Graphical models - MVA 2017/2018
Homework 1

Rémi Lespinet

1 Learning in discrete graphical models

Let $N \in \mathbb{N}$ and $\left(y^{(1)}, \dots, y^{(N)}\right)$ a sample of observations. $\forall i \in \{1 \dots N\}, y^{(i)} = (x^{(i)}, y^{(i)})$

$$\mathcal{L}(\theta, \pi) = p(y^{(1)}, \dots, y^{(N)}; \theta, \pi)$$

By independance of the observations, we can write

$$\begin{aligned}
\mathcal{L}(\theta, \pi) &= \prod_{i=1}^{N} p(y^{(i)}; \theta, \pi) \\
&= \prod_{i=1}^{N} p(x^{(i)} | z^{(i)}; \theta, \pi) p(z^{(i)}; \theta, \pi) \\
&= \prod_{i=1}^{N} \left(\prod_{m=1}^{M} \prod_{k=1}^{K} \theta_{mk}^{\hat{x}_{mk}^{(i)}} \right) \left(\prod_{m=1}^{M} \pi_m^{\hat{z}_m^{(i)}} \right) \\
&= \prod_{i=1}^{N} \left(\prod_{m=1}^{M} \left(\pi_m^{\hat{z}_m^{(i)}} \prod_{k=1}^{K} \theta_{mk}^{\hat{x}_{mk}^{(i)}} \right) \right)
\end{aligned}$$

where

$$\forall i \in \{1 \dots N\}, \forall m \in \{1 \dots M\}, \hat{z}_m^{(i)} = \begin{cases} 1 \text{ if } z^{(i)} = m \\ 0 \text{ otherwise} \end{cases} \quad \text{(One hot encoding)}$$

and

$$\forall i \in \{1 \dots N\}, \forall m \in \{1 \dots M\}, \forall k \in \{1 \dots K\}, \hat{x}_{km}^{(i)} = \begin{cases} 1 \text{ if } z^{(i)} = m \wedge x^{(i)} = k \\ 0 \text{ otherwise} \end{cases}$$

If we write the log likelihood,

$$\ln \mathcal{L}(\theta, \pi) = \sum_{i=1}^{N} \left(\sum_{m=1}^{M} \left(\hat{z}_m^{(i)} \ln(\pi_m) + \sum_{k=1}^{K} \hat{x}_{mk}^{(i)} \ln(\theta_{mk}) \right) \right)$$

This is a maximisation problem under the contraints

$$\begin{cases} \sum_{m=1}^{M} \pi_m = 1 \\ \forall m \in \{1 \dots M\} \sum_{k=1}^{K} \theta_{mk} = 1 \end{cases}$$

We don't need to add the constraints $\pi_i > 0$ and $\theta_{mk} > 0$ because the expression of the log likelihood tends to $-\infty$ when one of the π_i or θ_{mk} tends to 0.

Writing the lagrangian, we get

$$L(\pi, \theta, \lambda, \mu) = -\ln \mathcal{L}(\theta, \pi) + \lambda \left(\sum_{m=1}^{M} \pi_m - 1 \right) + \sum_{m=1}^{M} \mu_m \left(\sum_{k=1}^{K} \theta_{mk} - 1 \right)$$

Since the negative log likelihood is strictly convex with respect to π, θ, the lagrangian is also strictly convex with respect to these variables and we know that there exist a minimum and that it's unique.

Moreover there exists $(\pi_1, \dots \pi_M) \in [0; 1]^M$ such that

$$\sum_{i=1}^{M} \pi_i = 1$$

and $\theta \in \mathcal{M}_{M \times K}(\mathbb{R})$ such that

$$\forall m \in \{1 \dots M\}, \sum_{k=1}^{K} \theta_{mk} = 1$$

By Slater's constraint we have strong duality.

Let $j \in \{1 \dots M\}$ and $l \in \{1 \dots K\}$,

$$\frac{\partial L}{\partial \pi_j}(\pi, \theta, \lambda, \mu) = -\sum_{i=1}^{N} \frac{z_j^{(i)}}{\pi_j} + \lambda = -\frac{\Phi_j}{\pi_j} + \lambda$$

where $\forall j \in \{1, M \dots, \} \, \Phi_j = \#\left\{ i \ ; \ z^{(i)} = j \right\}$

$$\frac{\partial L}{\partial \theta_{lj}}(\pi, \theta, \lambda, \mu) = -\sum_{i=1}^{N} \frac{x_{jl}^{(i)}}{\theta_{jl}} + \mu_j = \frac{\Psi_{jl}}{\theta_{jl}} + \mu_j$$

where $\forall j \in \{1 \dots M\}, \forall l \in \{1 \dots K\}, \Psi_{jl} = \#\left\{ i \ ; \ x^{(i)} = l \wedge z^{(i)} = j \right\}$

At optimum, we have

$$\forall j \in \{1 \dots M\}, \frac{\Phi_j}{\pi_j} = \lambda$$

and

$$\forall j \in \{1 \dots M\}, \forall l \in \{1 \dots K\}, \frac{\Psi_{jl}}{\theta_{jl}} = \mu_j$$

since

$$\forall j \in \{1 \dots M\}, \sum_{j=1}^{M} \pi_j = \sum_{j=1}^{M} \frac{\Phi_j}{\lambda} = 1$$

