Estadística [continuación]

Santander, 2019-2020

Optimización de la función de coste para un caso general

- La ecuación de minimización del “loss” en la que el gradiente se iguala a 0 no siempre es resoluble.
- En esos casos es preciso aplicar algoritmos numéricos de optimización basados en el gradiente.
- Es el caso por ejemplo del “loss” de la regresión logística, en el que como vimos:

$$Loss = \frac{-1}{N} \sum_{i=0}^{N} y^{(i)} \log\left(\sigma\left(a + bx^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(a + bx^{(i)}\right)\right)$$

- Para minimizar este tipo de funciones se suele utilizar el método “gradient descent” o parecidos
  - Recordemos que es particularmente rápido el “stochastic gradiente descent”
- Recordemos también que este tipo de algoritmos utilizan el gradiente como elemento central:

$$\alpha_{n+1} = \alpha_n + \lambda \nabla_\alpha Loss$$

- Se trata de procesos iterativos en el que progresivamente nos acercamos al mínimo.

La funcion optim de R (I)

- Nosotros vamos a utilizar la función optim implementada en R para resolver este problema.

General-purpose Optimization

Description

General-purpose optimization based on Nelder–Mead, quasi-Newton and conjugate-gradient algorithms. It includes an option for box-constrained optimization and simulated annealing.

Usage

```
optim(par, fn, gr = NULL, ...,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN",
                 "Brent"),
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)

optimHess(par, fn, gr = NULL, ..., control = list())
```

Arguments

`par`
: Initial values for the parameters to be optimized over.

`fn`
: A function to be minimized (or maximized), with first argument the vector of parameters over which minimization is to take place. It should return a scalar result.

`gr`
: A function to return the gradient for the `"BFGS"`, `"CG"` and `"L-BFGS-B"` methods. If it is `NULL`, a finite-difference approximation will be used.

  For the `"SANN"` method it specifies a function to generate a new candidate point. If it is `NULL` a default Gaussian Markov kernel is used.

`...`
: Further arguments to be passed to `fn` and `gr`.

`method`
: The method to be used. See 'Details'. Can be abbreviated.

`lower, upper`
: Bounds on the variables for the `"L-BFGS-B"` method, or bounds in which to *search* for method `"Brent"`.

`control`
: A list of control parameters. See 'Details'.

`hessian`
: Logical. Should a numerically differentiated Hessian matrix be returned?

La funcion optim de R (II)

- ¿Cuáles son los parámetros más importantes de optim?
- Vector de parámetros inicial
  - Tal y como hemos dicho la optimización es un proceso iterativo
  - Necesitamos especificar a optim cuál va a ser nuestro punto de partida (los parámetros iniciales)
- Función de coste
  - Obviamente la función optim necesita saber cuál es la función que debe minimizar
  - Esta función dependerá de los parámetros pero también de los datos (x, y)
  - Esta función siempre devuelve un número
- Función gradiente
  - Para realizar la minimización optim necesita saber cuál es el gradiente de dicha función
  - Esta función dependerá también de los parámetros, pero también de los datos (x,y)
  - Esta función siempre devuelve un vector de la misma longitud que el vector de parámetros
- Método:
  - Método de optimización

La funcion optim de R (III)

- Estructura de uso de la función optim:

```
Loss < - function(alpha, x, y) {

        #alpha es el vector de parámetros
        #x es una matriz que contiene a las características (matriz X)
        #y es una matriz que contiene a las variables dependientes (matriz y)
        return(cost) #un número
}

Gradient < - function(alpha, x, y) {

        #alpha es el vector de parámetros
        #x es una matriz que contiene a las características (matriz X)
        #y es una matriz que contiene a las variables dependientes (matriz y)
        return(gr) # un vector
}

optim(par=initial_alpha, x=x, y=y, fn = Loss, gr = Gradient, method="BFGS")
```

La funcion optim de R: Ejemplo (I)

- Supongamos dos distribuciones gaussianas (*) de dos dimensiones con parámetros:

