# INFOF422 bonus 5

## Binary classification and logistic regression

G. Bontempi

### Question

Let us consider a binary classification task where the input $\mathbf{x} \in \Re^2$ is bivariate and the categorical output variable $\mathbf{y}$ may take two values: 0 (associated to red) and 1 (associated to green).

Suppose that the a-priori probability is $p(\mathbf{y} = 1) = 0.2$ and that the inverse (or class-conditional) distributions are the bivariate Gaussian distributions $p(x|\mathbf{y} = 0) = \mathcal{N}(\mu_0, \Sigma_0)$ and $p(x|\mathbf{y} = 1) = \mathcal{N}(\mu_1, \Sigma_1)$ where

- $\mu_0 = [0, 0]^T$
- $\mu_1 = [1, 1]^T$

and both $\Sigma_0$ and $\Sigma_1$ are diagonal identity matrices.

The student should

- by using the R function `rmvnorm`, sample a dataset of $N = 1000$ input/output observations according to the conditional distribution described above,
- visualise in a 2D graph the dataset by using the appropriate colors,
- fit a logistic classifier to the dataset (see details below),
- plot the evolution of the cost function $J(\alpha)$ during the gradient-based minimization,
- plot in the 2D graph the decision boundary.

Logistic regression estimates

$$\hat{P}(\mathbf{y} = 1|x) = \frac{\exp^{x^T \alpha_N}}{1 + \exp^{x^T \alpha_N}} = \frac{1}{1 + \exp^{-x^T \alpha_N}}, \qquad \hat{P}(\mathbf{y} = 0|x) = \frac{1}{1 + \exp^{x^T \alpha_N}}$$

where

$$\alpha_N = \arg\min_{\alpha} J(\alpha)$$

and

$$J(\alpha) = \sum_{i=1}^{N} \left( -y_i x_i^T \alpha + \log(1 + \exp^{x_i^T \alpha}) \right)$$

Note that $\alpha$ is the vector $[\alpha_0, \alpha_1, \alpha_2]$ and that $x_i = [1, x_{i1}, x_{i2}], i = 1, \ldots, N$.

The value of $\alpha_N$ has to be computed by gradient-based minimization of the cost function $J(\alpha)$ by perfoming $I = 200$ iterations of the update rule

$$\alpha^{(\tau)} = \alpha^{(\tau-1)} - \eta \frac{dJ(\alpha^{(\tau-1)})}{d\alpha}, \quad \tau = 1, \ldots, I$$

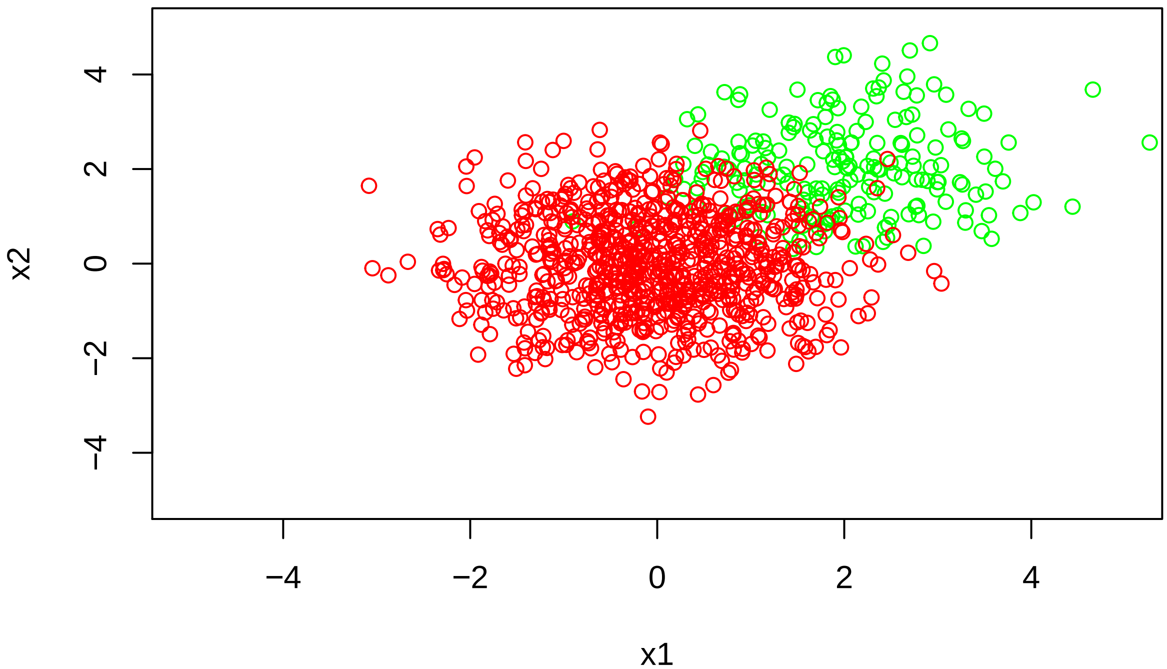
where $\alpha^{(0)} = [0, 0, 0]^T$ and $\eta = 0.001$.

