STAT 760 - Machine/Statistical Learning Project

Fisher Discriminant Algorithm

Amos Natido

February, 2019

```
pacman::p_load(tidyverse, xfun, kableExtra)

digits <- numbers_to_words(0:9)

train <- read.table("data/zip.train.gz") %>% as.matrix
```

Defined functions

```
# Compute the mean of each variable in a data set
mu_mat <- function(x) {
                      n <- nrow(x)
                      1/n * t(x) %*% matrix(1, n, 1)
}

# Compute the variance-covariance matrix of a data set
cov_mat <- function(x) {
  n <- nrow(x)
  x_bar <- mu_mat(x)
  1/n * t(x) %*% (diag(1,n) - 1/n*matrix(1,n,1) %*% matrix(1,1,n)) %*% x
}

# Rotate 2D data by some angle (radians)
rotate <- function(x, theta) {
  m <- matrix(c(cos(theta), -sin(theta), sin(theta), cos(theta)),
              nrow = 2, byrow = T)
  t(m %*% t(x))
}

# Projection of X on w
proj <- function(X, w) {
  as.numeric(t(w) %*% w)^-1 * (X %*% w) %*% t(w)
}
```

Task 1

We compute the mean and covariance matrix for each digit, 0 through 9 using the zip code data set from *The Elements of Statistical Learning.* The class digits are split up into individual matrices as follows

```r
digit_mat <- vector("list", 10)  # Initialize a list to store matrix by digit

for (k in 0:9) {
      digit_mat[[k+1]] <- train[train[,1] == k, -1]
}

names(digit_mat) <- digits
```

Mean features of each digit

```r
mean_mat <- matrix(NA_real_, 256, 10,
                   dimnames = list(
                     str_pad(as.character(1:256), 3, "left", "0"),
                     digits))
```

```r
for (k in digits) {
  mean_mat[,k] <- mu_mat(digit_mat[[k]])
}
```

First and last 5 features of the digits

	Handwritten Digit									
	zero	one	two	three	four	five	six	seven	eight	nine
001	-1.00	-1	-0.99	-1.00	-1.00	-1.00	-1.00	-0.97	-1.00	-1.00
002	-1.00	-1	-0.96	-0.98	-1.00	-1.00	-1.00	-0.88	-0.98	-1.00
003	-0.98	-1	-0.90	-0.91	-0.99	-0.97	-1.00	-0.76	-0.94	-1.00
004	-0.94	-1	-0.80	-0.73	-0.96	-0.93	-1.00	-0.60	-0.83	-0.98
005	-0.83	-1	-0.60	-0.42	-0.88	-0.88	-0.99	-0.42	-0.60	-0.94
252	-0.83	-1	-0.80	-0.61	-0.86	-0.51	-0.86	-1.00	-0.86	-0.96
253	-0.97	-1	-0.72	-0.82	-0.94	-0.78	-0.97	-1.00	-0.97	-0.99
254	-1.00	-1	-0.72	-0.93	-0.99	-0.92	-1.00	-1.00	-1.00	-1.00
255	-1.00	-1	-0.83	-0.99	-1.00	-0.98	-1.00	-1.00	-1.00	-1.00
256	-1.00	-1	-0.95	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

covariance matrix of features for each digit

Since the dimensions of each covariance matrix will be the same, I initialized a $[10 \times 256 \times 256]$ array, where each layer of the array will store the covariance matrix for each digit.

```r
cov_arr <- array(NA_real_, dim = c(10, 256, 256))

dimnames(cov_arr) <- list(digits, NULL, NULL)
```

```r
for (k in digits) {
  cov_arr[k,,] <- cov_mat(digit_mat[[k]])
}
```

The determinant for each covariance matrix

```r
cov_det <- vector("numeric", 10)
names(cov_det) <- digits

for (k in digits) {
  cov_det[k] <- det(cov_arr[k,,])
}
```

Since the determinant for each digit's covariance matrix is zero, we would need to use a method such as

Table 1: Determinant for each digit covariance matrix

zero	one	two	three	four	five	six	seven	eight	nine
0	0	0	0	0	0	0	0	0	0

adding in random noise or tolerance in order to take the inverse.

