[MVA]
Probabilistic graphical models
Homework 1

Maha ELBAYAD

22 octobre 2015

Learning in discrete graphical models:

Consider two discrete r.v z and x taking respectively M and K different values with $\mathbb{P}(z = m) = \pi_m$ and $\mathbb{P}(x = k|z = m) = \theta_{mk}$.

Maximum likelihood estimation of π and θ:

Let $\mathcal{D} = \{(x_i, z_i)\}_{1\leq i\leq n}$ be an i.i.d sample of observations.

Our objective is to maximize $\mathbb{P}(\mathcal{D}|\theta, \pi)$. We have:

$$\mathbb{P}(\mathcal{D}|\theta, \pi) = \prod_{i=1}^{n} \mathbb{P}(x_i, z_i|\theta, \pi) = \prod_{i=1}^{n} \mathbb{P}(x_i|z_i, \theta, \pi).\mathbb{P}(z_i|\theta, \pi) = \prod_{i=1}^{n} \theta_{z_i x_i} \pi_{z_i}$$

Let us introduce $\eta_m = \sum_{i=1}^{n} \mathbb{I}(z_i = m)$ and $\xi_{mk} = \sum_{i=1}^{n} \mathbb{I}(z_i = k, x_i = k)$.

It follows that $\sum_{k=1}^{K} \xi_{mk} = \eta_m$ $(m = 1, .., M)$ and $\sum_{m=1}^{M} \eta_m = n$.

Thus, we can rewrite our optimization problem as:

$$\begin{aligned} \text{maximize} \quad & l(\theta, \pi) = \sum_{m=1}^{M} \left[\eta_m \log \pi_m + \sum_{k=1}^{K} \xi_{mk} \log \theta_{mk} \right] \\ \text{subject to} \quad & \sum_{m=1}^{M} \pi_m = 1 \\ & \sum_{k=1}^{K} \theta_{mk} = 1 \ (m = 1, .., M) \end{aligned}$$

The corresponding Lagrangian is:

$$L(\theta, \pi, \mu, \lambda_1, ...\lambda_M) = \sum_{m=1}^{M} \left[\eta_m \log \pi_m + \sum_{k=1}^{K} \xi_{mk} \log \theta_{mk} \right] + \mu \left(1 - \sum_{m=1}^{M} \pi_m - 1 \right) + \sum_{m=1}^{M} \lambda_m \left(\sum_{k=1}^{K} \theta_{mk} - 1 \right)$$

We set the gradient with respect to π and θ to 0:

$$\frac{\partial L(...)}{\partial \pi_m} = \frac{\eta_m}{\pi_m} - \mu = 0 \qquad m = 1, ..., M$$

And:

$$\frac{\partial L(...)}{\partial \theta_{mk}} = \frac{\xi_{mk}}{\theta_{mk}} - \lambda_m = 0$$

Using the optimization constraints and the conditions on η and ξ we find:

$$\mu = n, \ \lambda_m = \eta_m \text{ and } \pi_m^* = \frac{\eta_m}{n} \ \theta_{mk}^* = \frac{\xi_{mk}}{\eta_m}$$

Linear classification:

Generative model (LDA):

(a) Maximum likelihood:

Let us consider an i.i.d sample of observations $\mathcal{D} = \{(x_i, y_i)\}_{1\leq i\leq n}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$.

Suppose

$$y \sim Bernoulli(\alpha) \qquad x|\{y = i\} \sim \mathcal{N}(\mu_i, \Sigma)$$

$$\mathbb{P}(\mathcal{D}|\alpha, \mu_0, \mu_1, \Sigma) = \prod_{i=1}^{n} \alpha^{y_i}(1-\alpha)^{1-y_i} f_0(x_i)^{1-y_i} f_1(x_i)^{y_i}$$

Where

$$f_i(x) = \frac{1}{2\pi\sqrt{det\Sigma}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right)$$

Hence the log-likelihood is:

$$\begin{aligned} l(\alpha, \mu_0, \mu_1, \Sigma) = \sum_{i=1}^{n} \Big[& y_i \log\alpha + (1-y_i)\log(1-\alpha) \\ & + (1-y_i)\left(-log(2\pi) - \frac{1}{2}\log det\Sigma - \frac{1}{2}(x_i-\mu_0)^T\Sigma^{-1}(x_i-\mu_0)\right) \\ & + y_i\left(-log(2\pi) - \frac{1}{2}\log det\Sigma - \frac{1}{2}(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)\right)\Big] \end{aligned}$$

we set the gradients to 0 for α, μ_0 and μ_1:

$$\frac{\partial l}{\partial\alpha}(..) = \sum_{i=1}^{n}[\frac{y_i}{\alpha} - \frac{1-y_i}{1-\alpha}] = 0$$

thus: $\alpha^* = \frac{1}{n}\sum_{i=1}^{n} y_i$

$$\frac{\partial l}{\partial\mu_0}(..) = -\frac{1}{2}\Sigma^{-1}\left(\sum_{i=1}^{n}(1-y_i)(x_i-\mu_0)\right) = \mathbf{0}$$

thus $\mu_0^* = \frac{\sum_{i=1}^{n}(1-y_i)x_i}{\sum_{i=1}^{n}(1-y_i)}$

And similarly $\mu_1^* = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} y_i}$

Finally for Σ, we introduce $A = \Sigma^{-1}$

$$\begin{aligned} \frac{\partial l}{\partial A}(..) &= \frac{\partial}{\partial A}\Bigg(\sum_{i=1}^{n}\frac{1-y_i}{2}(\log detA - Tr((x_i-\mu_0)^T A(x_i-\mu_0))) \\ &\qquad + \frac{y_i}{2}(\log detA - Tr((x_i-\mu_1)^T A(x_i-\mu_1)))\Bigg) \\ &= \sum_{i=1}^{n}\frac{1-y_i}{2}(A^{-1} - (x_i-\mu_0)(x_i-\mu_0)^T) + \frac{y_i}{2}(A^{-1} - (x_i-\mu_1)(x_i-\mu_1)^T) \\ &= \frac{n}{2}A^{-1} - \frac{n}{2}\bar{\Sigma}_0 - \frac{n}{2}\bar{\Sigma}_1 \end{aligned}$$

Where

$$\bar{\Sigma}_0 = \frac{1}{n}\sum_{i=1}^{n}(1-y_i)(x_i-\mu_0)(x_i-\mu_0)^T$$

and:

$$\bar{\Sigma}_1 = \frac{1}{n}\sum_{i=1}^{n}y_i(x_i-\mu_1)(x_i-\mu_1)^T$$

Thus

$$\Sigma^* = \bar{\Sigma}_0 + \bar{\Sigma}_1$$

(b):

Using Bayes'rule:

$$\begin{aligned}\mathbb{P}(Y=1|X=x) &= \frac{\mathbb{P}(X=x|Y=1)\mathbb{P}(Y=1)}{\mathbb{P}(X=x|Y=1)\mathbb{P}(Y=1)+\mathbb{P}(X=x|Y=0)\mathbb{P}(Y=0)}\\ &= \frac{1}{1+\frac{(1-\alpha)f_0(x)}{\alpha f_1(x)}}\end{aligned}$$

Given that:

$$\begin{aligned}\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=0|X=x)} &= \frac{\alpha f_1(x)}{(1-\alpha)f_0(x)}\\ &= \exp\left((\Sigma^{-1}(\mu_1-\mu_0))^T x + \log\frac{\alpha}{1-\alpha} - \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0)\right)\\ &= \exp(\omega^T x + b)\end{aligned}$$

with:

$$\omega = \Sigma^{-1}(\mu_1-\mu_0), \qquad b = \log\frac{\alpha}{1-\alpha} - \frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)$$

We end up with an expression similar to the logistic regression with an explicit intercept b.

$$\mathbb{P}(Y=1|X=x) = \frac{1}{1+\exp(-(\omega^T x+b))} = \sigma(\omega^T x + b)$$