## Data generation

```
library(mvtnorm)
set.seed(0)
N=1000
p1=0.2

mu1<-c(2,2)
Sigma1=diag(2)

mu0<-c(0,0)
Sigma0=diag(2)

N1=N*p1
N0=N*(1-p1)

X1=rmvnorm(N1,mu1,Sigma1)

X0=rmvnorm(N0,mu0,Sigma0)

Y=numeric(N)

Y[1:N1]=1
X=rbind(X1,X0)
X=cbind(numeric(N)+1,X)
```

## Data visualization

```
plot(0,0,xlim=c(-5,5),ylim=c(-5,5),type="n",xlab="x1",ylab="x2")

for (i in 1:N)
  if (Y[i]>0){
    points(X[i,2],X[i,3],col="green")
  }else{
    points(X[i,2],X[i,3],col="red")
  }
```




## Analytical derivation of gradient vector

In order to perform gradient descent we need to compute the derivatives of $J(\alpha)$ with respect to the three parameters $\alpha_0, \alpha_1, \alpha_2$.

Since the scalar function $J : \Re^3 \to \Re$ can be written as

$J(\alpha) = \sum_{i=1}^{N} J_i(\alpha)$ and

$$J_i(\alpha) = -y_i(\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2}) + \log(1 + \exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}})$$

we have

$$\frac{\partial J_i}{\partial \alpha_0} = -y_i + \frac{\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}}{1+\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}}$$
$$\frac{\partial J_i}{\partial \alpha_1} = -y_i x_{i1} + \frac{\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}}{1+\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}} x_{i1}$$
$$\frac{\partial J_i}{\partial \alpha_2} = -y_i x_{i2} + \frac{\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}}{1+\exp^{\alpha_0+\alpha_1 x_{i1}+\alpha_2 x_{i2}}} x_{i2}$$

and

$$\frac{\partial J}{\partial \alpha_0} = \sum_{i=1}^{N} \frac{\partial J_i}{\partial \alpha_0}$$
$$\frac{\partial J}{\partial \alpha_1} = \sum_{i=1}^{N} \frac{\partial J_i}{\partial \alpha_1}$$
$$\frac{\partial J}{\partial \alpha_2} = \sum_{i=1}^{N} \frac{\partial J_i}{\partial \alpha_2}$$

We write then a R function `Jcost` that returns for a given vector $\alpha$ both the scalar value $J(\alpha)$ and the vector $[\frac{\partial J}{\partial \alpha_0}, \frac{\partial J}{\partial \alpha_1}, \frac{\partial J}{\partial \alpha_2}]$.

