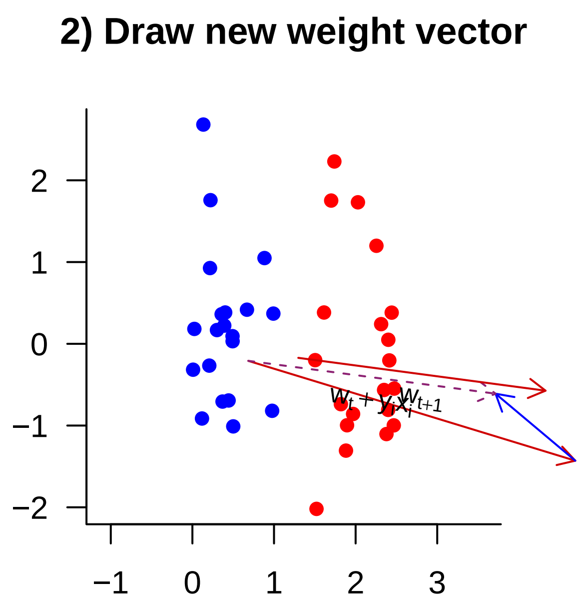

Hyperplane, iteration 11
X2
X1
weight
hyperplane
2
1
0
−1
−2
−1
0
1
2
3

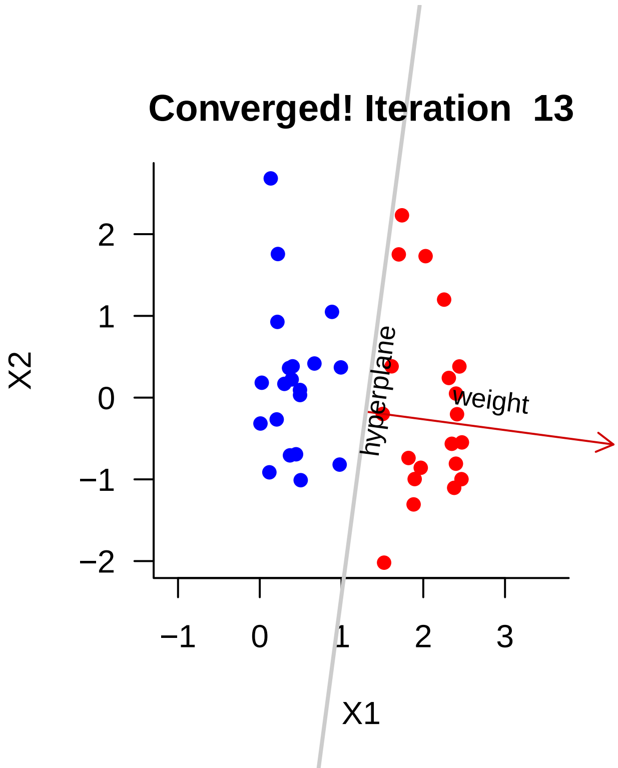

1) Select a misclassified point
x_i
$w_t + y_i x_i$
2
1
0
−1
−2
−1
0
1
2
3


2) Draw new weight vector
$w_t + y_i x_i$
w_{t+1}


Hyperplane, iteration 12
hyperplane
weight
X1
X2


1) Select a misclassified point
2
1
0
−1
−2
−1
0
1
2
3
x_i
w_t + y_i x_i


2) Draw new weight vector
2
1
0
−1
−2
−1
0
1
2
3
w_t + y_i x_i
w_{t+1}




```
p <- par()$usr
```

Summary of 13 iterations:

```
library(tidyverse)
library(scales)
save(weight_matrix, file = "w.rds")

colnames(X) <- c("X0", "X1", "X2")

x <- as.data.frame(X) %>% mutate(Y = Y)

df <- data.frame(slope = apply(weight_matrix, 1, slope),
                 intercept = apply(weight_matrix, 1, intercept))

df <- df %>%
  mutate(iteration = as.factor(1:nrow(df)))

# need to split up x since ggplot2 can only handle one color scale, and I need it for geom_abline
#
group1 <- x[Y == -1,]
group2 <- x[Y == 1,]

purples <- brewer_pal(palette = "Purples")(9)

ggplot(group1, aes(X1, X2)) +
  geom_point(size = 2, color = "blue") +
  geom_point(data = group2, size = 2, color = "red") +
```

```r
  geom_abline(data = df, aes(slope = slope, intercept = intercept,
                  color = iteration)) +
  scale_color_manual(values = gradient_n_pal(purples)(seq(0, 1, length.out = nrow(df)))) +
  geom_abline(data = df[nrow(df),], aes(slope = slope,
                  intercept = intercept), size = 2,
              color = "#53278F") +
  theme_classic() + xlim(p[1], p[2]) + ylim(p[3], p[4])
```