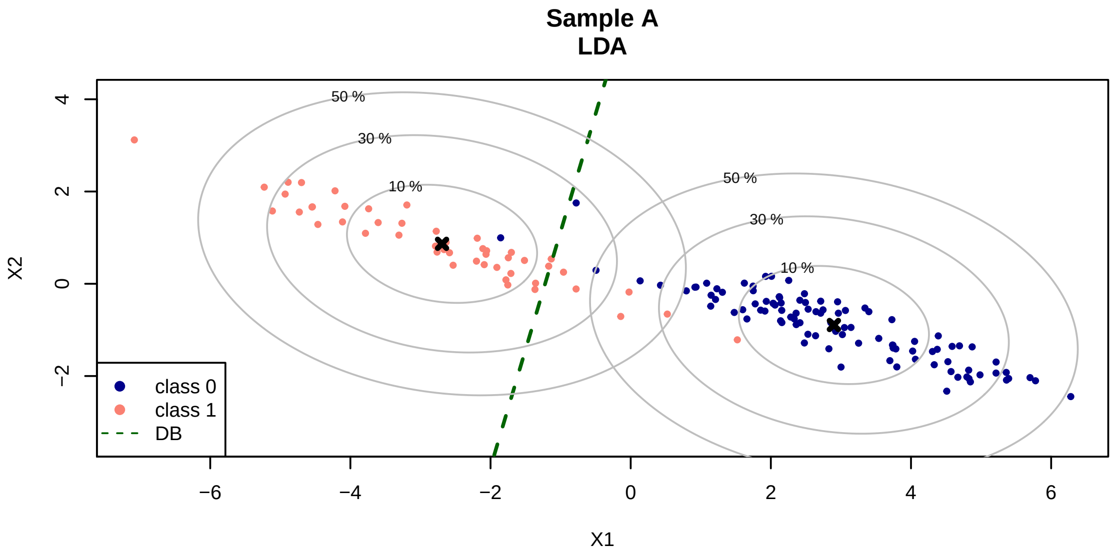
Our decision boundary would be $\mathbb{P}(Y=1|x) = .5$ which is equivalent to $w^T x + b = 0$

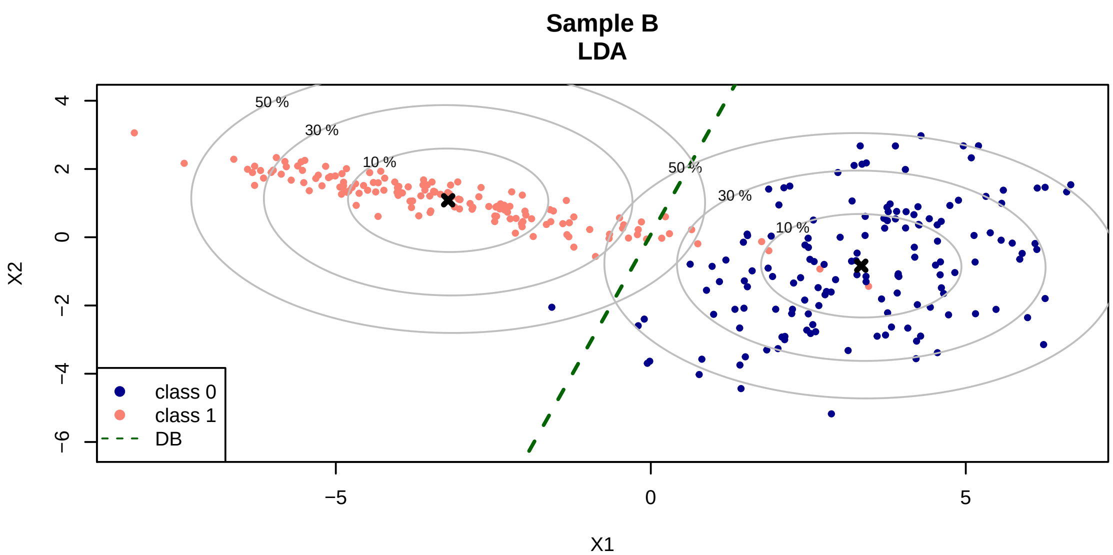
(c):

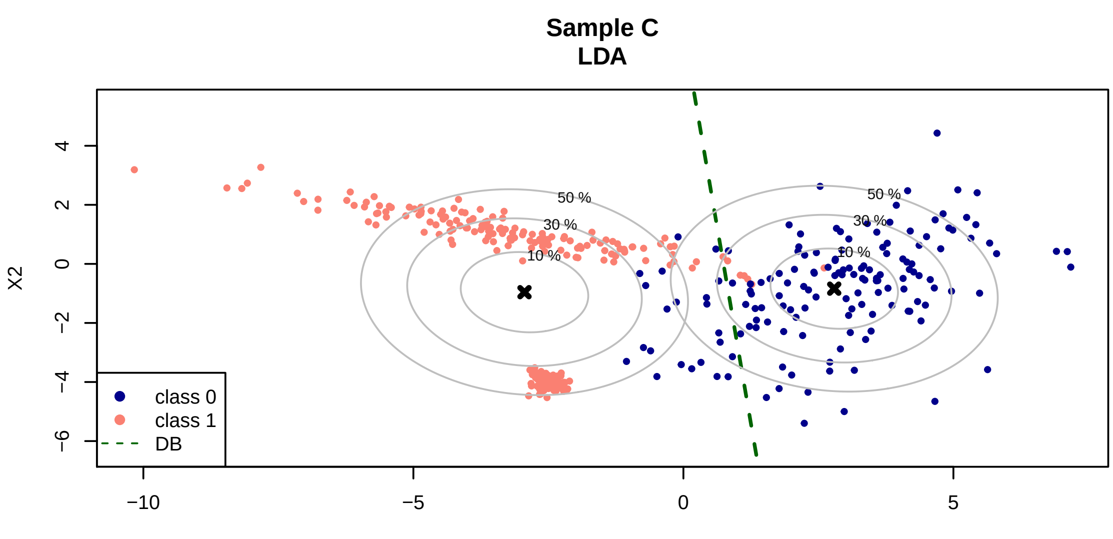
We implement the model above on a sample (X,Y):

```
MLE.LDA<-function(X,Y){
  alpha=mean(Y)
  mu1=colSums(X[Y==1,])/sum(Y)
  mu0=colSums(X[Y==0,])/sum(1-Y)
  sigma.0=(t(X[Y==0,]-mu0)%*%(X[Y==0,]-mu0))/length(Y)
  sigma.1=(t(X[Y==1,]-mu1)%*%(X[Y==1,]-mu1))/length(Y)
  sigma=sigma.0+sigma.1
  A=solve(sigma)
  w=A%*%(mu1-mu0)
  b=log(alpha/(1-alpha))+1/2*t(mu0+mu1)%*%A%*%(mu1-mu0)
  predict<-function(x){(t(w)%*%x+b>=0)+0}
  list(alpha=alpha,mu1=mu1,mu0=mu0,sigma=sigma,w=w,b=b,predict=predict)
}
```

Estimated gaussian distributions and decision boundaries on the training sets A,B and C:







The estimated parameters:

Table 1: LDA - sample A

alpha	mu1	mu0	sigma.1	sigma.2	w	b
0.33	-2.69	2.9	8.74	-1.31	-0.62	-0.76
0.33	0.87	-0.89	-1.31	7.78	0.12	-0.76

Table 2: LDA - sample B

alpha	mu1	mu0	sigma.1	sigma.2	w	b
0.5	-3.22	3.34	12	-0.21	-0.54	-0.01
0.5	1.08	-0.84	-0.21	10.91	0.17	-0.01

Table 3: LDA - sample C

alpha	mu1	mu0	sigma.1	sigma.2	w	b
0.62	-2.94	2.79	6.61	-0.69	-0.88	0.65
0.62	-0.96	-0.84	-0.69	8.77	-0.08	0.65

Logistic regression:

We will implement the logistic regression model with the assumption:

$$\log \frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=0|X=x)} = f(x) = w^T x$$

Where the intercept w_0 is included in w by adding adding a column of ones to the design matrix X.

