Tarea 5

Dalia Camacho

September 16, 2018

Ejercicio 1

En la Tarea 4, construye curvas ROC para cada uno de los tres modelos (una sola variable, todas las variables, y todas las variables más variables de ruido).

```
# Volver a correr tarea 4
source("Tarea4/tarea_4_codigo.R")
suppressMessages(library(tidyverse))
datos_titanic <- read_csv("Tarea4/tarea_4_datos/train.csv")
```

```
## Parsed with column specification:
## cols(
##   PassengerId = col_integer(),
##   Survived = col_integer(),
##   Pclass = col_integer(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_integer(),
##   Parch = col_integer(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
# Modificar bases
datos_titanic <- datos_titanic %>% select(Survived, Pclass, Age, Sex, Embarked) %>%
  filter(!is.na(Age), !is.na(Embarked))
summary(datos_titanic)
```

```
##     Survived          Pclass          Age            Sex
##  Min.   :0.0000   Min.   :1.00   Min.   : 0.42   Length:712
##  1st Qu.:0.0000   1st Qu.:1.00   1st Qu.:20.00   Class :character
##  Median :0.0000   Median :2.00   Median :28.00   Mode  :character
##  Mean   :0.4045   Mean   :2.24   Mean   :29.64
##  3rd Qu.:1.0000   3rd Qu.:3.00   3rd Qu.:38.00
##  Max.   :1.0000   Max.   :3.00   Max.   :80.00
##    Embarked
##  Length:712
##  Class :character
##  Mode  :character
##
##
##
```

```
datos <- datos_titanic %>%
  mutate(female = as.numeric(Sex == "female"),
         southampton = as.numeric(Embarked == "S"),
```

```
         cherbourg = as.numeric(Embarked == "C")) %>%
  select(-Embarked, -Sex)


datos$age_n <- datos$Age / 60
datos$pclass_n <-(datos$Pclass - 1) / 3
datos_trans <- datos %>% select(Survived, pclass_n, age_n, female, southampton, cherbourg)

set.seed(2850)
datos_trans <- datos_trans %>%
  mutate(u = runif(nrow(datos_trans)))
entrena <- datos_trans %>% filter(u <= 0.7) %>% select(-u)
prueba <- datos_trans %>% filter(u > 0.7) %>% select(-u)

x_ent <- as.matrix(entrena %>% select(-Survived))
x_pr <- as.matrix(prueba %>% select(-Survived))
y_ent <- entrena$Survived
y_pr <- prueba$Survived

# Estimación una sola variable
x_ent_1 <- x_ent[ , "cherbourg", drop = FALSE] # drop=false es para no convertir en vector
devianza_ent <- devianza_calc(x_ent_1, y_ent)
grad_ent <- grad_calc(x_ent_1, y_ent)
## termina esta línea para descenso en gradiente
```

Generar probabilidades para los tres modelos

```
# Una sola variable (Cherbourg)
n     <- 10000
z     <- descenso(n = n, z_0 = rep(0, 2), eta = 0.0001, h_deriv = grad_ent)
beta  <- z[n,]
prueba$Prob_mod_cher <- h(cbind(rep(1, nrow(prueba)),
                                prueba$cherbourg)%*%beta)
```

Para todas las variables

```
devianza_ent <- devianza_calc(x_ent, y_ent)
grad_ent     <- grad_calc(x_ent, y_ent)
z            <- descenso(n = n, z_0 = rep(0, 6),
                         eta = 0.0001, h_deriv = grad_ent)
beta2       <- z[n,]
prueba$Prob_mod_all <- h(cbind(rep(1, nrow(prueba)),
                               x_pr)%*%beta2)
```