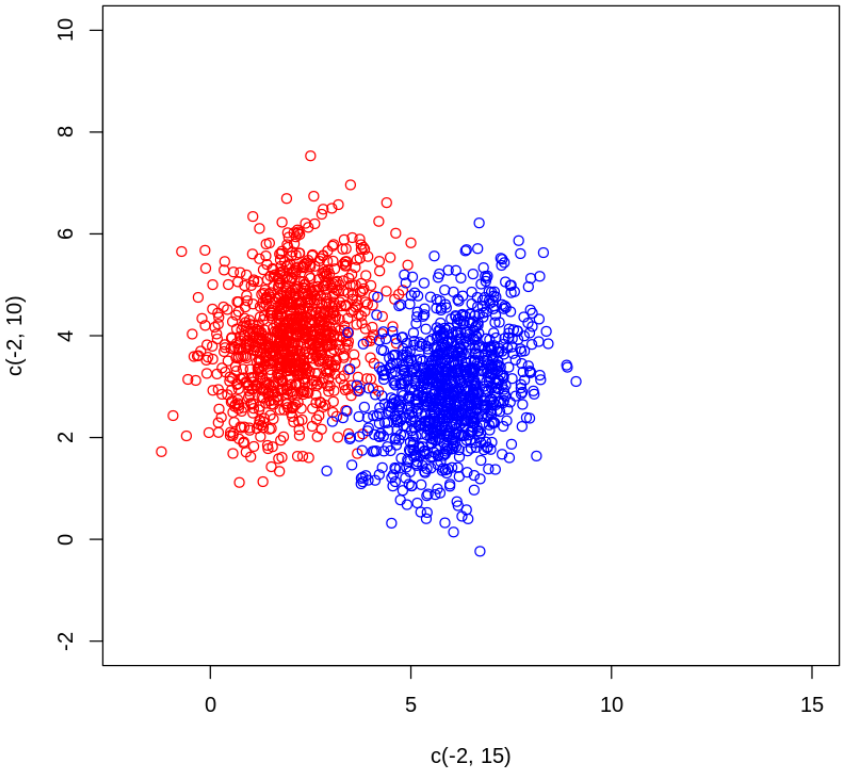
$$p(y=0|x_1,x_2)=\frac{1}{\sqrt{2\pi det\,(Cov_1)}}e^{\frac{-1}{2}(x-\mu_1)^T Cov_1 (x-\mu_1)}$$

$$p(y=0|x_1,x_2)=\frac{1}{\sqrt{2\pi det\,(Cov_2)}}e^{\frac{-1}{2}(x-\mu_2)^T Cov_2 (x-\mu_2)}$$

$$\mu_1=\begin{bmatrix}2\\4\end{bmatrix} \qquad Cov_1=\begin{bmatrix}1 & 0.3\\0.3 & 1\end{bmatrix} \qquad x=\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$$\mu_2=\begin{bmatrix}6\\3\end{bmatrix} \qquad Cov_2=\begin{bmatrix}1 & 0.3\\0.3 & 1\end{bmatrix}$$

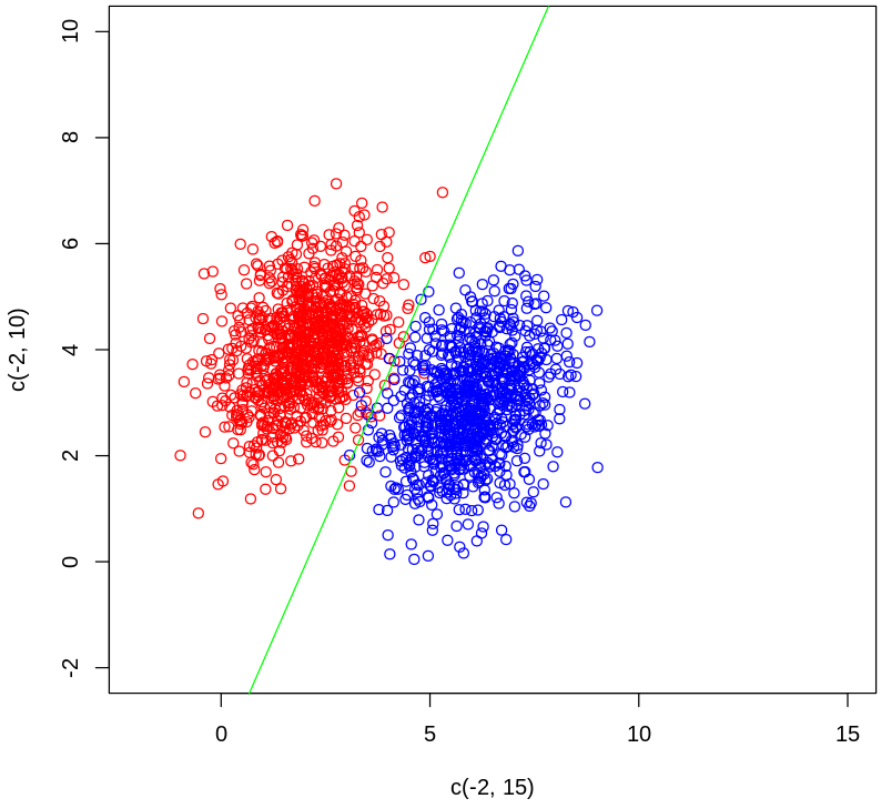
$$Z=\alpha_0+\alpha_1 x_1+\alpha_2 x_2$$