We have $\lambda = N$. By applying the same reasonment,

$$\forall j \in \{1 \dots M\}, \sum_{l=1}^{K} \theta_{jl} = \sum_{l=1}^{K} \frac{\Psi_{jl}}{\mu_j} = 1$$

which lead to $\mu_j = \#\left\{i \ ; \ z^{(i)} = j\right\}$

$$\forall m \in \{1 \dots M\}, \hat{\pi}_m = \frac{\#\left\{i \ ; \ z^{(i)} = m\right\}}{N}$$
$$\forall m \in \{1 \dots M\}, \forall k \in \{1 \dots K\}, \hat{\theta}_{mk} = \frac{\#\left\{i \ ; \ x^{(i)} = l \wedge z^{(i)} = j\right\}}{\#\left\{i \ ; \ z^{(i)} = j\right\}}$$

2 Linear classification

2.1 Generative model (LDA)

(a) The parameters for this model are π, μ_0, μ_1 and Σ. Let N be the number of observations in the sample, and let $\theta = \pi, \mu_0, \mu_1, \Sigma$, the parameters of the model. we can express the likelihood as follows

$$\mathcal{L}(\theta) = p((x^{(1)}, y^{(1)}), ...(x^{(N)}, y^{(N)}); \theta)$$

We assume that the observations $(x^{(i)}, y^{(i)})$ are i.i.d .

$$\begin{aligned}\mathcal{L}(\theta) &= \prod_{i=1}^{N} p(x^{(i)}, y^{(i)}; \theta) \\ &= \prod_{i=1}^{N} p(x^{(i)}|y^{(i)}; \theta) p(y^{(i)}; \theta)\end{aligned}$$

We have

$$p(y^{(i)}; \pi) = \pi^{y^{(i)}} (1 - \pi)^{(1 - y^{(i)})}$$

and

$$p(x^{(i)}|y^{(i)}; \mu_i, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}})\right)$$

The log likelihood is then expressed as

$$\begin{aligned}\ln \mathcal{L}(\theta) &= \sum_{i=1}^{N} y^{(i)} \ln(\pi) + (1-y^{(i)}) \ln(1-\pi) - \frac{1}{2}\ln\left((2\pi)^d \det \Sigma\right) - \frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) \\ &= \sum_{i=1}^{N} y^{(i)} \ln(\pi) + (1-y^{(i)}) \ln(1-\pi) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(\det \Sigma) - \frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}})\end{aligned}$$

Maximum likelihood estimator for π

The log likelihood is strictly concave w.r.t π, so finding the maximum likelihood estimator for π is of finding π which satisfies partial derivate w.r.t π.

$$\begin{aligned}\frac{\partial}{\partial \pi} \ln \mathcal{L}(\theta) &= \sum_{i=1}^{N} \frac{y^{(i)}}{\pi} - \frac{(1-y^{(i)})}{(1-\pi)} \\ &= \frac{N_1}{\pi} - \frac{N_0}{(1-\pi)}\end{aligned}$$

Where $N_1 = \sum_{i=1}^{N} y^{(i)}$ and $N_0 = \sum_{i=1}^{N} (1-y^{(i)})$ (we have $N_0 + N_1 = N$)

$$\begin{aligned}\frac{\partial}{\partial \pi} \ln \mathcal{L}(\theta) = 0 &\Leftrightarrow \frac{N_1}{\pi} = \frac{N_0}{(1-\pi)} \\ &\Leftrightarrow \pi N_0 = (1-\pi) N_1 \\ &\Leftrightarrow \pi = \frac{N_1}{N}\end{aligned}$$

Maximum likelihood estimator for μ_j

Let $j \in \{0, 1\}$. Because Σ is supposed definite, Σ^{-1} is also definite (if Σ is only supposed semi-definite, it may not be invertible).

$$\begin{aligned}\nabla_{\mu_j} \ln \mathcal{L}(\theta) &= -\sum_{i=1}^{N} \delta_{(y^{(i)},j)} 2\Sigma^{-1}(x^{(i)} - \mu_j) \\ &= 2\Sigma^{-1}(S_j - N_j \mu_j)\end{aligned}$$

Where

$$\forall i, j \in \{0,1\}\, \delta_{(i,j)} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

S_j is the sum of the observations $x^{(i)}$ such that $y^{(i)}$ is is j, or more formally

$$S_j = \sum_{i=1}^{N} \delta_{(y^{(i)},j)} x^{(i)}$$

And N_j is the number of i such that $y^{(i)}$ is j (as in the previous section)

$$N_j = \sum_{i=1}^{N} \delta_{(y^{(i)},j)}$$

We then have

$$\nabla_{\mu_j} \ln \mathcal{L}(\theta) = 0 \Leftrightarrow 2\Sigma^{-1}(S_j - N_j\mu_j) = 0$$

