Proyecto Final

Métodos Numéricos y Optimización

Equipo1

Paulina Gómez-Mont Wiechers `113018`

Rodrigo Floriano Verástegui `111954`

Carlos Alberto García Michel `181987`

1. Introducción

El objetivo de este proyecto es explorar una aplicación del modelo de **regresión logística multinomial** sobre el problema descrito en la competencia de Kaggle llamada "Reducing Commercial Aviation Fatalities", cuyo objetivo es identificar si los pilotos de aviones presentan algún estado cognitivo que pueda ser peligroso en diversas simulaciones de vuelo. Este modelo de clasificación se utiliza para asignar probabilidades de ocurrencia de cada una de las categorías a las que un fenómeno puede pertenecer.

1.1. Descripción de los datos

1.1.1. Variable respuesta

La información disponible disponible está conformada por datos psicológicos de 18 pilotos que fueron sujetos a distracciones en un ambiente de simulación de vuelo. El conjunto de entrenamiento está conformado por experimentos controlados, mientras que el conjunto de prueba consiste en datos en vuelos completos.

Con el objetivo de aislar el estado cognitivo de los pilotos ante situaciones, los pilotos fueron expuestos a distracciones con la intención de llevarlos a alguno de los siguientes estados cognitivos:

- **Atención Canalizada (`CDA`)** - Es la concentración en una tarea excluyendo todas las demás. Se induce obligando a los pilotos a jugar videojuegos.
- **Atención Desviada (`DA`)** - Es la desviación de la atención debido a acciones o procesos de pensamiento asociados a la toma de decisiones. Se induce resolviendo periodicamente problemas matemáticos antes de regresar a la tarea monitoreada.
- **Sorpresa (`SS`)** - Es inducido al presentar al sujeto de estudio cortes de películas en dónde el observador pueda asustarse/sorprenderse.

Cada experimento consistió en monitorear en un par de pilotos de cada tripulación (`crew`) expuestos *a priori* o inducidos desde el **estado base** `A` a situaciones o experimentos que pudieran inducir a alguna condición cognitiva en los pilotos. Por ello, en cada observación se tiene solamente la observación de un ejercicio asociado a los estados `CDA, DA` o `SS`. Esta información está guardada en la variable `experiment` (que mencionaremos más adelante). En el evento final se observa si el individuo alcanzó **solamente** uno de los estados `CDA, DA, SS` o `A`, en donde `A` implica el fallo en conseguir la condición cognitiva esperada. Por ejemplo, en el experimento `CDA`, los pilotos estaban en un estado base `A` o en el estado `CDA` y se les indicó jugar videojuegos, por lo que se espera que presenten la condición `CDA` o la ausencia de ella (`A`). Esta información se encuentra almacenada en la variable categórica `event`.

Por lo anterior, el problema mencionado al prrincipio del proyecto se traduce en generar un clasificador para cada simulación realizada en las categorías `CDA, DA, SS` o `A`.

1.1.2. Variables independientes

Además de la variable `event`, el estudio registró la siguiente lista de variables:

1. `crew` (*entera*): Un identificador único por cada par de pilótos. Hay 9 pares en el estudio.
2. `experiment`(*categórica*): Experimento al que fueron expuestos los pilotos (`CA`,`DA` o `SS`).
3. `time` (*numérica*): Tiempo en el experimento.
4. `seat` (*binaria*): 0 si el sujeto está en el asiento izquierdo y 1 en el derecho.
5. Variables con el prefijo `eeg` (*numéricas*): Registros de encefalogramas. (`eeg_fp1`, `eeg_f7`, `eeg_f8`, `eeg_t4`, `eeg_t6`, `eeg_t5`, `eeg_t3`, `eeg_fp2`, `eeg_o1`, `eeg_p3`, `eeg_pz`, `eeg_f3`, `eeg_fz`, `eeg_f4`, `eeg_c4`, `eeg_p4`, `eeg_poz`, `eeg_c3`, `eeg_cz`, `eeg_o2`)
6. `ecg` (*numérica*): Señal de electrocardiográma.
7. `r` (*numérica*): Respiración del piloto.
8. `gsr`(*numérica*): Actividad electrodermal.