Para variables con ruido

```
p_ruido <- 50 # agregamos 50 variables sin información
n_ent   <- nrow(x_ent)
n_pr    <- nrow(x_pr)
mat_ent <- matrix(runif(n_ent * p_ruido), n_ent, p_ruido)
mat_pr  <- matrix(runif(n_pr * p_ruido), n_pr, p_ruido)

devianza_ent <- devianza_calc(cbind(x_ent, mat_ent), y_ent)
grad_ent <- grad_calc(cbind(x_ent, mat_ent), y_ent)
## termina esta línea
z     <- descenso(n = n, z_0 = rep(0,56), eta = 0.0001, h_deriv = grad_ent)
```

```
beta3 <- z[n,]
prueba$Prob_mod_ruido <- h(cbind(rep(1, nrow(prueba)),x_pr, mat_pr)%*%beta3)
```

Generar curvas ROC

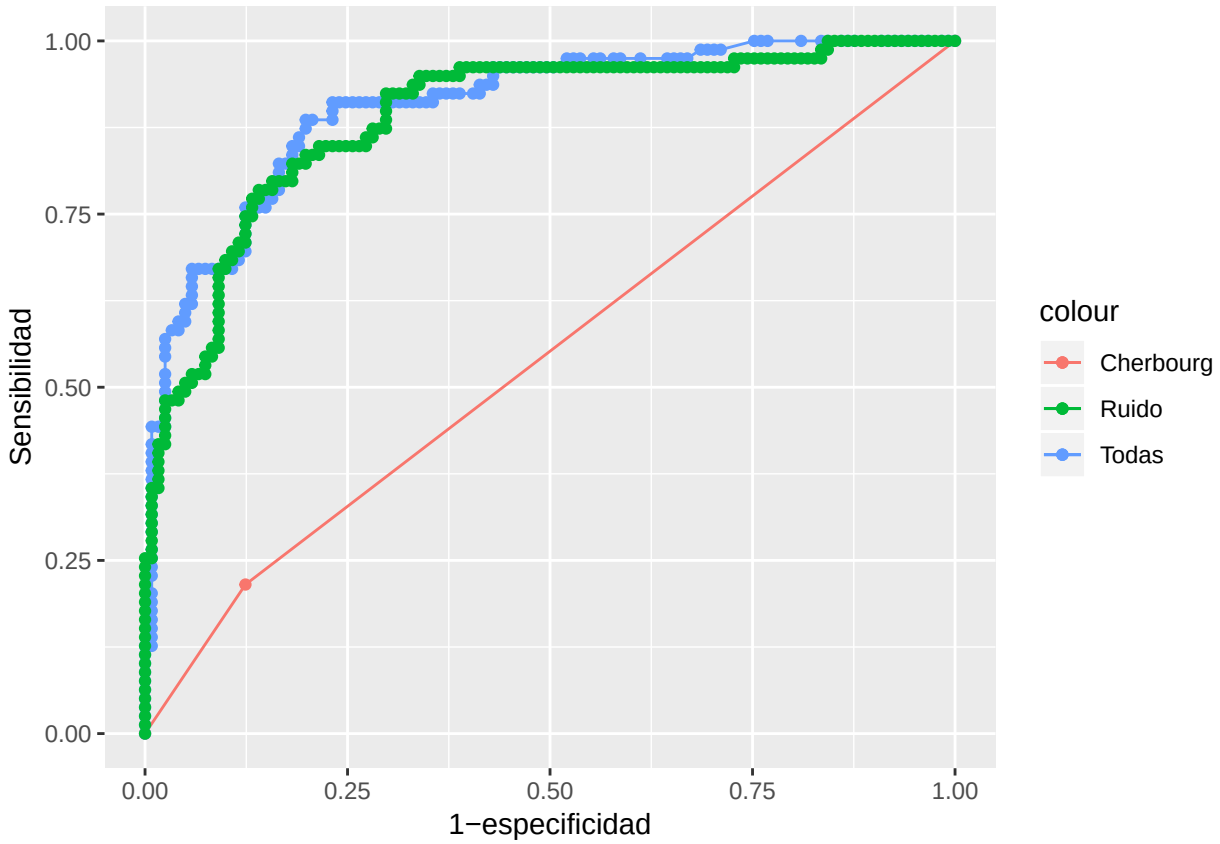
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# Cherbourg
pred_rocr1 <- prediction(prueba$Prob_mod_cher, prueba$Survived)
perf1      <- performance(pred_rocr1, measure = "sens", x.measure = "fpr")
graf_roc_1 <- data_frame(tfp = perf1@x.values[[1]], sens = perf1@y.values[[1]],
                         d = perf1@alpha.values[[1]])