Since Σ^{-1} is definite,

$$\nabla_{\mu_j} \ln \mathcal{L}(\theta) = 0 \Leftrightarrow \mu_j = \frac{S_j}{N_j} = \bar{x}_j$$

With $\bar{x}_j$ being the mean of the $x^{(i)}$ such that $y^{(i)} = j$

Maximum likelihood estimator for Σ

$$\nabla_\Sigma \ln \mathcal{L}(\theta) = \frac{\partial}{\partial \Sigma} \left(-\frac{N}{2} \ln (\det \Sigma) - \frac{1}{2} \sum_{i=1}^{N} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) \right)$$

Let $\Omega = \Sigma^{-1}$ and let g the function such that

$$g_{\pi,\mu_0,\mu_1}(\Omega) = L(\pi, \mu_0, \mu_1, \Omega^{-1})$$

Which we can rewrite

$$\begin{aligned} \nabla_\Sigma \ln \mathcal{L}(\theta) = \nabla_\Sigma \Bigg(& -\frac{N}{2} \ln (\det \Sigma) \\ & - \frac{1}{2} \sum_{i=1}^{N} y^{(i)} (x^{(i)} - \mu_1)^T \Sigma^{-1} (x^{(i)} - \mu_1) \\ & - \frac{1}{2} \sum_{i=1}^{N} (1 - y^{(i)}) (x^{(i)} - \mu_0)^T \Sigma^{-1} (x^{(i)} - \mu_0) \Bigg) \end{aligned}$$

Since

$$\begin{aligned} \sum_{i=1}^{N} (x^{(i)} - \mu_j)^T \Sigma^{-1} (x^{(i)} - \mu_j) &= \sum_{i=1}^{N} Tr \left((x^{(i)} - \mu_j)^T \Sigma^{-1} (x^{(i)} - \mu_j) \right) \\ &= \sum_{i=1}^{N} Tr \left(\Sigma^{-1} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T \right) \end{aligned}$$

We know that

$$\nabla_{\Sigma^{-1}} \ln(\det \Sigma^{-1}) = \Sigma$$

If we take the gradient with respect to Σ^{-1}, we obtain

$$\nabla_{\Sigma^{-1}} \ln \mathcal{L}(\theta) = \frac{N}{2}\Sigma - \frac{1}{2}\sum_{i=1}^{N} y_i (x^{(i)} - \mu_1)^T (x^{(i)} - \mu_1) - \frac{1}{2}\sum_{i=1}^{N} (1 - y_i)(x^{(i)} - \mu_0)^T (x^{(i)} - \mu_0)$$

which is 0 when

$$\begin{aligned}\Sigma &= \frac{1}{N}\left(\sum_{i=1}^{N} y_i (x^{(i)} - \mu_1)^T (x^{(i)} - \mu_1) + \sum_{i=1}^{N} (1 - y_i)(x^{(i)} - \mu_0)^T (x^{(i)} - \mu_0)\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} (x^{(i)} - \mu_{y_i})^T (x^{(i)} - \mu_{y_i})\end{aligned}$$

Conclusion

$$\begin{aligned}\hat{\pi}_{MLE} &= \frac{S_j}{N_j} \\ \hat{\mu_j}_{MLE} &= \frac{S_j}{N_j} \\ \hat{\Sigma}_{MLE} &= \frac{1}{N}\sum_{i=1}^{N} (x^{(i)} - \mu_{y_i})^T (x^{(i)} - \mu_{y_i})\end{aligned}$$

(b) We suppose $0 < \pi < 1$. From bayes rule, we have

$$\begin{aligned}p(y = 1|x) &= \frac{p(x|y = 1)p(y = 1)}{p(x|y = 1)p(y = 1) + p(x|y = 0)p(y = 0)} \\ &= \frac{1}{1 + \dfrac{p(x|y = 0)p(y = 0)}{p(x|y = 1)p(y = 1)}}\end{aligned}$$

Since we have

$$p(x|y = i) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma^{-1} (x - \mu_i)\right)$$

We can write

$$\begin{aligned}\frac{p(x|y = 0)p(y = 0)}{p(x|y = 1)p(y = 1)} &= \frac{\pi - 1}{\pi} \exp\left(-\frac{1}{2}\left((x - \mu_0)^T \Sigma^{-1} (x - \mu_0) - (x - \mu_1)^T \Sigma^{-1} (x - \mu_1)\right)\right) \\ &= \frac{\pi - 1}{\pi} \exp\left(-\frac{1}{2}\left(2x^T \Sigma^{-1}(\mu_0 - \mu_1) + \mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(2x^T \Sigma^{-1}(\mu_0 - \mu_1) + \mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1 + 2\ln(\pi) - 2\ln(\pi - 1)\right)\right) \\ &= \exp\left(x^T \omega + \beta\right)\end{aligned}$$

With $\omega = \Sigma^{-1}(\mu_1 - \mu_0)$ and $\beta = -\frac{1}{2}\left(\mu_0^T\Sigma^{-1}\mu_0 + \mu_1^T\Sigma^{-1}\mu_1\right) + \ln(\pi - 1) - \ln(\pi)$