2. Modelo logístico multinomial

El objetivo de este problema minimizar la **devianza** utilizando los parámetros del modelo como variables de control, dados los datos disponibles. La descripción del modelo es la siguiente:

Sean $g \in \{1, \ldots, K\} = \mathcal{C}$ las clases a las que pertenecen las observaciones $x^{(i)},\ i = 1, \ldots, N$ obtenidas de una vector aleatorio X. Las probabilidades de pertenecer a una clase, condicional a una observación x son:

$p_g(x) = P(G = g|X = x),\ g \in \{1, \ldots, K\}$ en donde:

$$
\begin{aligned}
p_1(x) &= \exp\left(\beta_{0,1} + \beta_{1,1}x_1 + \ldots + \beta_{p,1}x_p\right)/Z \\
p_2(x) &= \exp\left(\beta_{0,2} + \beta_{1,2}x_2 + \ldots + \beta_{p,2}x_p\right)/Z \\
&\ldots \\
p_{K-1}(x) &= \exp\left(\beta_{0,K-1} + \beta_{1,K-1}x_2 + \ldots + \beta_{p,K-1}x_p\right)/Z \\
p_K(x) &= 1/Z \\
Z &= 1 + \sum_{j=1}^{K-1} \exp\left(\beta_0^j + \beta_1^j x_1 + \ldots + \beta_p^j x_p\right)
\end{aligned}
$$

Formalmente, problema de **minimización de devianza** mencionado está descrito como:

$$
\begin{aligned}
\min_{\boldsymbol{\beta}} \qquad & D(\boldsymbol{\beta}) = -2\sum_{i=1}^{N} \log p_{g^{(i)}}(x^{(i)}) \\
\boldsymbol{\beta} = \big(&\beta_0^1, \beta_1^1, \ldots, \beta_p^1, \beta_0^2, \beta_1^2, \ldots, \beta_p^2, \ldots \beta_0^{K-1}, \beta_1^{K-1}, \ldots, \beta_p^{K-1}\big)
\end{aligned}
\tag{1}
$$

3. Métodos de descenso

Sea $f : \mathbb{R}^p \to \mathbb{R}$ donde $f \in \mathcal{C}^2(\mathbb{R}^p)$ es una función convexa. Se busca resolver el problema:

$$
\underset{x}{\text{minimizar}} \qquad f(x) \tag{2}
$$

Dado que es un problema convexo, sabemos que $\nabla^2 f(x)$ es simétrica positiva definida en $\mathbb{R}^p$, por lo que buscamos un punto x^* que cumpla las condiciones necesarias de primer orden: $\nabla f(x^*) = 0$.

Una familia de algoritmos para resolver este problema son los **métodos de descenso**. Estos métodos buscan encontrar un óptimo local (y global en el caso de funciones convexas) partiendo de un punto inicial x_0 y recorriendo una senda de actualiazaciones de la forma $x_{k+1} = x_k + \eta_k p_k$ de forma que cumpla $p_k^T \nabla f(x_k) < 0$. Esta condición implica que p_k es una **dirección de descenso**: f se reduce a lo largo de p_k. La variable η_k es la longitud recorrida sobre el vector de descenso p_k, conocida como *tamaño del paso*. Se puede probar que

$\{x_k\}_{k=0}^{\infty} \underset{n\to\infty}{\longrightarrow} x^*$. De forma operacional, esta familia de algoritmos tienen la siguiente estructura:

Dado un **punto inicial** x y la iteración inicial $k \longleftarrow 0$

la **tolerancia** τ y el **número máximo de iteraciones** $\overline{k}$

la **dirección de descenso** p

la **función de selección de tamaño del paso** g

Mientras $\|\nabla f(x)\| > \tau$ y $k \leq \overline{k}$:

$\eta \longleftarrow g(x)$

$x \longleftarrow x + \eta p$

$k \longleftarrow k + 1$

El negativo del gradiente ($p_k = -\nabla f(x_k)$) es una dirección de descenso (**método de descenso en gradiente** o **máximo descenso**) ya que cumple que $p_k^T \nabla f(x_k) = -\nabla f(x_k)^T \nabla f(x_k) = -\|\nabla f(x_k)\|_2^2 < 0$. Otra función de descenso es la dirección de Newton $p_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$, debido a que también cumple $p_k^T \nabla f(x_k) = -\nabla f(x_k)^T \nabla^2 f(x_k)^{-1} \nabla f(x_k) < 0$ (ya que por convexidad de f se cumple que $\nabla^2 f(x_k)$ es simétrica positiva definida por lo que su inversa también lo es).

4. Implementación

Para el problema de "Reducing Commercial Aviation" se definen las variables de la siguiente forma:

- $N = 4,867,421$ observaciones (50 % de entrenamiento y 50 % de prueba)
- Clases: $\mathcal{C} = \{\texttt{CA}, \texttt{DA}, \texttt{SS}, \texttt{A}\}$
- Variables independientes ($p = 26$): $X = ($`crew`, `time`, `seat`, `eeg_fp1`, `eeg_f7`, `eeg_f8`, `eeg_t4`, `eeg_t6`, `eeg_t5`, `eeg_t3`, `eeg_fp2`, `eeg_o1`, `eeg_p3`, `eeg_pz`, `eeg_f3`, `eeg_fz`, `eeg_f4`, `eeg_c4`, `eeg_p4`, `eeg_poz`, `eeg_c3`, `eeg_cz`, `eeg_o2`, `ecg`, `r`, `gsr`). Se omitió la variable `experiment` para tener solamente vqriables numéricas.
- Función a minimizar: $D(\boldsymbol{\beta})$ con $\boldsymbol{\beta} \in \mathbb{R}^{81}$
- Dirección de descenso: $p_k = -\nabla f(x_k) \;\; \forall\, k \geq 0$ (**método de descenso en gradiente**)
- Función de tamaño del paso constante: $\eta_k = g(x_k) = \eta$
- Tolerancia del algoritmo: Dada la comlejidad y tamaño del problema, no se impuso una tolerancia mínima a la norma del gradiente (no hay un criterio de paro cuando $\|\nabla f(x)\| <= \tau$), pero sí se impuso un criterio al número de iteraciones $\overline{k}$.

La implementación del problema se realizó tres veces para el **método de descenso en gradiente** y con diferentes criterios para $\boldsymbol{\beta}_0$,η y $\overline{k}$:

4.1. Primera implementación:

- Punto inicial $\boldsymbol{\beta}_0 = (4, \mathbf{1}_{1\times 80})^T$
- Función de tamaño del paso constante: $\eta_k = g(x_k) = \eta = 5 \times 10^{-7}$
- Tolerancia del algoritmo: $\overline{k} = 1900$

4.2. Segunda implementación:

Basándose en el resultado provisto por el ajuste previo:

- Punto inicial $\boldsymbol{\beta}_0 = \boldsymbol{\beta}_1^*$ donde $\boldsymbol{\beta}_1^*$ es el resultado de la primera implementación.
- Función de tamaño del paso constante: $\eta_k = g(x_k) = \eta = 10^{-7}$
- Tolerancia del algoritmo: $\overline{k} = 3000$

4.3. Tercera implementación:

Basándose en el resultado provisto por el ajuste previo:

- Punto inicial $\boldsymbol{\beta}_0 = \boldsymbol{\beta}_2^*$ donde $\boldsymbol{\beta}_2^*$ es el resultado de la segunda implementación.
- Función de tamaño del paso constante: $\eta_k = g(x_k) = \eta = 5 \times 10^{-8}$
- Tolerancia del algoritmo: $\overline{k} = 1000$

Las razones por las que se realizó este modelo por partes son dos:

1. Dado el tiempo de cómputo del problema, era preferible ir guardando por tramos la información generada, en caso de alguna *error fatal* de **R**.
2. Para poder modificar manualmente el valor de η.