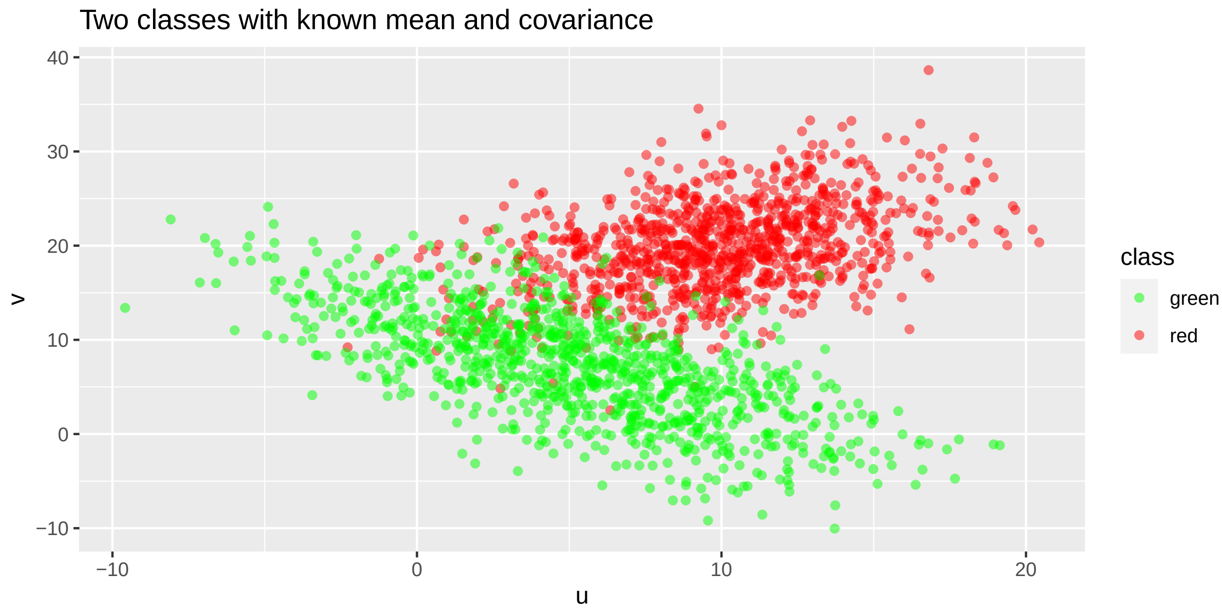
Task 2

Here, we implemented the Fisher discriminant criterion via simulation. Two multivariate classes were simulated from a gaussian distribution, and then used a rotation matrix to transform the data to form an elliptical shape

Data Simulation

```
# Number of observations per class
n <- 1000

# Red Class
red.mu <- c(10, 20)
red.sd <- c(3, 5)

red <- matrix(c(rnorm(n, 0, red.sd[1]),
                rnorm(n, 0, red.sd[2])),
              nrow = n, byrow = F) %>%
             rotate(-pi/6)

red[,1] <- red[,1] + red.mu[1]
red[,2] <- red[,2] + red.mu[2]

# mean and covariance matrix
red.cov <- cov_mat(red)
red.mu <- mu_mat(red)

red.tbl <- tibble(u = red[,1],
                  v = red[,2],
                  class = "red")

# Green Class
green.mu <- c(5, 7)
green.sd <- c(3, 7)

green <- matrix(c(rnorm(n, 0, green.sd[1]),
                  rnorm(n, 0, green.sd[2])),
                nrow = n, byrow = F) %>%
                       rotate(pi/5)

green[,1] <- green[,1] + green.mu[1]
green[,2] <- green[,2] + green.mu[2]

#mean and covariance matrix
green.cov <- cov_mat(green)
green.mu <- mu_mat(green)
```

```r
green.tbl <- tibble(u = green[,1],
                    v = green[,2],
                    class = "green")
```



To calculate the Fisher's linear discriminant, we can use the formula

$$\vec{w} \propto (\Sigma_0 + \Sigma_1)^{-1}(\vec{\mu_1} - \vec{\mu_0})$$

which gives us the direction of the best vector to project our data onto.

```r
cov_sum <- red.cov + green.cov
w <- solve(cov_sum) %*% (red.mu - green.mu)
w <- w / as.numeric(sqrt(t(w) %*% w))  # Normalize the vector
```

```r
p1 <- bind_rows(red.tbl, green.tbl) %>%
                        mutate_at(vars(class), as.factor) %>%
                        ggplot(aes(x = u, y = v, color = class)) +
                        geom_point(alpha = 0.3) +
                        geom_abline(slope = -w[1] / w[2], intercept = c.prime,
                                    linetype = "dashed") +
                        geom_abline(slope = w[2] / w[1], intercept = 0) +
                        scale_color_manual(values = c("green", "red")) +
                        coord_fixed()
```

```r
p2 <- bind_rows(tibble(proj = proj_red_w[,1], class = "red"),
          tibble(proj = proj_green_w[,1], class = "green")) %>%
                mutate_at(vars(class), as.factor) %>%
                ggplot(aes(x = proj, y = ..density.., fill = class, color = class)) +
                geom_histogram(alpha = 0.3, position = "identity", bins = 35) +
                geom_vline(xintercept = c, linetype = "dashed") +
                scale_fill_manual(values = c("green", "red")) +
                scale_color_manual(values = c("green", "red"))
```

The optimal separating hyperplane was found by estimating the normal distributions of the projected classes, and using a root finding numerical(Bisection) to find the point where the two distributions are equal.