(c) I've implemented a class LDA in the file *classify.py*. The function **`train`** is used to learn the parameters of the model and the function **`probability`** is used to predict the probability given samples

From the previous question, we can obtain the expression of the line defined by $p(y = 1|\theta)$

$$\begin{aligned} p(y = 1|x) = \frac{1}{2} &\Leftrightarrow \frac{1}{1 + \exp(\omega^T x + \beta)} = \frac{1}{2} \\ &\Leftrightarrow \exp(\omega^T x + \beta) = 1 \\ &\Leftrightarrow \omega^T x + \beta = 0 \end{aligned}$$

Note : I could have implemented the drawing of the line using this equation (In fact, I dit it at the beginning and it worked well), but I later faced the problem of drawing quadratic curve for the QDA, and I found the magic function **`contour`** which is very easy to use, and provides fancy result so I used it to draw every curves in order to have homogeneous figures. (The code is still available as a comment in the file *classify.py*

The sample data and decision boundary of this classifier are represented on Figure 1 p. 8. We define the *training error* (resp. the *test error*) as the ratio of observations in the training data (resp. in the test data) that are wrongly predicted by our predictor.

The parameters learnt for this methods are presented in the Table 1 p. 7.

sample file	π	μ_0	μ_1	Σ
classificationA	0.33	(2.90, -0.89)	(-2.69, 0.87)	$\begin{pmatrix} 2.46 & -1.14 \\ -1.14 & 0.62 \end{pmatrix}$
classificationB	0.5	(3.34, -0.84)	(-3.21, 1.08)	$\begin{pmatrix} 3.36 & -0.14 \\ -0.14 & 1.74 \end{pmatrix}$
classificationC	0.625	(2.79, -0.84)	(-2.94, -0.96)	$\begin{pmatrix} 2.89 & -0.64 \\ -0.64 & 5.21 \end{pmatrix}$

TABLE 1 – Parameters learnt for the LDA method for the different training sample

2.2 Logistic regression

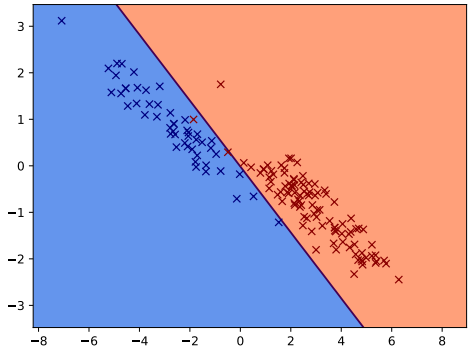
I have implemented the logistic regression using the Newton-Raphson method (IRLS) as suggested.

If we note

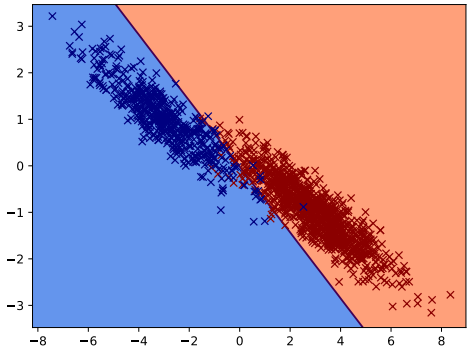
$$\forall i \in \{1 \dots N\}, n_i = \sigma(w^T x_i)$$

The update equation is

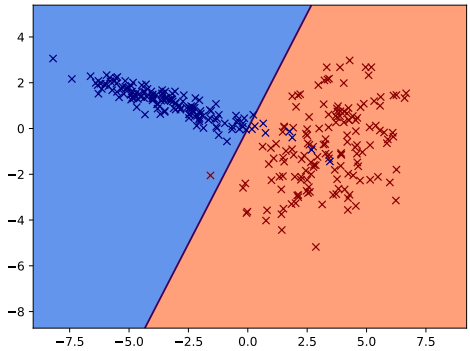
$$w_{t+1} = w_t + Hl(w_t)^{-1}\nabla l(w)$$



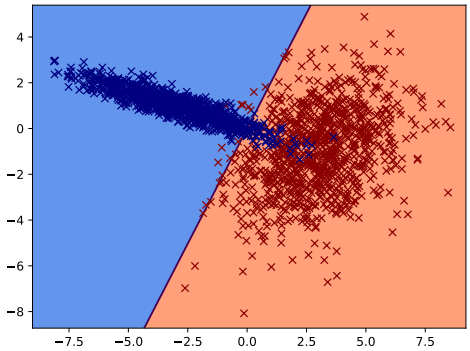
(a) Training observations A (150 points)



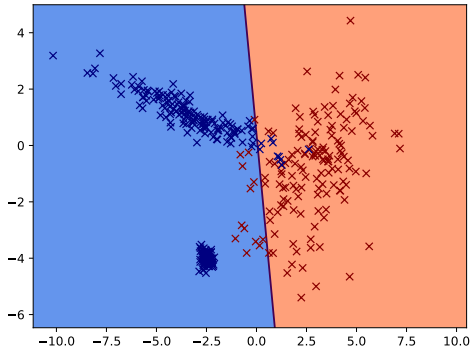
(b) Test observations A (1500 points)