g1 <- ggplot(graf_roc_1, aes(x = tfp, y = sens, colour=d)) + geom_point() + geom_line()+
  xlab('1-especificidad') + ylab('Sensibilidad')

# Todas las variables
pred_rocr2 <- prediction(prueba$Prob_mod_all, prueba$Survived)
perf2      <- performance(pred_rocr2, measure = "sens", x.measure = "fpr")
graf_roc_2 <- data_frame(tfp = perf2@x.values[[1]], sens = perf2@y.values[[1]],
                         d = perf2@alpha.values[[1]])

g2 <- ggplot(graf_roc_2, aes(x = tfp, y = sens, colour=d)) + geom_point() + geom_line()+
  xlab('1-especificidad') + ylab('Sensibilidad')

# Todas las variables más ruido
pred_rocr3 <- prediction(prueba$Prob_mod_ruido, prueba$Survived)
perf3      <- performance(pred_rocr3, measure = "sens", x.measure = "fpr")
graf_roc_3 <- data_frame(tfp = perf3@x.values[[1]], sens = perf3@y.values[[1]],
                         d = perf3@alpha.values[[1]])

ggplot()+
  geom_point(aes(graf_roc_1$tfp, graf_roc_1$sens, colour="Cherbourg"))+
  geom_line(aes(graf_roc_1$tfp, graf_roc_1$sens, colour="Cherbourg"))+

  geom_point(aes(graf_roc_2$tfp, graf_roc_2$sens, colour="Todas"))+
  geom_line(aes(graf_roc_2$tfp, graf_roc_2$sens, colour="Todas"))+

  geom_point(aes(graf_roc_3$tfp, graf_roc_3$sens, colour="Ruido"))+
  geom_line(aes(graf_roc_3$tfp, graf_roc_3$sens, colour="Ruido"))+

  xlab('1-especificidad') + ylab('Sensibilidad')
```



¿Cuál tiene mejor desempeño?

La curva con todas las variables sin ruido parece ser la que tiene mejor desempeño en la mayoría de los casos

Calcula el AUC para cada una de las tres curvas.

```
#Cherbourg
auc_1 <- performance(pred_rocr1, measure = 'auc')@y.values
auc_1[[1]]
```

```
## [1] 0.5456115
```

```
#Todas
auc_2 <- performance(pred_rocr2, measure = 'auc')@y.values
auc_2[[1]]
```

```
## [1] 0.9055341
```

```
#Todas con ruido
auc_3 <- performance(pred_rocr3, measure = 'auc')@y.values
auc_3[[1]]
```

```
## [1] 0.8897374
```

El modelo con mejor desempeño en términos de AUC es el modelo con todas las variables sin ruido.

Ejercicio 2

Para el ejemplo de regresión logística multinomial que vimos en clase (clasificación de dígitos 0-9), construye la gráfica de coeficientes (sección 4.3.3) para:

a)

+El modelo que vimos en clase donde no habían convergido los coeficientes

```
digitos_entrena <- read_csv('digitos/zip-train.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
digitos_prueba  <- read_csv('digitos/zip-test.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
names(digitos_entrena)[1] <- 'digito'
names(digitos_entrena)[2:257] <- paste0('pixel_', 1:256)
names(digitos_prueba)[1] <- 'digito'
names(digitos_prueba)[2:257] <- paste0('pixel_', 1:256)
```

```
library(nnet)
mod_mult <- multinom(digito ~ ., data = digitos_entrena, MaxNWt=100000, maxit = 20)
```