El último punto es crucial en el algoritmo implementado. Utilizar $\eta_k = g(x_k) = \eta$ puede ahorrar cálculos en la búsqueda de un η que cumpla $f(x_{k+1}(\eta)) < f(x_k)$. Por otro lado, para un η fijo puede existir una iteración q en donde $f(x_{q+1}(\eta)) > f(x_q)$ que provoque una falla en el algoritmo. La forma correcta de abordar el problema es calcular en cada iteración una η^* que cumpla con la condición de **Armijo-Goldstein**:

$$f(x_{k+1}) = f(x_k + \eta_k^* p_k) \leq f(x_k) + c\eta_k^* p_k^T \nabla f(x_k), c \in (0,1)$$

Existen varios métodos para encontrar η^*. Los más utilizados son los métodos de **búsqueda de línea** y de **regiones de confianza**. Se deja la búsqueda de este parámetro como extensión al proyecto presente, tomando en cuenta que es un aspecto fundamental del algoritmo utilizado.

En cuestión computacional, el lenguaje de programación utilizado fue el lenguaje estadístico **R**. Es necesario recalcar que para evaluar el desempeño de este modelo, se utilizó la mitad de los datos para realizar las estimaciones de parámetros y se comparó con el ajuste realizado con el algoritmo incluido en el paquete **nnet** para esta clase de modelos.

Las funciones utilizadas para hacer los cálculos correspondientes son:

```r
# Función logística multinomial
pred_ml <- function(x, beta){
  p <- ncol(x)
  K <- length(beta)/(p+1) + 1
  beta_mat <- matrix(beta, K - 1, p + 1 , byrow = TRUE)
  u_beta <- exp(as.matrix(cbind(1, x)) %*% t(beta_mat))
  Z <- 1 + apply(u_beta, 1, sum)
  p_beta <- cbind(u_beta, 1)/Z
  as.matrix(p_beta)
}

# Cálculo de devianza
devianza_calc <- function(x, y){
  dev_fun <- function(beta){
    p_beta <- pred_ml(x, beta)
    p <- sapply(1:nrow(x), function(i) p_beta[i, y[i]+1])
   -2*sum(log(p))
  }
  dev_fun
}

# Cálculo de gradiente
grad_calc <- function(x_ent, y_ent){
  p <- ncol(x_ent)
  K <- length(unique(y_ent))
  y_fact <- factor(y_ent)
```

```
  # Matriz de indicadoras de clase
  y_dummy <-  model.matrix(~-1 + y_fact)

  salida_grad <- function(beta){
    p_beta <-  pred_ml(x_ent, beta)
    e_mat <-  (y_dummy  - p_beta)[, -K]
    grad_out <- -2*(t(cbind(1,x_ent)) %*% e_mat)
    as.numeric(grad_out)
  }
  salida_grad
}

# Método de descenso con información de iteraciones
descenso <- function(n, z_0, eta, h_deriv, dev_fun){
  z <- matrix(0,n, length(z_0))
  z[1, ] <- z_0
  for(i in 1:(n-1)){
    z[i+1, ] <- z[i, ] - eta * h_deriv(z[i, ])
    if(i %% 100 == 0){
      print(paste0(i, ' Devianza: ', dev_fun(z[i+1, ])))
    }
  }
  z
}
```