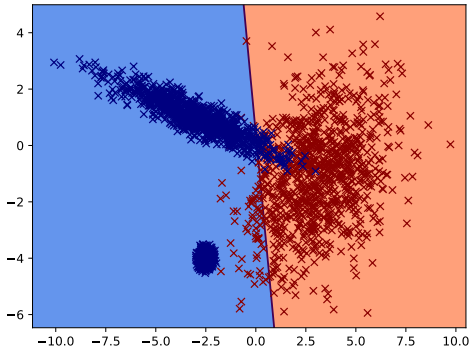
(c) Training observations B (150 points)



(d) Test observations B (1500 points)



(e) Training observations C (150 points)



(f) Test observations C (1500 points)

FIGURE 1 – Sample data and decision boundary representation for the LDA classifier on the three files

with

$$
\begin{aligned}
l(w) &= \sum_{i=1}^{N} \Big(y_i \ln\left(\sigma(w^T x)\right) + (1 - y_i) \ln\left(1 - \sigma(w^T x)\right) \Big) \\
Hl(w) &= -X^T diag(n \odot (1_{\mathbb{R}^N} - n)) X \\
\nabla l(w) &= X^T (y - n)
\end{aligned}
$$

Note : I've taken these equations from the lecture notes, to apply them, we need to add a column vectors of 1 to X, for the affine part

$\odot$ is the hadamar product of two matrices.

For the implementation, I used 10 iterations, and an initial vector of zeros. The parameters learnt are presented in the Table 2 p. 9

sample file	w
classificationA	$(42.63, 73.03, 7.80)$
classificationB	$(1.71, -1.02, -1.35)$
classificationC	$(2.20, -0.71, -0.96)$

TABLE 2 – Parameters learnt for the different training sample (no regularization)

We observe that for the *classificationA*, $\|w\|$ tends to ∞. This is due to the fact that the data are separable, We have

$$
p(y = 1|x) = p \Leftrightarrow w^T x = \ln\left(\frac{1-p}{p}\right)
$$

So the line $p(y = 1|x) = 0.5$ is the same when we multiply w by a constant $\alpha \in \mathbb{R}$, but for a fixed $p \in [0; 1]$, by multiplying w by α, we obtain

$$
\begin{aligned}
\alpha w^T x &= \alpha \ln\left(\frac{1-p}{p}\right) \\
&= \ln\left(\left(\frac{1}{p} - 1\right)^{\alpha}\right)
\end{aligned}
$$

If we consider q such that

$$
\frac{1}{q} - 1 = \left(\frac{1}{p} - 1\right)^{\alpha} \Leftrightarrow q = \frac{p^\alpha}{p^\alpha + (1-p)^\alpha}
$$

Which means that by multiplying w by α, the line $p(y = 1|x) = p$ becomes the line

$$
p(y = 1|x) = \frac{p^\alpha}{p^\alpha + (1-p)^\alpha}
$$

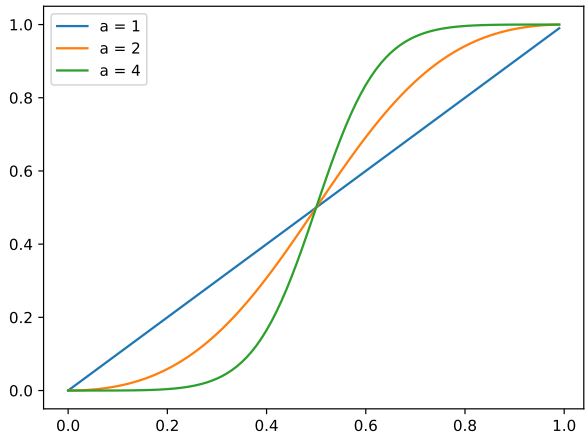


FIGURE 2 – Representation of f_α for different α

If we plot the function $f_\alpha : p \to \dfrac{p^\alpha}{p^\alpha + (1-p)^\alpha}$, (see Fig 2 p. 10) we see that for $\alpha > 1$

$$f(p) > p \text{ if } p > 0.5$$
$$f(p) < p \text{ if } p < 0.5$$

and also,

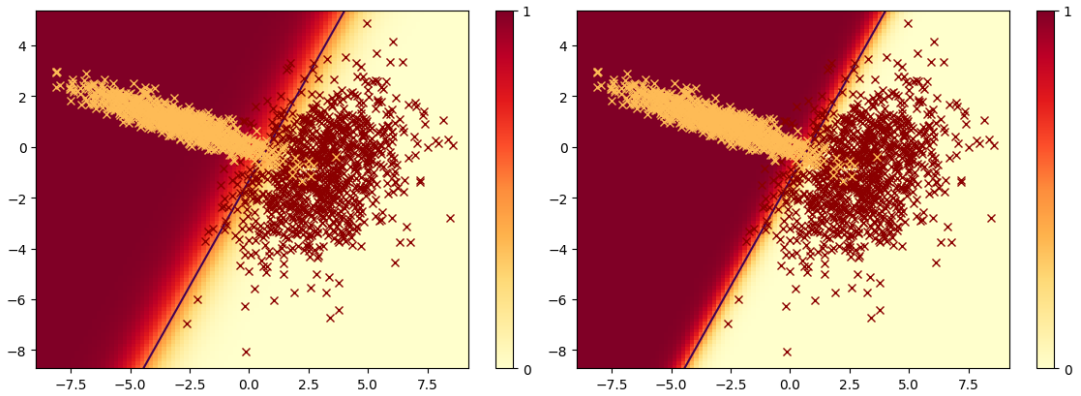
$$\lim_{\alpha \to \infty} f_\alpha(p) = \begin{cases} 0 & \text{if} \quad p < 0.5 \\ 0.5 & \text{if} \quad p = 0.5 \\ 1 & \text{if} \quad p > 0.5 \end{cases}$$