(*) En R pueden generarse estas distribuciones usando MASS::mvrnorm

La función optim de R: Ejemplo (II)

- Utilizando el esquema propuesto para el uso de de la función optim corremos este ejemplo.



La función optim de R: Ejemplo (III)

- Recordemos que optim devuelve el valor de los parámetros para los que el Loss es mínimo
- ¿Cómo podemos obtener la ecuación de la frontera?
- Basta con observar que si queremos poner la frontera en el punto medio de la sigmoide…
- … tenemos que buscar cuando sigma(z) = 0.5

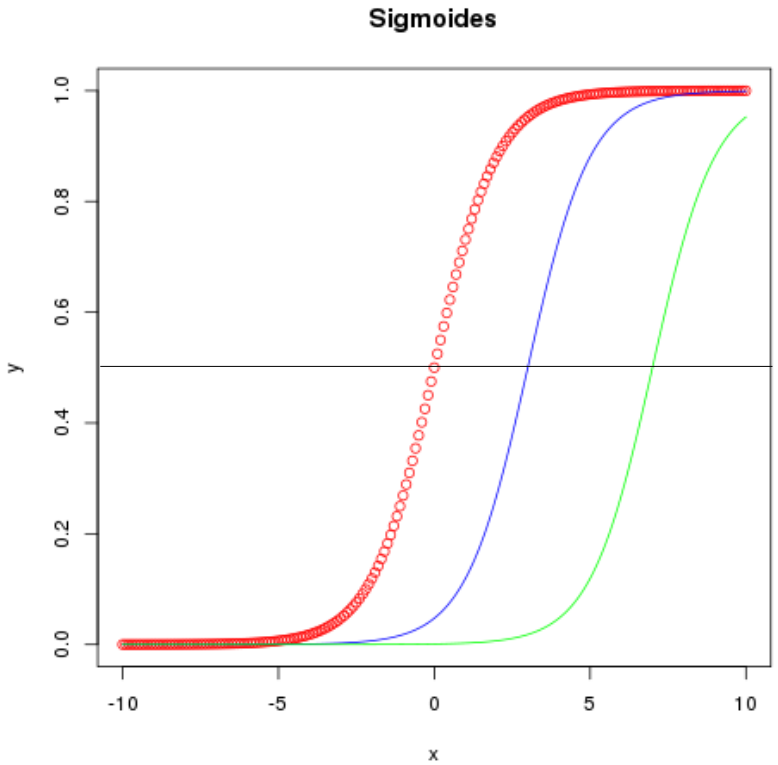
$$\sigma(x) = \frac{1}{1+e^{-z}} = 0.5$$

- Lo que implica que Z = 0
- Pero como tenemos que:

$$Z = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$$

- Entonces:

$$Z = 0 = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \Rightarrow x_2 = \frac{-\alpha_0}{\alpha_2} - \frac{\alpha_1}{\alpha_2} x_1$$



Interpretación de la sigmoide (I)

- La sigmoide que hemos utilizado con los parámetros que minimizan la función de coste es nuestro **modelo de probabilidad pdf(y | x)**.