5. Resultados

En esta sección únicamente se reportarán los resultados de la tercera implentación debido a que el resultado corresponde a las iteraciones más avanzadas.

```
# Preparación de datos
set.seed(530)
ent <- sample_frac(datos,0.5) %>% data.frame()
x_ent <- ent %>% select(-c(experiment,event)) %>% as.matrix
y_ent <- ent %>% select(c(event))
x_ent_s <- x_ent %>% scale()
```

```r
y_ent$event[y_ent$event == "A"] <- 0
y_ent$event[y_ent$event == "B"] <- 1
y_ent$event[y_ent$event == "C"] <- 2
y_ent$event[y_ent$event == "D"] <- 3
y_ent <- unlist(y_ent) %>% as.numeric()

# Primera implementación
z <- descenso(1900, c(-2,rep(-0.05, (ncol(x_ent_s)+1)*3-1)), eta=0.0000005,
                        h_deriv = grad, dev_fun = dev_ent)

# Segunda implementación
beta <- z[1900,]
z_1 <- descenso(3000, beta, eta=0.0000001,
                        h_deriv = grad, dev_fun = dev_ent)
# Tercera implementación
beta <- z_1[3000,]
z_2 <- descenso(1000, beta, eta=0.00000005,
                        h_deriv = grad, dev_fun = dev_ent)
```

5.1. Salida en R para el algoritmo de máximo descenso

```
[1] "100 Devianza: 4447411.98439558"
[1] "200 Devianza: 4447411.95722662"
[1] "300 Devianza: 4447411.93229635"
[1] "400 Devianza: 4447411.90941638"
[1] "500 Devianza: 4447411.88841463"
[1] "600 Devianza: 4447411.86913385"
[1] "700 Devianza: 4447411.85143031"
[1] "800 Devianza: 4447411.8351726"
[1] "900 Devianza: 4447411.8202405"

[1] "Las Betas son:"

 [1]  2.5152357792 -0.0094035018 -0.0783573952 -0.0934118017  0.0469958022
 [6] -0.0113642982 -0.0056523805 -0.0049762573  0.0052138317  0.0098957216
```

```
[11] -0.0131825552 -0.0048777917 -0.0008949587 -0.0069815220  0.0074359649
[16]  0.0154885829 -0.0040054927  0.0012584697  0.0324663724 -0.0108931834
[21] -0.0002014416 -0.0498430195  0.0070941043 -0.0038071022 -0.0308156835
[26]  0.1324762361  0.1292709001 -0.5921768090  0.0031653976 -0.0169236963
[31] -0.1694706357  0.1011852804  0.0117589079 -0.0416747880 -0.0398222835
[36] -0.0048950370  0.0167990229 -0.0440089703  0.0603609373 -0.0056715425
[41] -0.0328480834 -0.0303276355  0.0109307223  0.0112259228  0.0083319640
[46] -0.0092201229  0.0365799910 -0.0170533143 -0.0157978921  0.0032562951
[51]  0.0076686743 -0.1199428772  0.2875999898  0.3960317808  1.9587124041
[56] -0.0768916597 -0.0249768461 -0.0453397102  0.0028448006  0.0095957968
[61]  0.0148144586  0.0110999584  0.0030102485  0.0084616783 -0.0036447172
[66] -0.0537275889 -0.0025480879  0.0119397764  0.0127792051  0.0049566584
[71] -0.0046895205  0.0049841605  0.0209726336  0.0008676623  0.0155065144
[76] -0.0684265008  0.0074996534 -0.0147433618 -0.2301021159  0.1473359281
[81]  0.3515367148

[1] "El valor de la norma del gradiente es: 51.28602"

[1] "El valor de la devianza es: 4447412"
```

Observamos que $\|\nabla f(x_{\overline{k}})\|_2 = 51.28602$, el cual se encuentra lejos de ser 0. A pesar de ello, recordemos que por la escala del problema y las características de la máquina, podríamos tener un margen de error. Realizamos el comparativo con el modelo implementado en **nnet**:

```
# Calcula el modelo
mod_mult <- multinom(y_ent ~ x_ent_s, data = ent, MaxNWt=100000, maxit = 500)
# Imprimimos modelo
mod_mult
```