When the data is separable, the best solution is obviously to take w such that the hyperplane $w^T x = 0$ separates the data, and make $\|w\|$ tend to infinity such that all the observations x are such that $p(y = 1|x) = 0$ or $p(y = 1|x) = 1$ depending on whether they are located on one side of the half space (supported by the hyperplan) or the other. On the Figure 3 p. 11 I have plotted the value of $p(y = 1|x)$ on the entire plane, and we see the effect of multiplying w by $a > 1$ graphicaly.

We can avoid this effect by introducing a regularization term, e.g minimizing

$$\tilde{l}(w) = \sum_{i=1}^{N} \Big(y_i \ln\left(\sigma(w^T x)\right) + (1 - y_i) \ln\left(1 - \sigma(w^T x)\right) \Big) + \frac{\lambda}{2} \|w\|^2$$

Note : I've implemented this regularization method with a parameter $\lambda = 0.001$ in the file *classify.py*. This gives the same values for the parameters and misclassification error

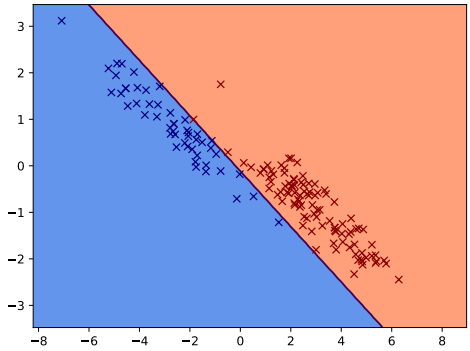


(a) $a = 1$ (w unchanged)

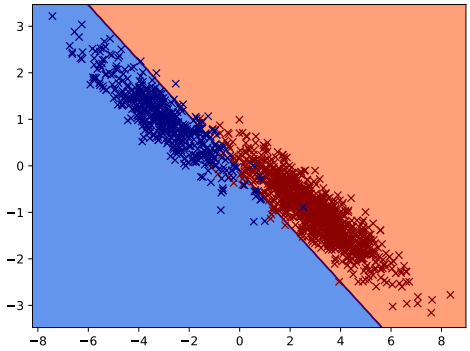
(b) $a = 2$

(c) $a = 4$

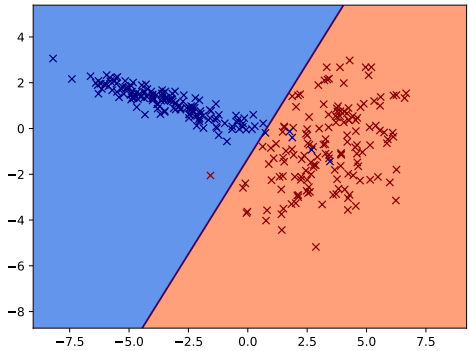
FIGURE 3 – Representation of the variation of the probability $p(y = 1|x)$ over the entire plane for the logistic regression when multiplying w by a constant term a (file *classificationB*)



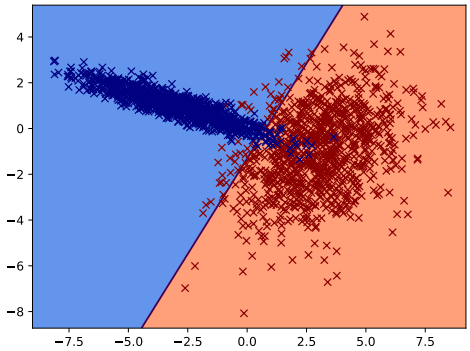
(a) Training observations A (150 points)



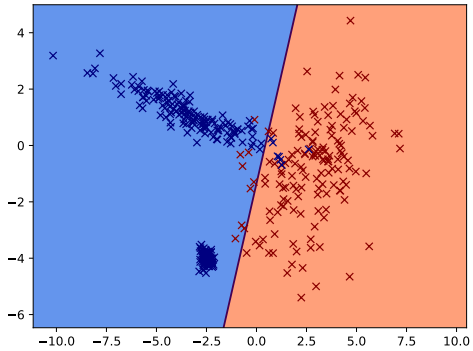
(b) Test observations A (1500 points)