We have established in the course that:

$$l(w) = \sum_{i=1}^{n} y_i w^T x_i + \log \sigma(-w^T x_i)$$

Thus:

$$\nabla_w l(w) = \sum_{i=1}^{n} (y_i - \eta_i) x_i$$

with $\eta_i = \sigma(w^T x_i)$

$$Hl(w) = -X^T D_\eta X$$

with $D_\eta = Diag(\eta_i(1-\eta_i))$.

The Newton-Raphson method consists of updating w as follows:

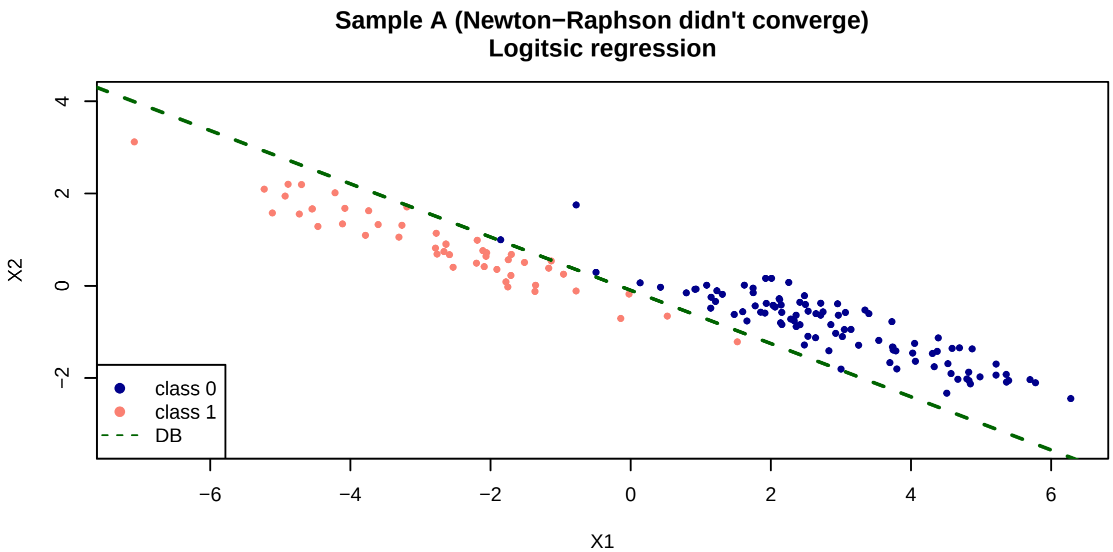
$$w^{(new)} = w^{(old)} + \left(X^T D_{\eta^{(old)}} X\right)^{-1} X^T (Y - \eta^{(old)})$$

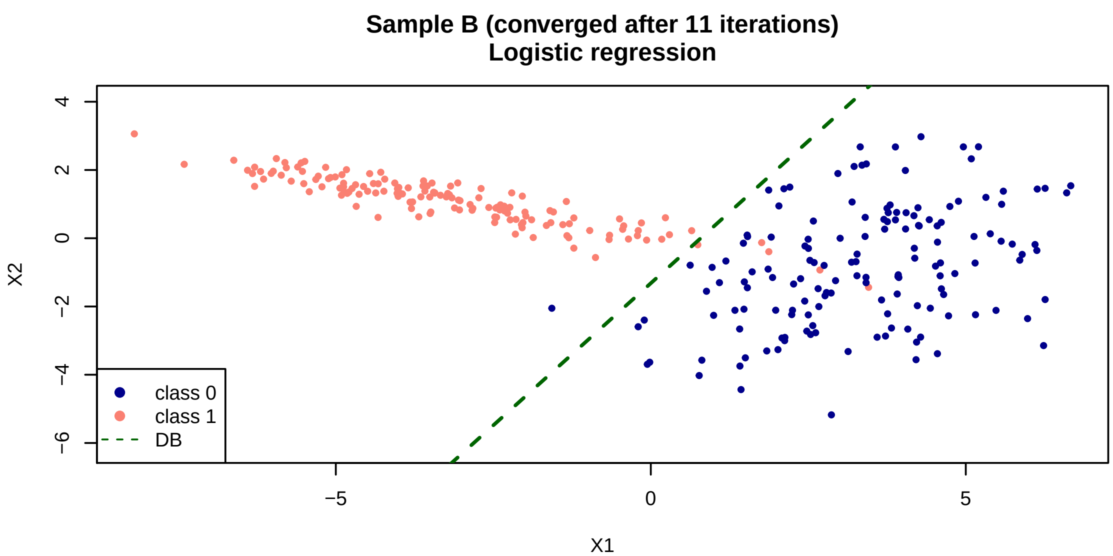
assuming $X^T D_\eta X$ is invertible.

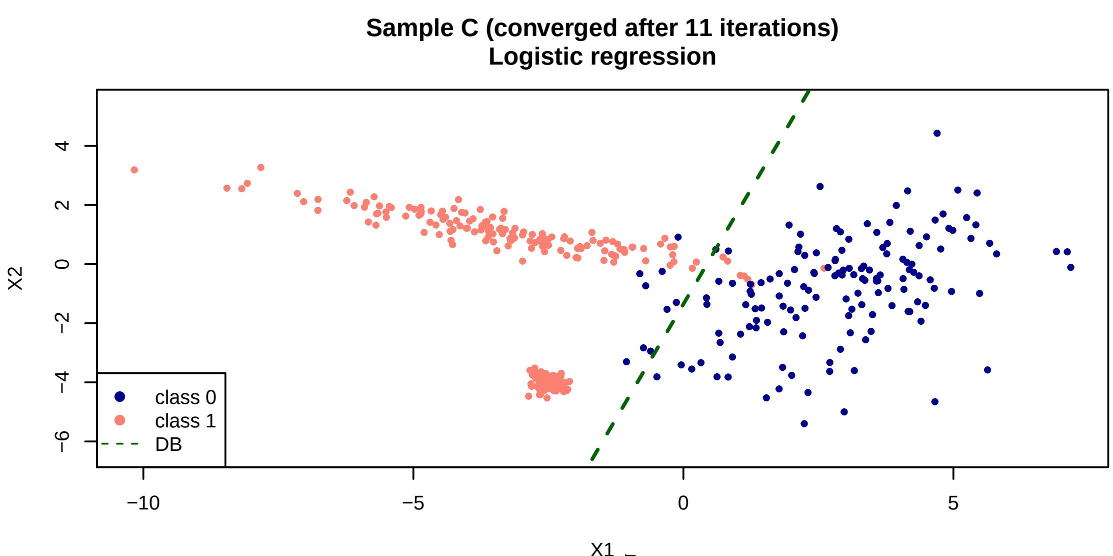
The descision boundary defined as $\mathbb{P}(Y=1|x) = .5$ is equivalent to $w^T x = 0$

```r
sigmoid<- function(x){1/(1+exp(-x))}
p<-function(x,w){sigmoid(apply(x,1,function(u){t(u)%*%w}))}
grad = function(x,y,w){-t(x)%*%(y-p(x,w))}
hess = function(x,y,w){
  t(x)%*% diag(p(x,w)*(1-p(x,w))) %*%x
}

NR.Logit<-function(X,Y,max_iter=1000,tol=1e-7){
  X=cbind(1,X)
  #Initialize w:
  #w=rnorm(dim(X)[2])
  w=numeric(ncol(X))
  flag=T
  iter=1
  while(iter<max_iter & flag){
    w.old=w
    w=w.old-ginv(hess(X,Y,w.old))%*%grad(X,Y,w.old)
    norm(w-w.old,'2')
    flag =norm(w-w.old,'2') > tol
    iter=iter+1
  }
  predict<-function(x){(t(w)%*%c(1,x)>=0)+0}
  list(w=w,iter=iter,cvg=iter<max_iter,predict=predict)
}
```

Estimated decision boundaries on the training sets A,B and C:







The estimated parameters:

Table 4: Logit - sample A

w	iter	cvg
-200.6	1000	FALSE
-1179	1000	FALSE
-2043	1000	FALSE

Table 5: Logit - sample B

w	iter	cvg
1.35	11	TRUE
-1.71	11	TRUE
1.02	11	TRUE

Table 6: Logit - sample C

w	iter	cvg
0.96	11	TRUE
-2.2	11	TRUE
0.71	11	TRUE

Linear regression:

For the affine model $y = w^T x + b$ or $y = w^T x$ after offset reparametrization.