```
## # weights:  2580 (2313 variable)
## initial  value 16788.147913
## iter  10 value 2598.959017
## iter  20 value 1494.978090
## final  value 1494.978090
## stopped after 20 iterations
```

```
coefs <- coef(mod_mult)
coefs_reng <- coefs[1, , drop =FALSE]
coefs <- rbind(coefs_reng, coefs)
coefs[1 , ] <- 0
```

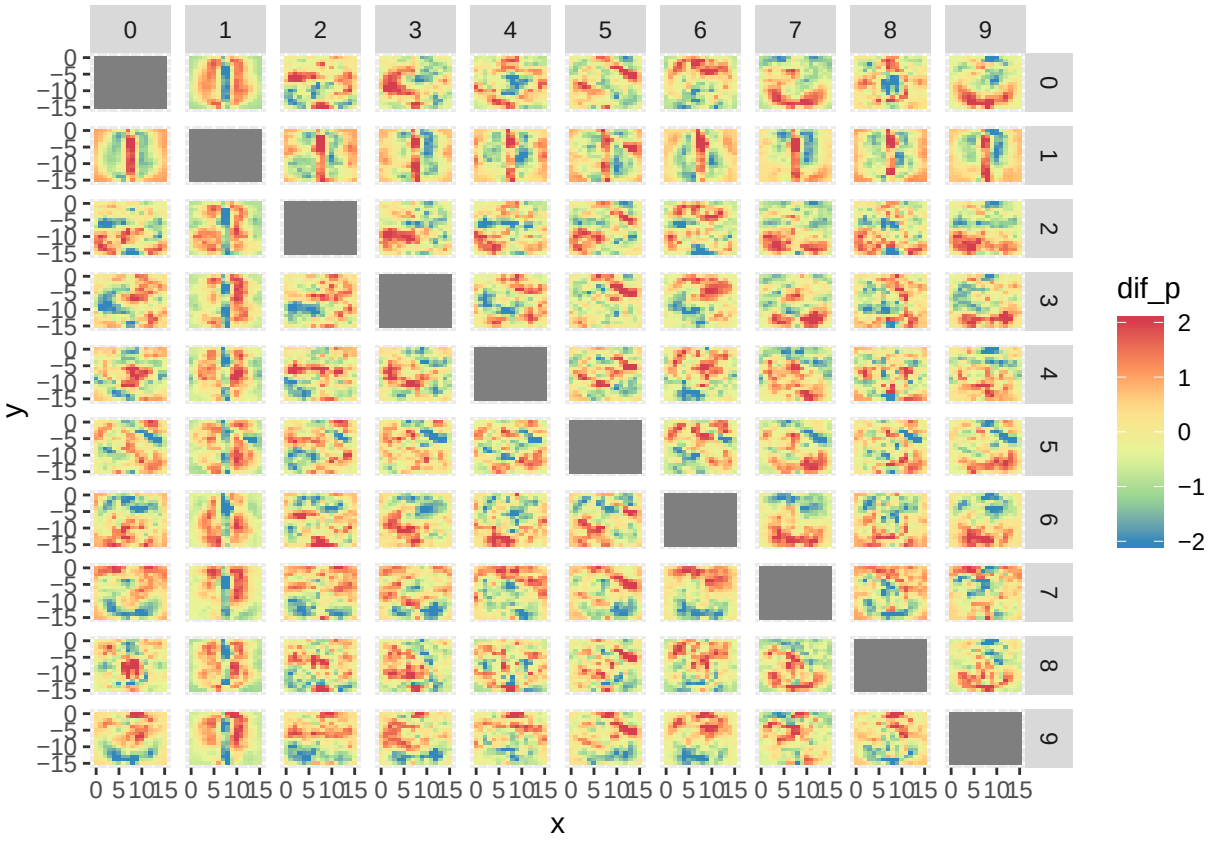
```
beta_df <- coefs[,-1] %>% as.data.frame %>%
  mutate(digito = 0:(nrow(coefs)-1)) %>%
  gather(pixel, valor, contains('pixel')) %>%
  separate(pixel, into = c('str','pixel_no'), sep='_') %>%
  mutate(x = (as.integer(pixel_no)-1) %% 16, y = -((as.integer(pixel_no)-1) %/% 16))
```