The linear regression model that we can use to do classification is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

where y is either 0 (green class) or 1 (red class). If we let $y = 1/2$ be our threshold for classification, then our best prediction for a new observation, x, is

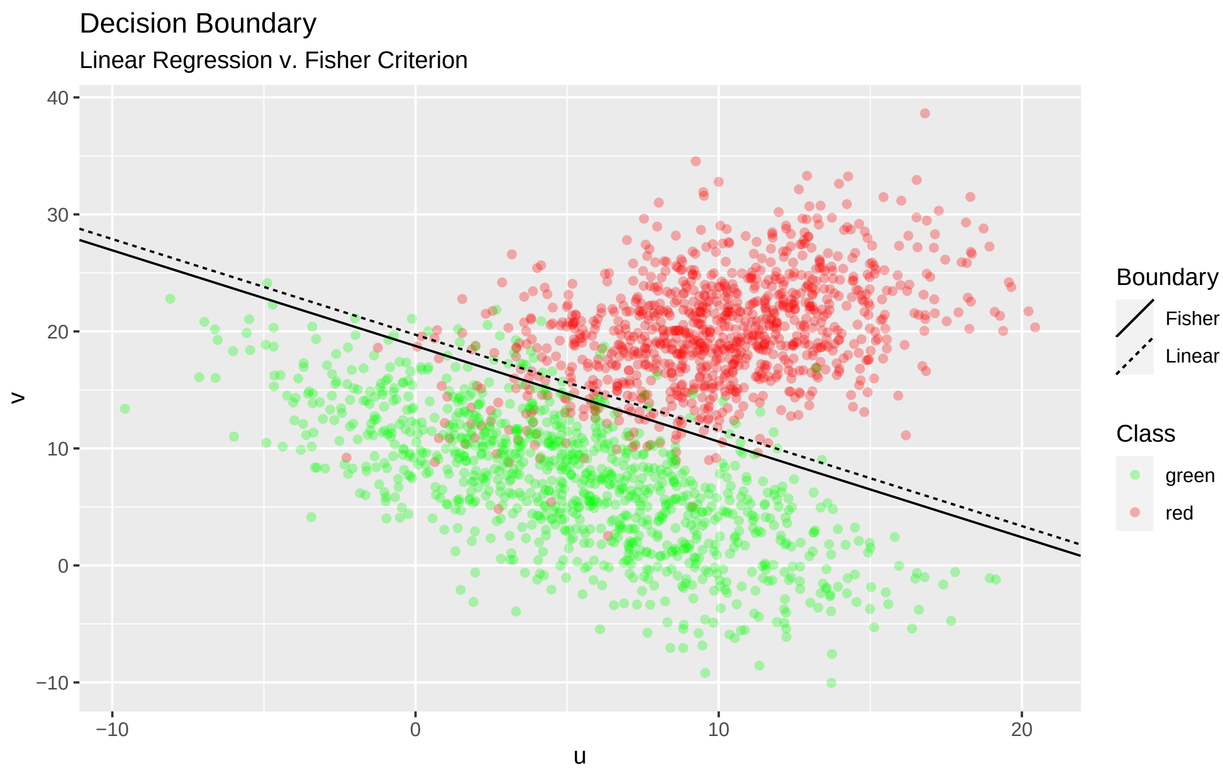
$$\begin{cases} x \in k_0 & \text{if } \beta_0 + \beta_1 x_1 + \beta_2 x_2 \leq \frac{1}{2} \\ x \in k_1 & \text{if } \beta_0 + \beta_1 x_1 + \beta_2 x_2 > \frac{1}{2} \end{cases}$$

Comparing Linear regression and Fisher criterion Decision boundaries

```
lin.tbl <- bind_rows(tibble(X1 = green[,1],
                            X2 = green[,2],
                            Y  = 0),
                     tibble(X1 = red[,1],
                            X2 = red[,2],
                            Y  = 1))

model_fit <- glm(Y ~ X1 + X2, data = lin.tbl, family = "binomial")

B <- model_fit$coefficients

logi <- data.frame(intercept = -B[1]/B[3],
```

```r
                   slope     = -B[2]/B[3],
                   Boundary      = "Logistic")

fish <- data.frame(intercept = c.prime,
                   slope     = -w[1] / w[2],
                   Boundary      = "Fisher")

lin_fit <- function(X, y) {
  X <- cbind(matrix(1, nrow(X), 1), X)

  solve((t(X) %*% X)) %*% (t(X) %*% y)
}

X.lin <- select(lin.tbl, X1, X2) %>% as.matrix()
y.lin <- lin.tbl$Y %>% as.matrix()
z <- lin_fit(X.lin, y.lin)
linear <- data.frame(intercept = (1 - 2*z[1]) / (2*z[3]),
                     slope     = -z[2] / z[3],
                     Boundary  = "Linear")
```

```r
bind_rows(red.tbl, green.tbl) %>%
              mutate_at(vars(class), as.factor) %>%
              ggplot(aes(x = u, y = v, color = class)) +
              geom_point(alpha = 0.3) +
              geom_abline(data = fish,
                          aes(slope = slope, intercept = intercept, linetype = Boundary)) +
              geom_abline(data = linear,
                          aes(slope = slope, intercept = intercept, linetype = Boundary)) +
              scale_color_manual(name = "Class", values = c("green", "red")) +
              labs(title = "Decision Boundary",
                   subtitle = "Linear Regression v. Fisher Criterion")
```


Decision Boundary
Linear Regression v. Fisher Criterion
40
30
20
10
0
−10
−10
0
10
20
u
v
Boundary
Fisher
Linear
Class
green
red