The solution to the normal equations is:

$$w^* = (X^T X)^{-1} X^T Y$$

and the noise variance:

$$\hat{\sigma}^2 = \frac{1}{n}(Y - Xw^*)^T (Y - Xw^*)$$

The decision boundary $\mathbb{P}(Y = 1|x) = .5$ is equivalent to:

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(1 - w^T x)^2}{\sigma^2}\right) = .5$$

$$w^T x + \sqrt{\sigma^2 \log(\frac{2}{\pi\sigma^2})} - 1 = 0$$

```
LR<- function(X,Y){
  X=cbind(1,X)
  w=ginv(t(X)%*%X)%*%t(X)%*%Y
  sigma=mean((Y-X%*%w)^2)
  #the extra intercept:
  alpha=sqrt(sigma*log(2/pi/sigma))-1
  predict<-function(x){(t(w)%*%c(1,x)+alpha>=0)+0}
  list(w=w,sigma=sqrt(sigma),alpha=alpha,predict=predict)
}
```

The estimated parameters:

Table 7: LR - sample A

w	sigma	alpha
0.49	0.2	-0.67
-0.26	0.2	-0.67
-0.37	0.2	-0.67

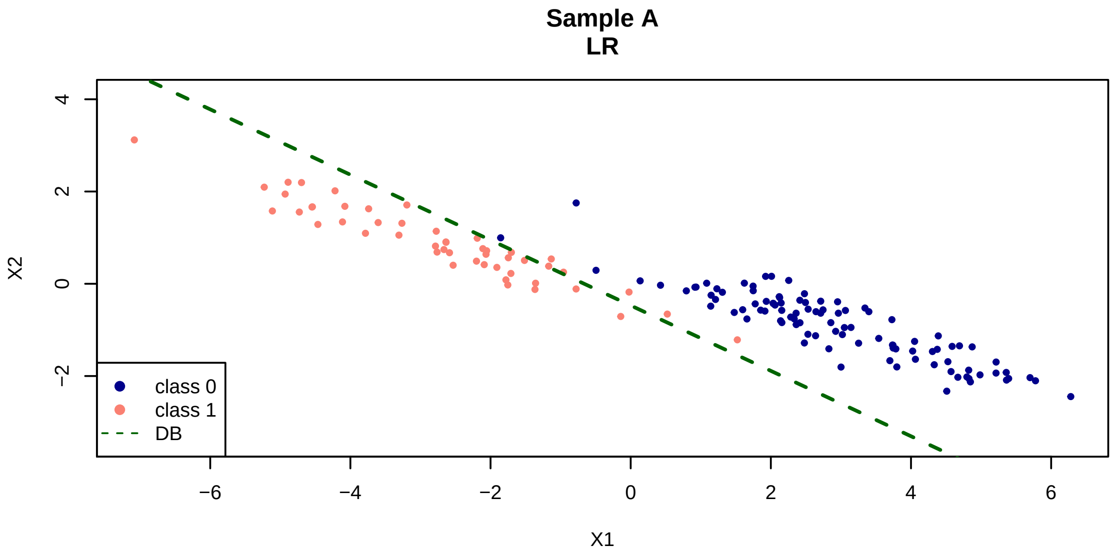
Table 8: LR - sample B

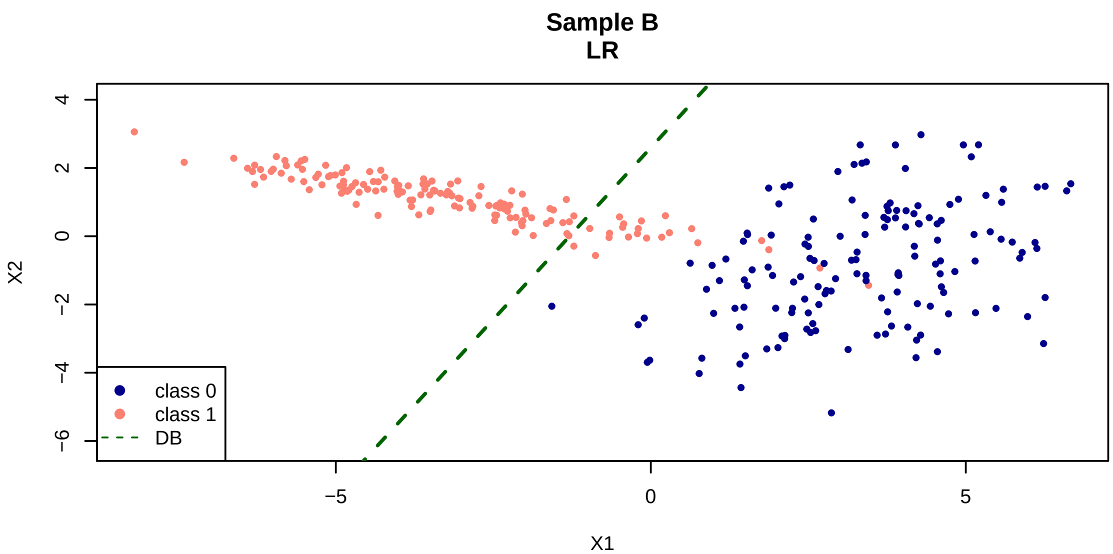
w	sigma	alpha
0.5	0.23	-0.63
-0.1	0.23	-0.63
0.05	0.23	-0.63

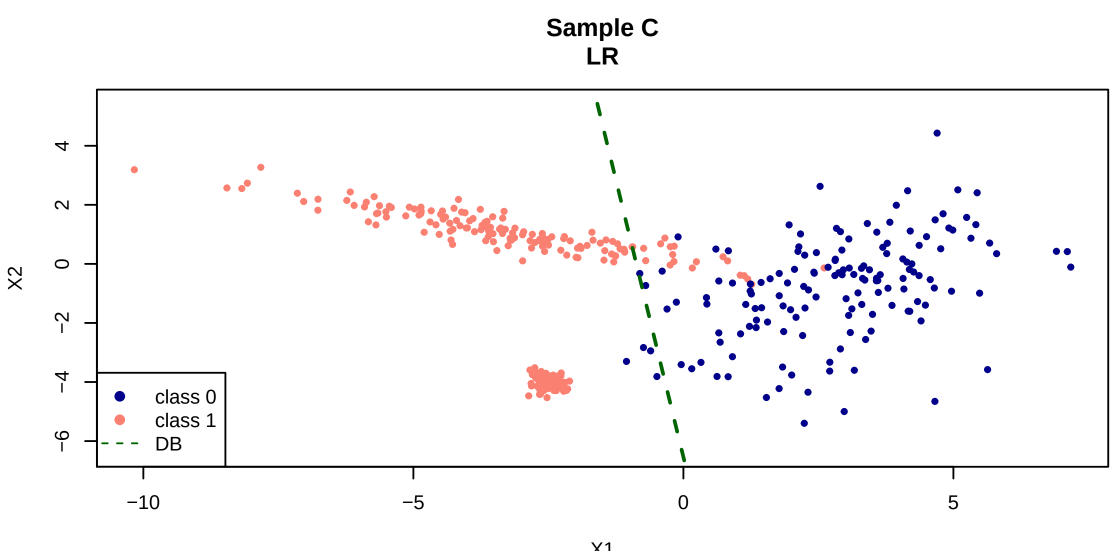
Table 9: LR - sample C

w	sigma	alpha
0.51	0.25	-0.62
-0.13	0.25	-0.62
-0.02	0.25	-0.62

Estimated decision boundaries on the training sets A,B and C:







Models' performance:

```
#LDA - A
##Train:
LDA.A.Train=apply(X.A,1,LDA.A$predict)
Confusion.LDA.A.Train=CM(LDA.A.Train,Y.A)
##Test:
LDA.A.Test=apply(X.AT,1,LDA.A$predict)
Confusion.LDA.A.Test=CM(LDA.A.Test,Y.AT)

#Confusion matrices:
Confusion.LDA.A.Train$C
```

```
##       true  0  1
## pred
## 0          99  6
## 1           1 44
```

```
Confusion.LDA.A.Test$C
```

```
##       true   0   1
## pred
## 0          998  60
## 1            2 440
```

```
#LDA - B
##Train:
LDA.B.Train=apply(X.B,1,LDA.B$predict)
Confusion.LDA.B.Train=CM(LDA.B.Train,Y.B)
##Test:
LDA.B.Test=apply(X.BT,1,LDA.B$predict)
Confusion.LDA.B.Test=CM(LDA.B.Test,Y.BT)

#Confusion matrices:
Confusion.LDA.B.Train$C
```

```
##       true   0   1
## pred
## 0          149   9
## 1            1 141
```

```
Confusion.LDA.B.Test$C
```

```
##       true   0   1
## pred
## 0          972  69
## 1           28 931
```

```
#LDA - C
##Train:
LDA.C.Train=apply(X.C,1,LDA.C$predict)
Confusion.LDA.C.Train=CM(LDA.C.Train,Y.C)
##Test:
LDA.C.Test=apply(X.CT,1,LDA.C$predict)
Confusion.LDA.C.Test=CM(LDA.C.Test,Y.CT)

#Confusion matrices:
Confusion.LDA.C.Train$C
```

```
##       true   0   1
## pred
## 0          128   7
## 1           22 243
```

```
Confusion.LDA.C.Test$C
```

```
##       true    0    1
## pred
## 0           867   28
## 1           133 1972
```

```
#Logit - A
##Train:
Logit.A.Train=apply(X.A,1,Logit.A$predict)
Confusion.Logit.A.Train=CM(Logit.A.Train,Y.A)
##Test:
Logit.A.Test=apply(X.AT,1,Logit.A$predict)
Confusion.Logit.A.Test=CM(Logit.A.Test,Y.AT)

#Confusion matrices:
Confusion.Logit.A.Train$C
```

```
##       true   0   1
## pred
## 0          100   0
## 1            0  50
```

```
Confusion.Logit.A.Test$C
```

```
##       true   0   1
## pred
## 0          982  35
## 1           18 465
```

```
#Logit - B
##Train:
Logit.B.Train=apply(X.B,1,Logit.B$predict)
Confusion.Logit.B.Train=CM(Logit.B.Train,Y.B)
##Test:
Logit.B.Test=apply(X.BT,1,Logit.B$predict)
Confusion.Logit.B.Test=CM(Logit.B.Test,Y.BT)

#Confusion matrices:
Confusion.Logit.B.Train$C
```

```
##       true   0   1
## pred
## 0          149   5
## 1            1 145
```

```
Confusion.Logit.B.Test$C
```

```
##       true   0   1
## pred
## 0          946  32
## 1           54 968
```

```
#Logit - C
##Train:
Logit.C.Train=apply(X.C,1,Logit.C$predict)
Confusion.Logit.C.Train=CM(Logit.C.Train,Y.C)
##Test:
Logit.C.Test=apply(X.CT,1,Logit.C$predict)
Confusion.Logit.C.Test=CM(Logit.C.Test,Y.CT)

#Confusion matrices:
Confusion.Logit.C.Train$C
```

```
##       true   0   1
## pred
## 0          141   7
## 1            9 243
```

```
Confusion.Logit.C.Test$C
```

```
##       true    0    1
## pred
## 0           973   41
## 1            27 1959
```

```
#LR - A
##Train:
LR.A.Train=apply(X.A,1,LR.A$predict)
Confusion.LR.A.Train=CM(LR.A.Train,Y.A)
##Test:
LR.A.Test=apply(X.AT,1,LR.A$predict)
Confusion.LR.A.Test=CM(LR.A.Test,Y.AT)

#Confusion matrices:
Confusion.LR.A.Train$C
```

```
##       true   0   1
## pred
## 0          100   6
## 1            0  44
```

```
Confusion.LR.A.Test$C
```

```
##       true   0   1
## pred
## 0          996  63
## 1            4 437
```

```
#LR - B
##Train:
LR.B.Train=apply(X.B,1,LR.B$predict)
Confusion.LR.B.Train=CM(LR.B.Train,Y.B)
##Test:
LR.B.Test=apply(X.BT,1,LR.B$predict)
Confusion.LR.B.Test=CM(LR.B.Test,Y.BT)

#Confusion matrices:
Confusion.LR.B.Train$C
```

```
##       true   0   1
## pred
## 0          150  22
## 1            0 128
```

```
Confusion.LR.B.Test$C
```

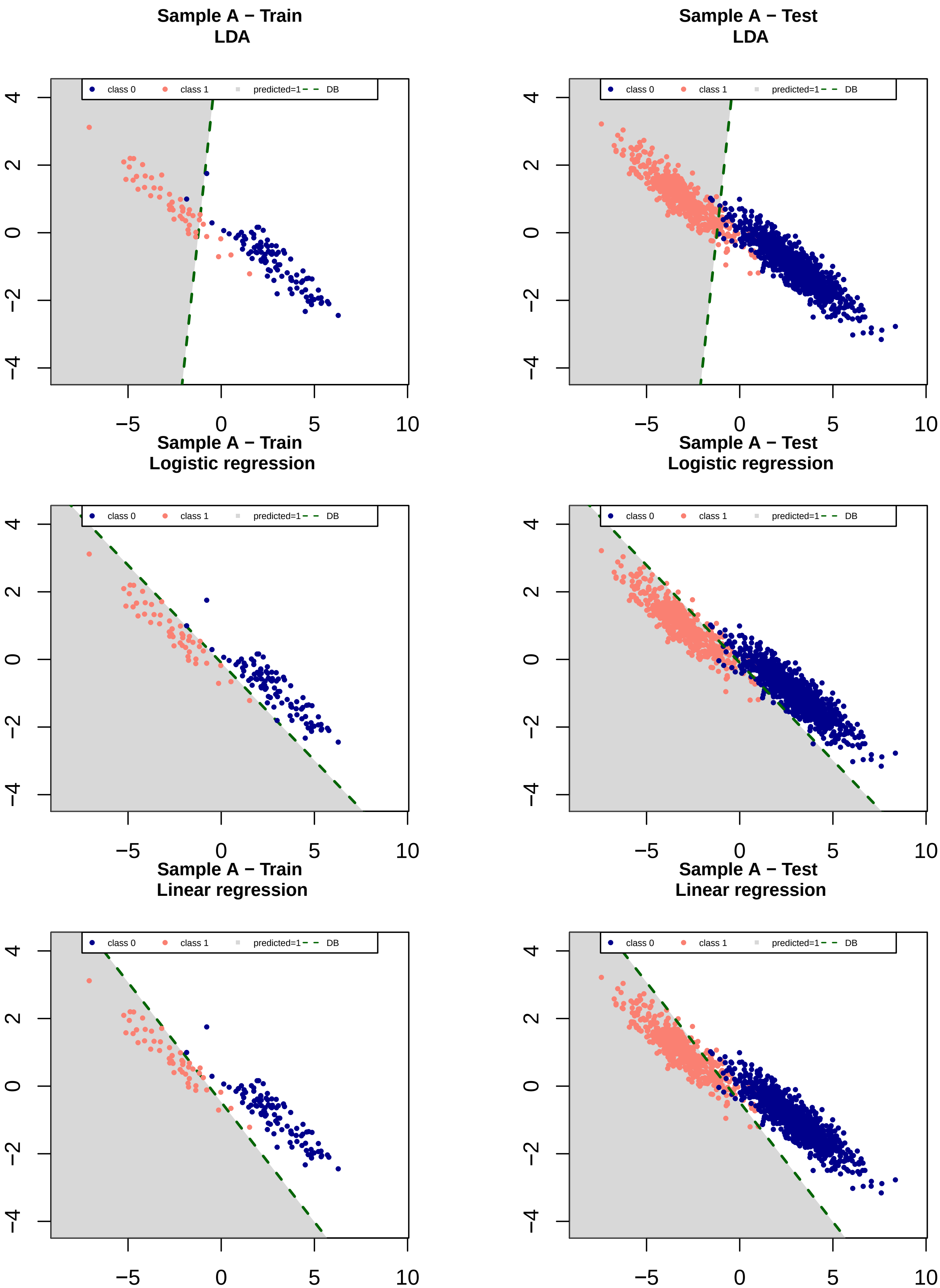
```
##       true   0   1
## pred
## 0          997 174
## 1            3 826
```