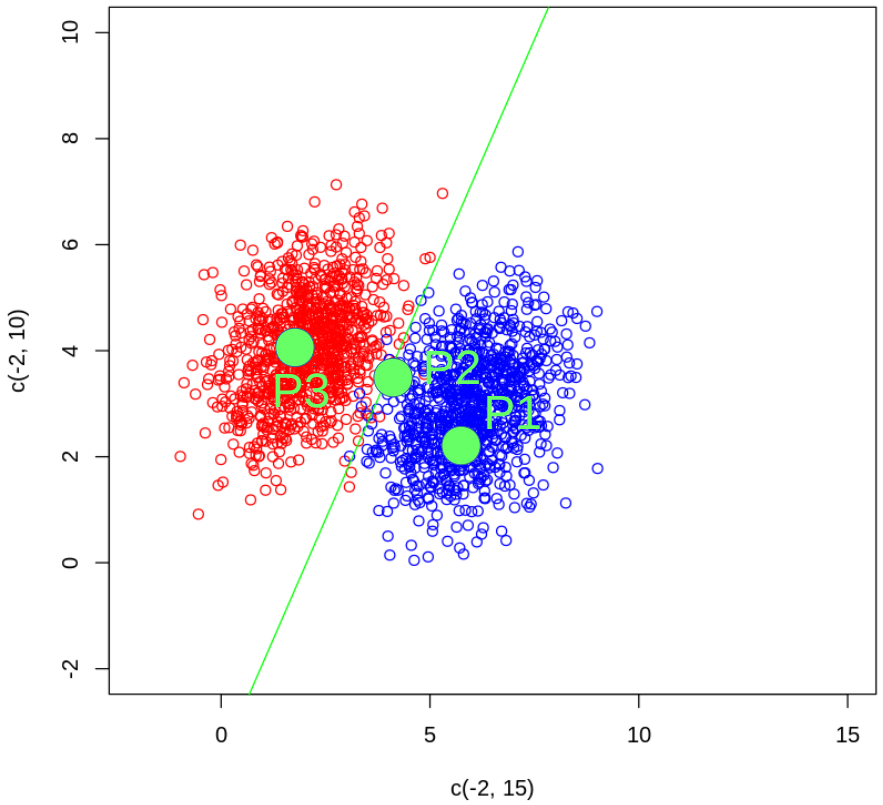
- Cojamos 3 puntos: P1(6, 2), P2(4, 3), P3(2,4)
- Evaluemos el valor de sigma en cada caso:

$$\sigma(x)=\frac{1}{1+e^{-(\alpha_0+\alpha_1 x_1+\alpha_2 x_2)}}$$

$$P1 \rightarrow \sigma(x)=0.9999984$$

$$P2 \rightarrow \sigma(x)=0.7445965$$

$$P3 \rightarrow \sigma(x)=1.397586e\text{-}5$$



Interpretación de la sigmoide (II)

- La manera que tenemos de clasificar consiste en asignar una probabilidad límite a cada categoría
- Obsérvese que antes hemos elegido que esa probabilidad sea P(x) = 0.5 para hacer la frontera
- En ese caso cuando un punto P1 nos daba P(P1) >= 0.5 lo consideramos de la categoría 1
- En el caso contrario en el que el punto P1 nos de P(P1) < 0.5 lo consideramos de la categoría 0
- Sin embargo no es la única opción, ese threshold puede elegirse arbitrariamente según nos interese
- Si escogemos un threshold muy alto **dificultamos que el modelo nos catalogue como 1**
- Si por el contrario ponemos un threshold bajo **dificultamos que el modelo catalogue como 0**

Interpretación de la sigmoide: Errores (I)

- De hecho existen unos cuantos parámetros interesantes que pueden medirse en un clasificador

	El suceso es positivo (A)	El suceso es de negativo (B)
Clasificado positivo (A)	Positivo Verdadero (True Positive) TP	Positivo Falso (False Positive) FP
Clasificado negativo (A)	Negativo Falso (False Negative) FN	Negativo Verdadero (True Negative) TN

Interpretación de la sigmoide: Errores (II)

- Con los números del cuadro anterior se definen varias variables interesantes

	El suceso es positivo (A)	El suceso es de negativo (B)
Clasificado positivo (A)	Positivo Verdadero (True Positive) TP True Positive Rate: TP/(TP+FN)	Positivo Falso (False Positive) FP False Positive Rate: FP/(FP+TN)
Clasificado negativo (A)	Negativo Falso (False Negative) FN False Negative Rate: FN/(TP+FN)	Negativo Verdadero (True Negative) TN True Negative Rate: NV/(FP+TN)