5.2. Salida en R para el algoritmo multinom de nnet:

```
Call:
multinom(formula = y_ent ~ x_ent_s, data = ent, MaxNWt = 1e+05,
    maxit = 500)

Coefficients:
  (Intercept)  x_ent_screw x_ent_stime x_ent_sseat x_ent_seeg_fp1
```

```
1  -3.1073982  0.012565283  0.06144471 -0.07558367     0.05403301
2  -0.5565312 -0.067488303  0.05338108  0.04807909    -0.04418044
3  -2.5151817  0.009372292  0.07835066  0.09279014    -0.04544649
  x_ent_seeg_f7 x_ent_seeg_f8 x_ent_seeg_t4 x_ent_seeg_t6 x_ent_seeg_t5
1    0.02293653  -0.036350605  -0.034792220  -0.010021405   0.006834137
2    0.02095067   0.020450258   0.016077963  -0.002200095  -0.001439793
3    0.01165121   0.006773187   0.004826149  -0.005396417  -0.009840566
  x_ent_seeg_t3 x_ent_seeg_fp2 x_ent_seeg_o1 x_ent_seeg_p3 x_ent_seeg_pz
1  -0.030709950    0.065679350 -0.0047799718  -0.025794735  -0.037767649
2   0.009537926   -0.048803899 -0.0016579392   0.018920002   0.005344146
3   0.012989467    0.002590312  0.0007930064   0.007093219  -0.007428398
  x_ent_seeg_f3 x_ent_seeg_fz x_ent_seeg_f4 x_ent_seeg_c4 x_ent_seeg_p4
1  -0.004589053  0.0152738112   0.007096826   -0.04143070    0.04709076
2  -0.010524619 -0.0006808108   0.003717972   -0.01149388    0.01174165
3  -0.015511365  0.0039267864  -0.001253713   -0.03275954    0.01149405
  x_ent_seeg_poz x_ent_seeg_c3 x_ent_seeg_cz x_ent_seeg_o2  x_ent_secg
1  -0.0169383137    0.03422764 -0.0039272991   0.011448882 -0.08914547
2   0.0157128325   -0.01857742  0.0004018396  -0.010936591 -0.19928725
3   0.0001378756    0.04938889 -0.0069241602   0.003792776  0.03081214
     x_ent_sr x_ent_sgsr
1  0.15464419  0.2667631
2  0.01485108  0.2222638
3 -0.13187653 -0.1292517


Residual Deviance: 4447412
AIC: 4447574
```

```
dev_ent(coef(mod_mult))
```

```
[1]  14835867
```

```
grad(coef(mod_mult)) %>% norm(type = '2')
```

```
[1]  2936879
```

Podemos observar que la devianza con este modelo es mayor. Esto implica que el resultado del **método de máximo descenso** tiene un mejor desempeño que el que utiliza **nnet**. A pesar de ello, pueden existir factores que hayn contribuido a por lo que es necesario que desmenucemos algunos:

1. El punto inicial en el que comenzamos puede estar más cercano al óptimo global.
2. La tasa de convergencia puede ser muy lenta debido a que el segundo modelo sí busca tener procesos de actualización automatizados.
3. La complejidad es mayor en el segundo proceso, debido a la simplicidad asociada al primer método.

Como mencionamos previamente, extensiones como la búsqueda de acttualización óptima para η son elementos que se deben agregar, así como un análisis de convergencia más riguroso. Por otro lado, sabemos que en la práctica, el hacer un paro prematuro en modelos logísticos es recomendable a realizar todas las iteraciones. Por ello, hace falta hacer un comparativo utilizando los datos que no fueron seleccionados en la muestra de entrenamiento.