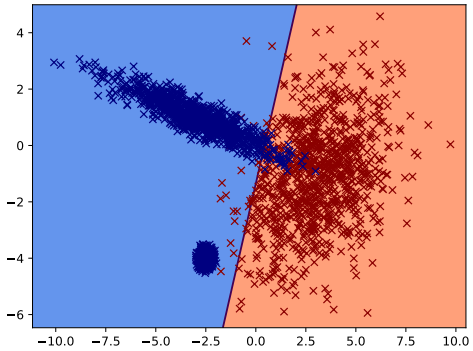
(c) Training observations B (150 points)



(d) Test observations B (1500 points)



(e) Training observations C (150 points)



(f) Test observations C (1500 points)

FIGURE 4 – Sample data and decision boundary representation for the *logistic regression* classifier on the three files

2.3 Linear Regression

(a) I've implemented the linear regression using the normal equation as suggested.

$$\hat{w} = (X^T X)^{-1} X^T y$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - w^T x^{(i)} \right)$$

The values of $\hat{w}$ and $\hat{\sigma}^2$ are given for the 3 training files :

sample file	$\hat{w}$	$\hat{\sigma}^2$
classificationA	$(-0.26, \ -0.37, \ 0.49)$	0.050
classificationB	$(-0.10, \ 0.05, \ 0.50)$	0.054
classificationC	$(-0.13, \ -0.02, \ 0.51)$	0.062

TABLE 3 – Parameters learnt for the linear regression method for the different training sample

(b) The representation of the data and the decision boundary are represented on Figure 5 p. 14

2.4 Comparison of the results

(a) I'll first present the results I've obtained on the train and test data. These results can be obtain by running the python script *classify.py* (it outputs train error and test error as well as the parameters for the different methods).

Here are the results of the different methods on the 3 sample files presented in an array. Each cell consists of two values E_{train} and E_{test} which are respectively the misclassification error on the training sample and on the test sample for this file.

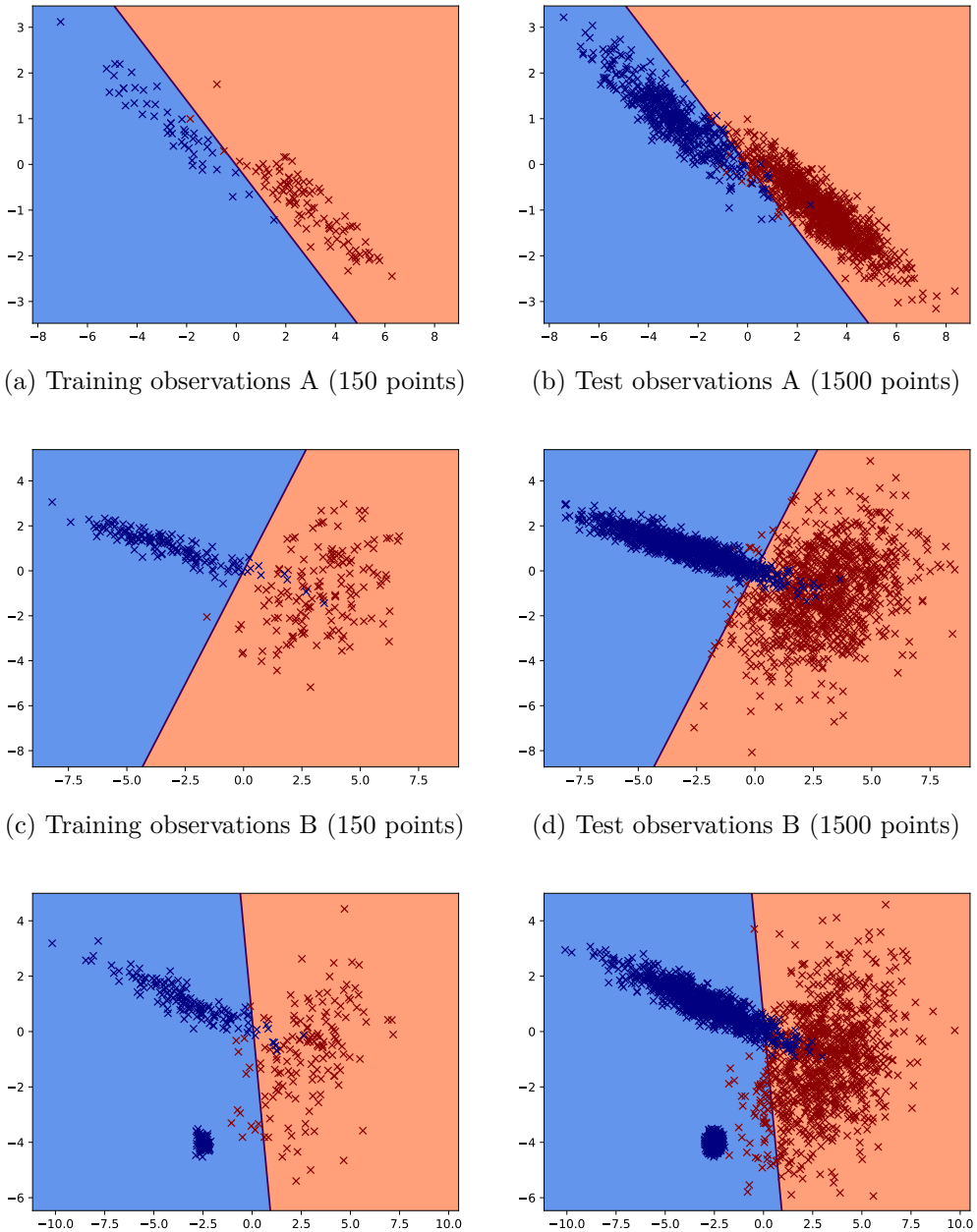
Method	*classificationA*	*classificationB*	*classificationC*
LDA	1.33% / 2.00%	3.00 / 4.15%	5.50% / 4.23%
Logistic Regression	0.00% / 3.47%	2.00 / 4.30%	4.00% / 2.27%
Linear Regression	1.33% / 2.07%	3.00 / 4.15%	5.50% / 4.23%
QDA	0.67% / 2.00%	1.33% / 2.00%	5.25% / 3.83%