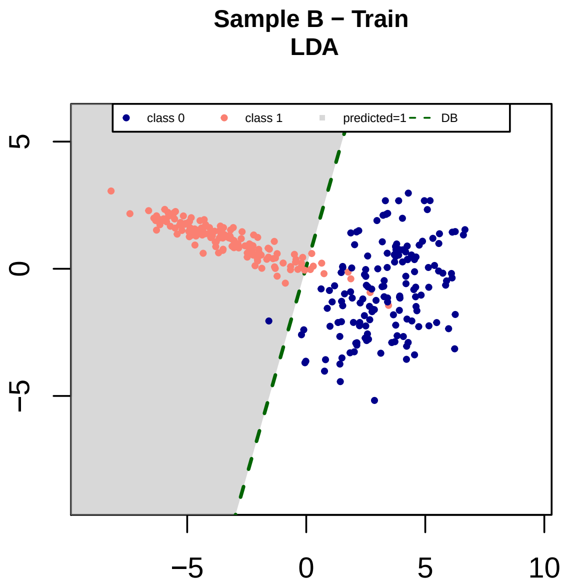
```
#LR - C
##Train:
LR.C.Train=apply(X.C,1,LR.C$predict)
Confusion.LR.C.Train=CM(LR.C.Train,Y.C)
##Test:
LR.C.Test=apply(X.CT,1,LR.C$predict)
Confusion.LR.C.Test=CM(LR.C.Test,Y.CT)

#Confusion matrices:
Confusion.LR.C.Train$C
```

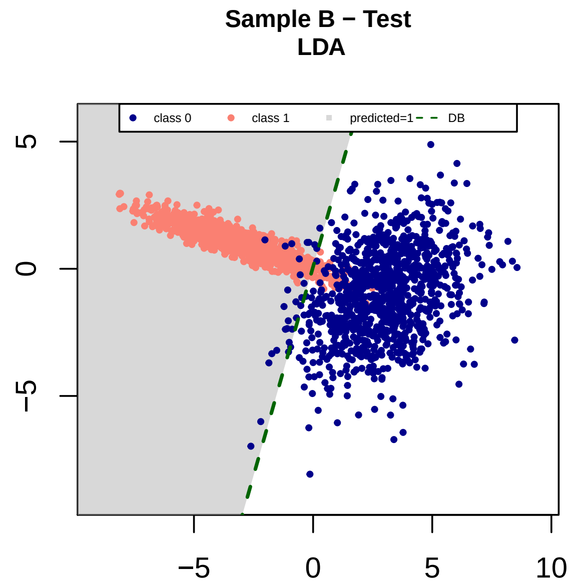
```
##       true   0   1
## pred
## 0          146  19
## 1            4 231
```

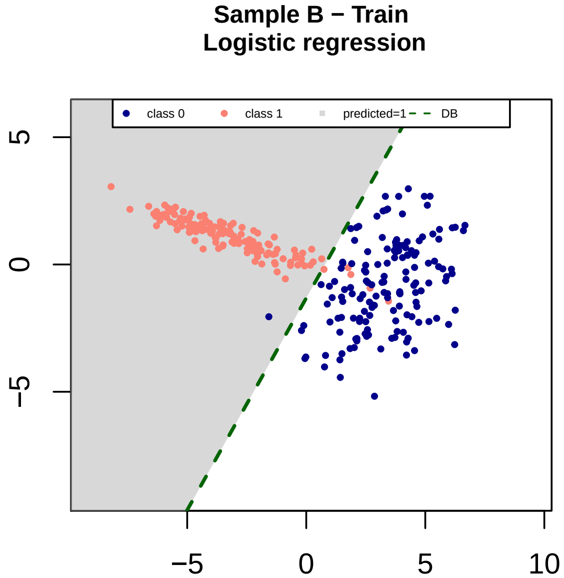
```
Confusion.LR.C.Test$C
```

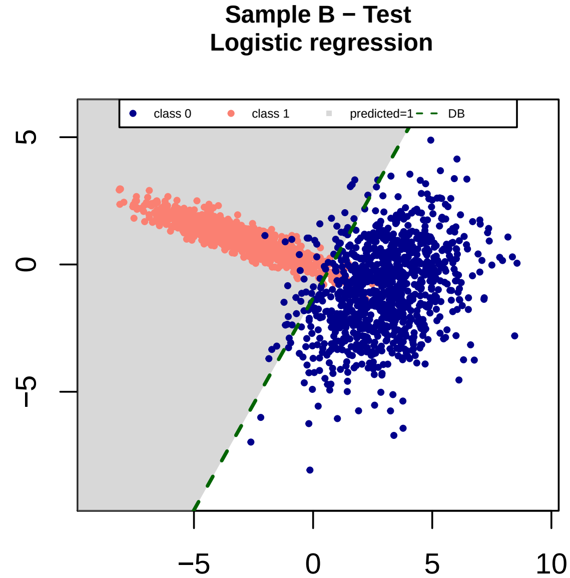
```
##       true    0    1
## pred
## 0           983  141
## 1            17 1859
```

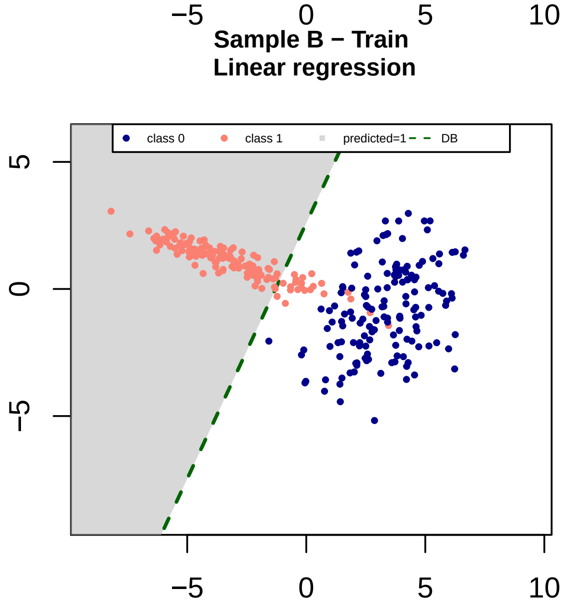

Sample A − Train
LDA
Sample A − Test
LDA
Sample A − Train
Logistic regression
Sample A − Test
Logistic regression
Sample A − Train
Linear regression
Sample A − Test
Linear regression
class 0
class 1
predicted=1
DB