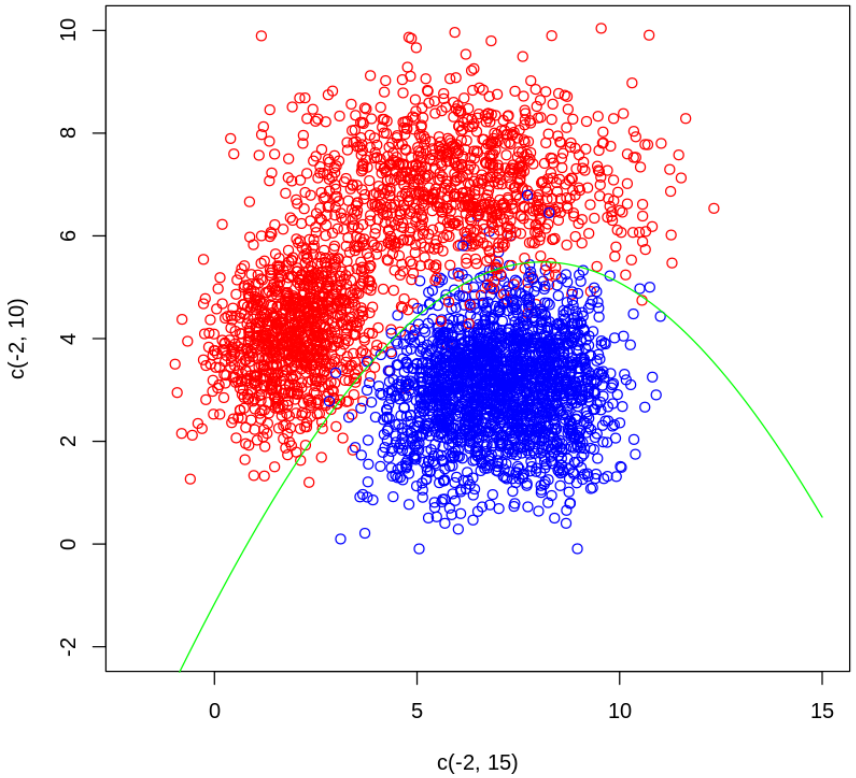
Interpretación de la sigmoide: Errores (III)

- Cambiando el threshold de las sigmoide podemos modificar todos estos rates en función del interés

	El suceso es positivo (A)	El suceso es de negativo (B)
Clasificado positivo (A)	Positivo Verdadero (True Positive) TP True Positive Rate: TP/(TP+FN)	Positivo Falso (False Positive) FP False Positive Rate: FP/(FP+TN)
Clasificado negativo (A)	Negativo Falso (False Negative) FN False Negative Rate: FN/(TP+FN)	Negativo Verdadero (True Negative) TN True Negative Rate: NV/(FP+TN)

Regresión logística con polinomios

- Tal y como se hace en la regresión lineal podemos ampliar nuestro modelo a algo polinomial
- Basta con ampliar la matriz X con las sucesivas potencias de las features y minimizar.



Ejercicio 5

1) Crea una función a la que se le pase como input: la media en el eje x, la media en el eje y, la varianza en el eje x, la varianza en el eje y y la covarianza de x e y, junto con un número de puntos N, y devuelva una matriz con N filas y 2 columnas con los números que salen de la distribución gaussiana de dos dimensiones definidas por los valores de input (usar la función MASS::mvrnorm)
2) Genera una matrix x1 usando la función anterior y tomando: N = 1000, mu_x = 2, mu_y = 4, var_x = var_y = 1, y Cov(x,y) = 0.3. Crea una matriz "y1" con tantas filas como la matriz x y asígnale el valor 0.
3) Repite 2) para otra muestra con N = 1000, mu_x = 6, mu_y = 3, var_x = var_y = 1, y Cov(x,y) = 0.3. Crea una matriz "y2" con tantas filas como la matriz x2 y asígnale el valor 1.
4) Junta las matrices x1, x2, y y1, y2 en una sola matriz x y una sola matriz y.
5) Usando las funciones de coste y gradiente del ejercicio 5, utiliza optim para un modelo en el que z = alpha_0 + alpha_1 x1 + alpha_2 x2, tomando como vector de parámetros inicial el (0, 0, 0). Calcula y pinta la frontera entre ambas distribuciones.
6) Repite otra vez 1, 2, 3 y 4 para obtener otras matrices x e y independientes. Utilizando la "sigma" calculada anteriormente calcula el TPR, TNR, FPR y FNR para valores del threshold = 0.3, 0.5, 0.7.