```
tab_coef <- beta_df %>% select(digito, x, y, valor)
tab_coef_1 <- tab_coef
names(tab_coef_1) <- c('digito_1','x','y','valor_1')
tab_cruzada <- full_join(tab_coef_1, tab_coef) %>% mutate(dif = valor_1 - valor)
```

```
## Joining, by = c("x", "y")
```

```
tab_cruzada <- tab_cruzada %>% group_by(digito, digito_1) %>%
  mutate(dif_s = (dif - mean(dif))/sd(dif)) %>%
  mutate(dif_p = pmin(pmax(dif_s, -2), 2))
```

```
ggplot(tab_cruzada, aes(x=x, y=y)) + geom_tile(aes(fill = dif_p)) +
  facet_grid(digito_1~digito) + scale_fill_distiller(palette = "Spectral")
```



En el modelo que llega a converger se ve el sobre ajuste, ya que no es distinguible a qué dígito corresponden los pixeles en la muestra de prueba.

b)

+El modelo después de correr hasta convergencia (usa la función multinom) Compara las gráficas. ¿Cuál es más interpretable? ¿Puedes ver el sobreajuste del segundo modelo en estas gráficas?

```
library(nnet)
mod_mult2 <- multinom(digito ~ ., data = digitos_entrena, MaxNWt=100000, maxit = 500)
```

```
## # weights:  2580 (2313 variable)
## initial  value 16788.147913
## iter  10 value 2598.959017
## iter  20 value 1494.978090
## iter  30 value 903.291402
## iter  40 value 443.785686
## iter  50 value 260.626756
## iter  60 value 190.835491
## iter  70 value 160.773160
## iter  80 value 114.048146
```

```
## iter  90 value 88.746976
## iter 100 value 76.302570
## iter 110 value 63.400188
## iter 120 value 54.375215
## iter 130 value 46.291174
## iter 140 value 38.303470
## iter 150 value 28.822810
## iter 160 value 17.888648
## iter 170 value 9.531256
## iter 180 value 2.985614
## iter 190 value 0.714996
## iter 200 value 0.209654
## iter 210 value 0.066710
## iter 220 value 0.030412
## iter 230 value 0.014036
## iter 240 value 0.006702
## iter 250 value 0.004146
## iter 260 value 0.001844
## iter 270 value 0.001128
## iter 280 value 0.000744
## iter 290 value 0.000462
## iter 300 value 0.000308
## iter 310 value 0.000265
## iter 320 value 0.000231
## final  value 0.000076
## converged
```