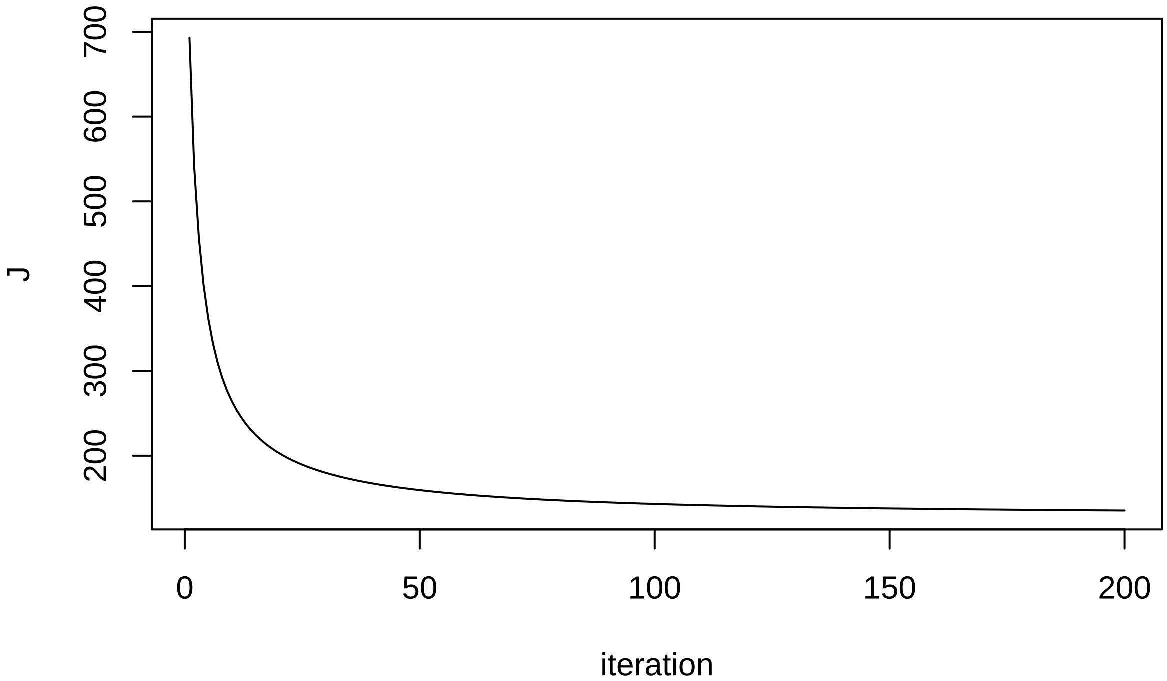
```
Jcost<-function(alpha,X,Y){
  N=length(Y)

  J=0
  dJ=numeric(3)
  for (i in 1:N){
    J=J-Y[i]*t(X[i,])%*%alpha+log(1+exp(t(X[i,])%*%alpha))
    dJ[1]=dJ[1]-Y[i]*X[i,1]+(exp(t(X[i,])%*%alpha))/(1+(exp(t(X[i,])%*%alpha)))*X[i,1]
    ## note that X[i,1]=1

    dJ[2]=dJ[2]-Y[i]*X[i,2]+(exp(t(X[i,])%*%alpha))/(1+(exp(t(X[i,])%*%alpha)))*X[i,2]
    dJ[3]=dJ[3]-Y[i]*X[i,3]+(exp(t(X[i,])%*%alpha))/(1+(exp(t(X[i,])%*%alpha)))*X[i,3]

  }

  list(J=J,dJ=dJ)
}
```

### Gradient-based parametric identification

In the code below we perform $I = 200$ iterations of the gradient descent formula and we store the sequence of values of $J(\alpha^{(\tau)})$ for the graphical visualisation.

The outcome of the gradient-based parametric identification is $\alpha_N = \alpha^{(I)}$.

```
eta=0.001

I=200
J=NULL
alpha=numeric(3)
for (r in 1:I){
  JJ=Jcost(alpha,X,Y)
  J=c(J,JJ$J)
  dJ=JJ$dJ
  alpha[1]<-alpha[1]-eta*dJ[1] ## gradient descent iteration for alpha0
  alpha[2]<-alpha[2]-eta*dJ[2] ## gradient descent iteration for alpha1
  alpha[3]<-alpha[3]-eta*dJ[3] ## gradient descent iteration for alpha2

}


plot(J,type="l",xlab="iteration",ylab="J")
```




The gradient-descent procedure returns that $\alpha_N$ is equal to **-4.8751862, 1.8168868, 1.8694582** .

### Boundary line plot

Once fitted the logistic regression model, the boundary region between the two classes is the set of points where

$$\hat{P}(\mathbf{y}=1|x) = \hat{P}(\mathbf{y}=0|x)$$

that is the set of points where

$$\exp^{x^T \alpha_N} = 1$$

or equivalently

$$x^T \alpha_N = 0$$

Since

$$x^T \alpha_N = \alpha_{0N} + \alpha_{1N} x_1 + \alpha_{2N} x_2$$

the equation of the boundary line in the reference system $(x1, x_2)$ is

$$x_2 = -\frac{\alpha_{0N}}{\alpha_{2N}} - \frac{\alpha_{1N}}{\alpha_{2N}} x_1$$

The code below shows the data points together with the linear decision boundary returned by the logistic regression classifier.

```
plot(0,0,xlim=c(-5,5),ylim=c(-5,5),type="n",xlab="x1",ylab="x2")

for (i in 1:N)
  if (Y[i]>0){
    points(X[i,2],X[i,3],col="green")
  }else{
    points(X[i,2],X[i,3],col="red")
  }
x1=seq(-5,5,by=0.1)
```

```
lines(x1,-alpha[1]/alpha[3]-alpha[2]/alpha[3]*x1,type="l",lwd=4)
```