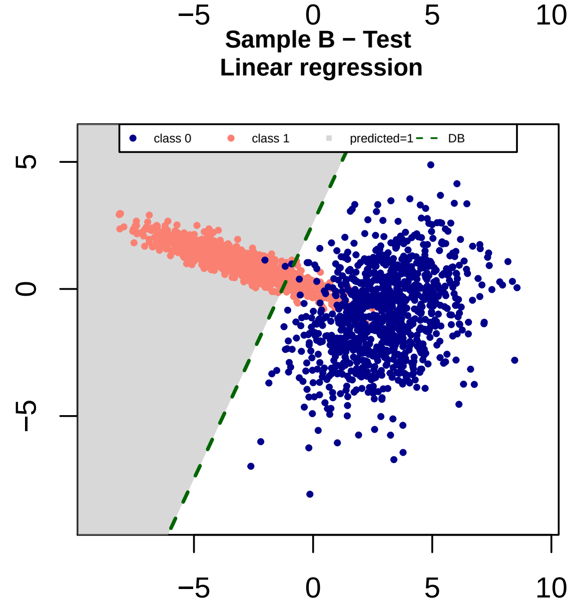

Sample B − Train
LDA
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10


Sample B − Test
LDA
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10


Sample B − Train
Logistic regression
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10


Sample B − Test
Logistic regression
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10


Sample B − Train
Linear regression
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10


Sample B − Test
Linear regression
class 0
class 1
predicted=1
DB
5
0
−5
−5
0
5
10

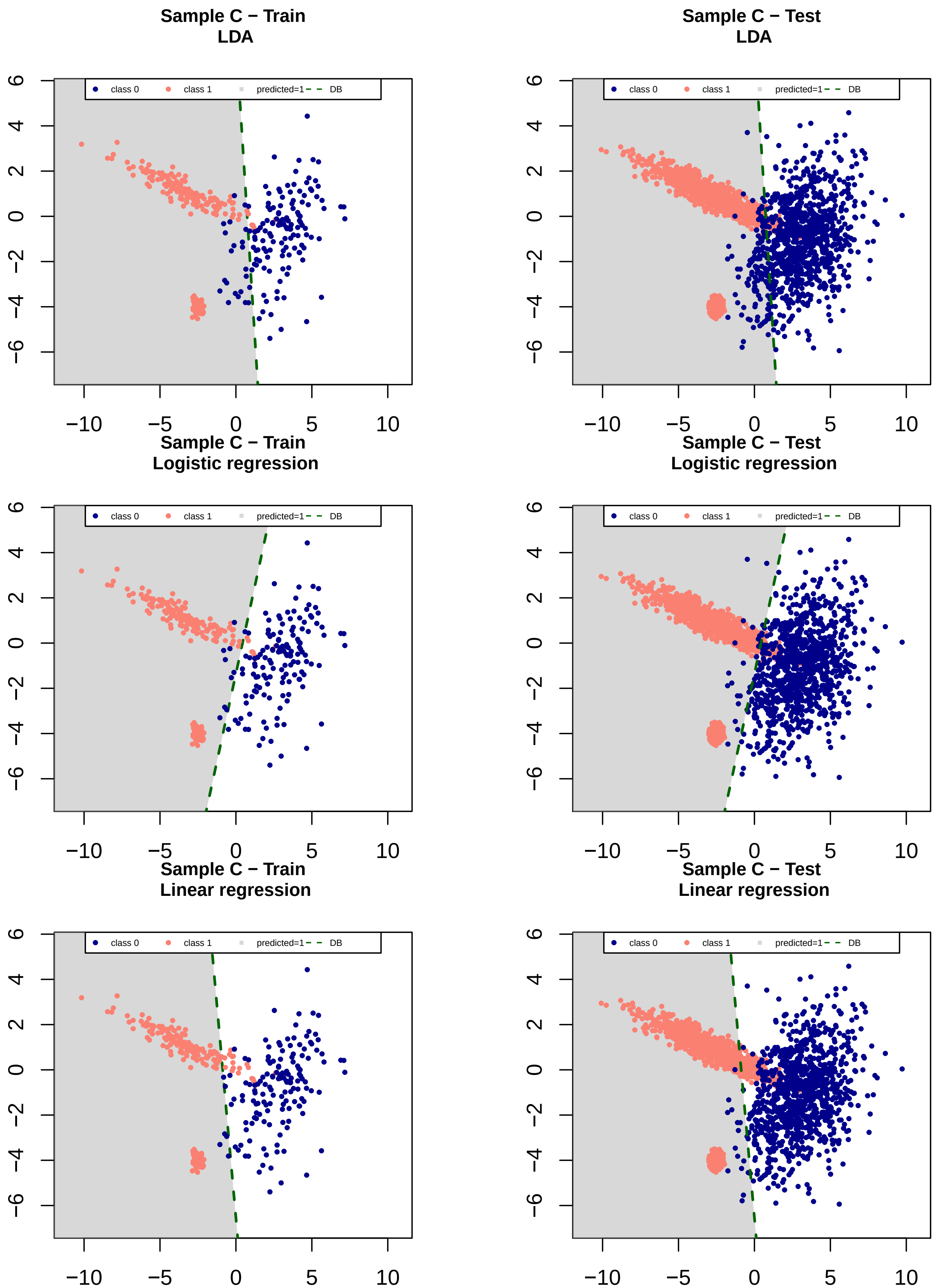

Sample C − Train
LDA
Sample C − Test
LDA
Sample C − Train
Logistic regression
Sample C − Test
Logistic regression
Sample C − Train
Linear regression
Sample C − Test
Linear regression
class 0
class 1
predicted=1
DB

Table 10: Comparison of the 3 models on 3 samples - error in %

Model	A.train	A.test	B.train	B.test	C.train	C.test
LDA	4.667	4.133	3.333	4.85	7.25	5.367
Logit	0	3.533	2	4.3	4	2.267
LR	4	4.467	7.333	8.85	5.75	5.267

We can see that the logistic regression outperforms LDA and linear regression on the 3 datasets even on A where the training set is linearly separable, thus allowing the algorithm to diverge and the computed weights to be large and prone to overfit the training set. The LDA do well on the sets A and B where the model of gaussian distribution is quite accurate; although the mislclassification measure favors the logitic regression, the decision boundaries of LDA on A and B seems more logic - the generative classification is suitable here - while the existence of two separate clusters on a single class in the set C affects the results of LDA badly.

QDA model:

When relaxing the assumption that the two classes share the same covariance matrix, the maximum likelihood yields the same results excpet from:

$$\Sigma_0^* = \frac{n}{\sum_{i=1}^n (1-y_i)} \bar{\Sigma}_0$$

$$\Sigma_1^* = \frac{n}{\sum_{i=1}^n y_i} \bar{\Sigma}_1$$

As for the decision boundary, following the same step we get a boundary of quadratic equation:

$$x^T Q x + w^T x + b = 0$$

with:

$$Q = \frac{1}{2}(\Sigma_O^{-1} + \Sigma_1^{-1})$$

$$w = \Sigma_1^{-1}\mu_1 - \Sigma_0^{-1}\mu_0$$

$$b = \log\left(\frac{\alpha\sqrt{det\Sigma_0}}{(1-\alpha)\sqrt{det\Sigma_1}}\right) - \frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma_0^{-1}\mu_0$$

We implement the QDA in the following function:

```
MLE.QDA<-function(X,Y){
  alpha=mean(Y)
  mu1=colSums(X[Y==1,])/sum(Y)
  mu0=colSums(X[Y==0,])/sum(1-Y)
  emp.sigma.0=(t(X[Y==0,]-mu0)%*%(X[Y==0,]-mu0))/length(Y)
  emp.sigma.1=(t(X[Y==1,]-mu1)%*%(X[Y==1,]-mu1))/length(Y)
  sigma.0=length(Y)/sum(1-Y)*emp.sigma.0
  sigma.1=length(Y)/sum(Y)*emp.sigma.1
  A=solve(sigma.0)
  B=solve(sigma.1)
  Q=(A-B)/2
  w=B%*%mu1-A%*%mu0
  b=log(alpha*sqrt(det(sigma.0))/(1-alpha)/sqrt(det(sigma.1)))
  -t(mu1)%*%B%*%mu1/2+t(mu0)%*%B%*%mu0/2
  Conic=cbind(rbind(Q,t(w)/2),c(w/2,b))
  predict<-function(x){((t(c(x,1))%*%Conic%*%c(x,1))>=0)+0}
```

```
  list(alpha=alpha,mu1=mu1,mu0=mu0,sigma.0=sigma.0,
       sigma.1=sigma.1,Q=Conic,predict=predict)
}
```