```
coefs2 <- coef(mod_mult2)
coefs_reng2 <- coefs2[1, , drop =FALSE]
coefs2 <- rbind(coefs_reng2, coefs2)
coefs2[1 , ] <- 0
```

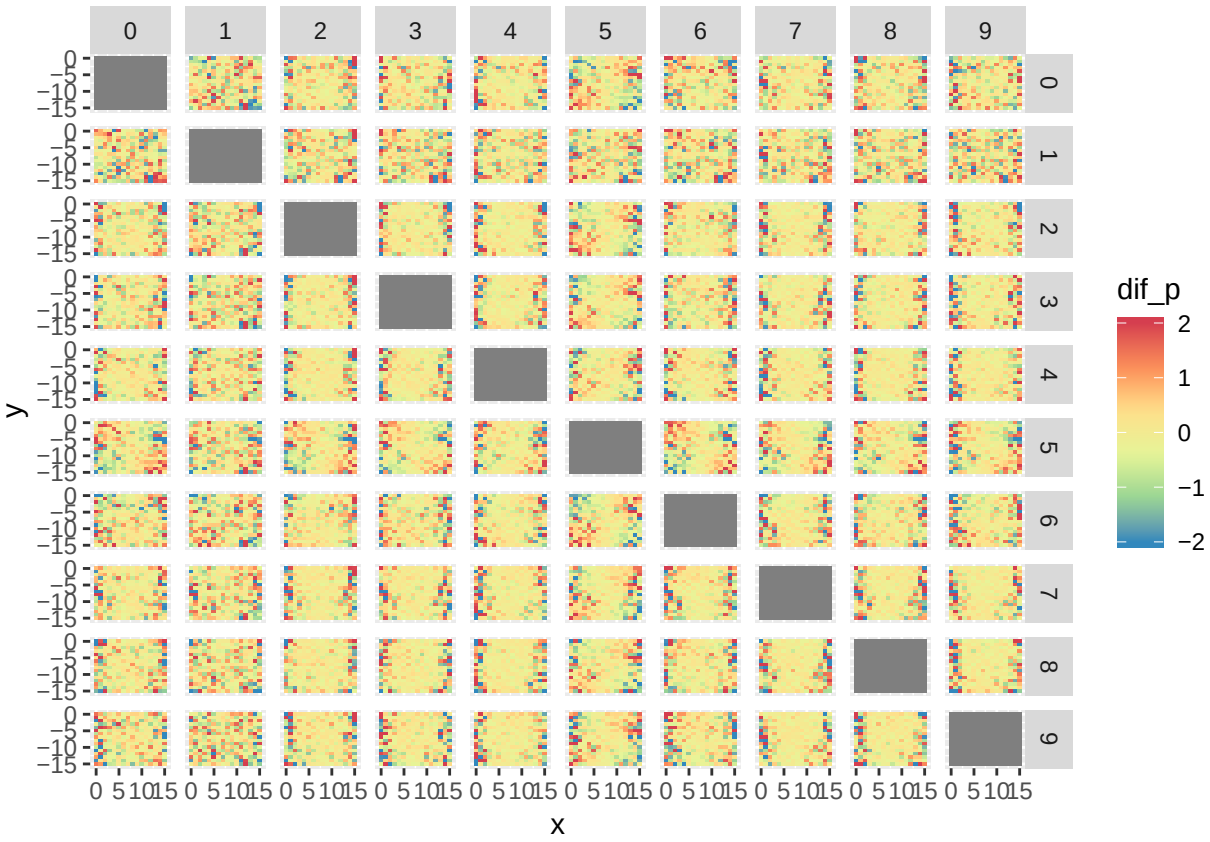
```
beta_df2 <- coefs2[,-1] %>% as.data.frame %>%
  mutate(digito = 0:(nrow(coefs2)-1)) %>%
  gather(pixel, valor, contains('pixel')) %>%
  separate(pixel, into = c('str','pixel_no'), sep='_') %>%
  mutate(x = (as.integer(pixel_no)-1) %% 16, y = -((as.integer(pixel_no)-1) %/% 16))
```

```
tab_coef2 <- beta_df2 %>% select(digito, x, y, valor)
tab_coef_12 <- tab_coef2
names(tab_coef_12) <- c('digito_1','x','y','valor_1')
tab_cruzada2 <- full_join(tab_coef_12, tab_coef2) %>% mutate(dif = valor_1 - valor)
```

```
## Joining, by = c("x", "y")
```

```
tab_cruzada2 <- tab_cruzada2 %>% group_by(digito, digito_1) %>%
  mutate(dif_s = (dif - mean(dif))/sd(dif)) %>%
  mutate(dif_p = pmin(pmax(dif_s, -2), 2))
```

```
ggplot(tab_cruzada2, aes(x=x, y=y)) + geom_tile(aes(fill = dif_p)) +
  facet_grid(digito_1~digito) + scale_fill_distiller(palette = "Spectral")
```



En el modelo que llega a converger se ve el sobre ajuste, ya que no es distinguible a qué dígito corresponden los pixeles en la muestra de prueba.