The estimated parameters:

Table 11: QDA - sample A

alpha	mu1	mu0	sigma.0.1	sigma.0.2	sigma.1.1	sigma.1.2
0.33	-2.69	2.9	8.84	-1.21	8.53	-1.5
0.33	0.87	-0.89	-1.21	8.12	-1.5	7.11

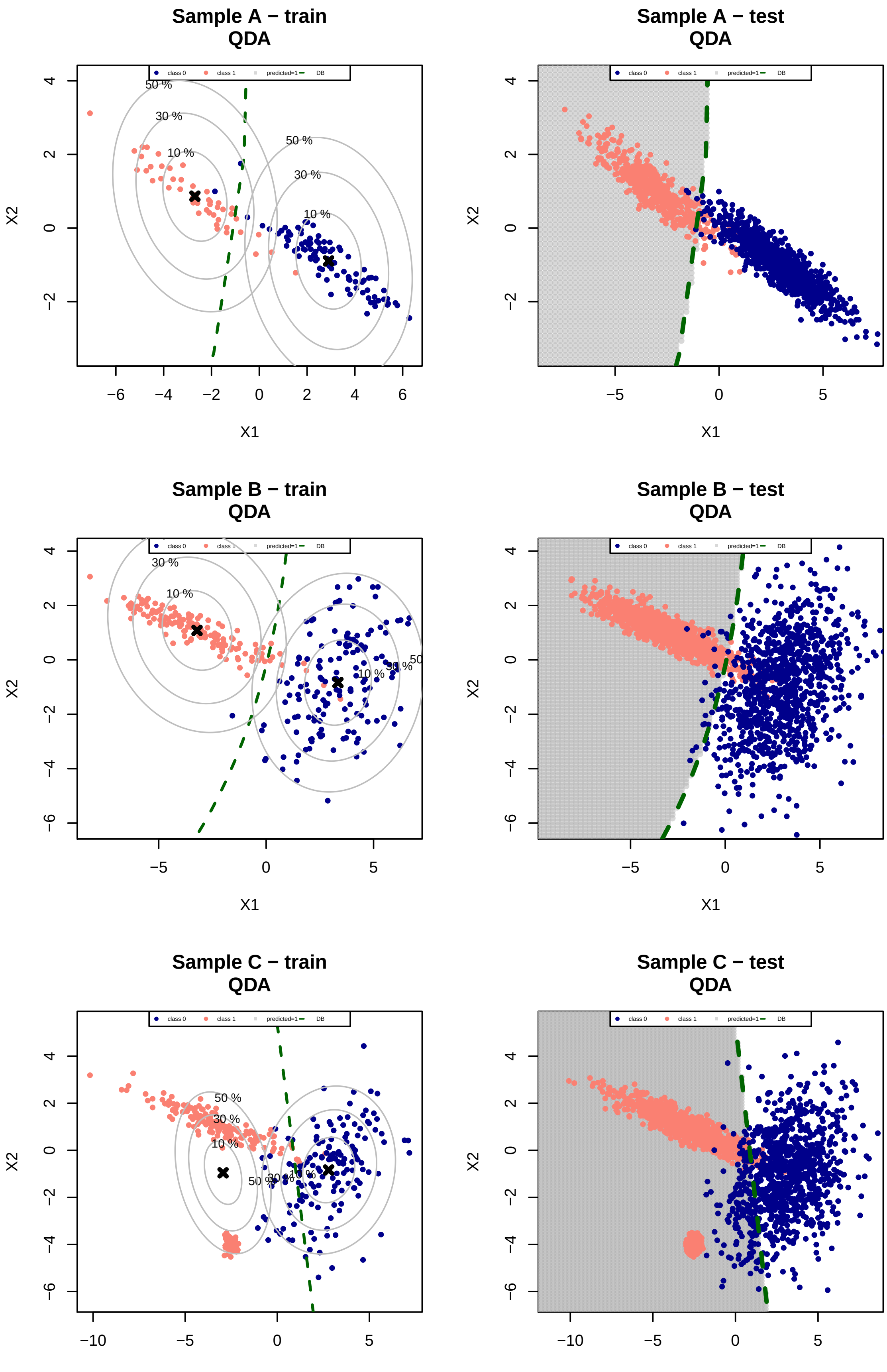
Table 12: QDA - sample B

alpha	mu1	mu0	sigma.0.1	sigma.0.2	sigma.1.1	sigma.1.2
0.5	-3.22	3.34	11.55	1.2	12.45	-1.61
0.5	1.08	-0.84	1.2	11.65	-1.61	10.16

Table 13: QDA - sample C

alpha	mu1	mu0	sigma.0.1	sigma.0.2	sigma.1.1	sigma.1.2
0.62	-2.94	2.79	9.49	1.07	4.89	-1.74
0.62	-0.96	-0.84	1.07	9.18	-1.74	8.52

Estimated gaussian distributions and decision boundaries on the training sets A,B and C:



Performance of QDA:

```
#QDA - A
##Train:
QDA.A.Train=apply(X.A,1,QDA.A$predict)
Confusion.QDA.A.Train=CM(QDA.A.Train,Y.A)
##Test:
QDA.A.Test=apply(X.AT,1,QDA.A$predict)
Confusion.QDA.A.Test=CM(QDA.A.Test,Y.AT)
#Confusion matrices:
Confusion.QDA.A.Train$C
```

```
##      true  0  1
## pred
## 0         98  5
## 1          2 45
```

```
Confusion.QDA.A.Test$C
```

```
##      true   0   1
## pred
## 0         996  41
## 1           4 459
```

```
#QDA - B
##Train:
QDA.B.Train=apply(X.B,1,QDA.B$predict)
Confusion.QDA.B.Train=CM(QDA.B.Train,Y.B)
##Test:
QDA.B.Test=apply(X.BT,1,QDA.B$predict)
Confusion.QDA.B.Test=CM(QDA.B.Test,Y.BT)
#Confusion matrices:
Confusion.QDA.B.Train$C
```

```
##      true   0   1
## pred
## 0         149   8
## 1           1 142
```

```
Confusion.QDA.B.Test$C
```

```
##      true   0   1
## pred
## 0         976  64
## 1          24 936
```

```
#QDA - C
##Train:
QDA.C.Train=apply(X.C,1,QDA.C$predict)
Confusion.QDA.C.Train=CM(QDA.C.Train,Y.C)
##Test:
QDA.C.Test=apply(X.CT,1,QDA.C$predict)
```

```
Confusion.QDA.C.Test=CM(QDA.C.Test,Y.CT)
#Confusion matrices:
Confusion.QDA.C.Train$C
```

```
##      true   0   1
## pred
## 0         125   5
## 1          25 245
```

```
Confusion.QDA.C.Test$C
```

```
##      true    0    1
## pred
## 0          836   26
## 1          164 1974
```

Table 14: Comparison of the 4 models on 3 samples - error in %

Model	A.train	A.test	B.train	B.test	C.train	C.test
LDA	4.667	4.133	3.333	4.85	7.25	5.367
Logit	0	3.533	2	4.3	4	2.267
LR	4	4.467	7.333	8.85	5.75	5.267
QDA	4.667	3	3	4.4	7.5	6.333

As we expected, the QDA did well on sample B where the assumption of gaussian distributions with different covariances is suitable. Its performance on A is quite similar to this of LDA, but on sample C, QDA seems more sensitive to outliers, thus performing poorly.