TABLE 4 – Comparison of the misclassification errors (training / test) for the different methods and files

(b) For each file, the best classification method for the test sample is represented in green in the array.

SampleA

For the sample A, the 3 methods perform well, by looking at the data, we can guess that it has been generated from a weighted sum of 2 normal distribution with the same covariant



(a) Training observations A (150 points)

(b) Test observations A (1500 points)

(c) Training observations B (150 points)

(d) Test observations B (1500 points)

(e) Training observations C (150 points)

(f) Test observations C (1500 points)

FIGURE 5 – Sample data and decision boundary representation for the *linear regression* classifier on the three files

matrix (and of course different means), so this is not surprising that the *LDA* performs well (the model fits the data perfectly). Moreover the training data is separable (we can see that by the logistic regression having a training misclassification error of 0.00), which explains why the training error are very low. The test data has very little overlapping, which explains why all the methods performs well. The logistic regression induces a small overfitting on this sample, it matches perfectly the training samples, but has the worst test error.

SampleB

For the sample *B*, we can guess that it has also been generated from a weighted sum of 2

normal distribution, but this time with different covariant matrix. That explains why the *QDA* is outperforming the 2 other methods.

There's an overlapping zone between the 2 gaussian kernels, which explains why the performance is overall worst that the SampleA, the 3 other methods that try to fit a line (e.g. LDA, Linear Regression and Logistic Regression), have very similar results on the test sample.

SampleC

The sample C is very similar to the sample B, the only change is that there's an additional dense gaussian kernel whose points are labeled $y^{(i)} = 1$. We can notice that all 3 methods perform well (less than 5% misclassification error on the test data), though none of the model fits the data perfectly.

The logistic regression performs better on this dataset. This is a really interesting example, since the small dense gaussian kernel added from SampleB to SampleC would be in fact correctly classified by all the models trained with B (e.g trained without this extra data) What we are testing is in fact the stability of the method, and we expect a small change in the boundary line in order for the method to be stable.

I have computed the angle between the boundary line and the x-axis for linear regression and logistic regression, for both SampleB and SampleC, the results are presented in the Table 5 p. 15. The first thing we can notice is that the angles for the two methods are similar for the file SampleB. We also see that the angle change between SampleB and SampleC is only 0.23 rad for the logistic regression versus 0.59 rad for the linear regression. This means that the logistic regression is much more stable than the linear regression.

Method	*classificationB*	*classificationC*	difference
Linear Regression	1.11 rad / 63 deg	1.70 rad / 98 deg	0.59 rad / 34 deg
Logistic Regression	1.03 rad / 59 deg	1.26 rad / 72 deg	0.23 rad / 13 deg

TABLE 5 – Comparison of the angles between the boundary line and the x-axis for the files *classificationB* and *classificationC*

2.5 QDA model

(a) For the QDA model, we can generalize the expression of the likelihood found for the QDA.

$$p(x^{(i)}|y^{(i)}) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_{y^{(i)}}}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T \Sigma_{y^{(i)}}^{-1} (x^{(i)} - \mu_{y^{(i)}})\right)$$

We obtain this expression for the log likelihood

$$\begin{aligned}
\ln \mathcal{L}(\theta) = & -\frac{dN}{2} \ln(2\pi) \\
& + \sum_{i=1}^{N} \left(y^{(i)} \ln(\pi) + (1 - y^{(i)}) \ln(1 - \pi) \right) \\
& - \frac{1}{2} \sum_{i=1}^{N} y^{(i)} \left(\ln \det \Sigma_1 + (x^{(i)} - \mu_1)^T \Sigma_1^{-1} (x^{(i)} - \mu_1) \right) \\
& - \frac{1}{2} \sum_{i=1}^{N} (1 - y^{(i)}) \left(\ln \det \Sigma_0 + (x^{(i)} - \mu_0)^T \Sigma_0^{-1} (x^{(i)} - \mu_0) \right)
\end{aligned}$$

We see that the maximum likelihood estimator for π, μ_0 and μ_1 are identical for the QDA and the LDA.

Moreover, using the same notation that we used for the LDA, we can obtain the maximum likelihood estimators for Σ_0 and Σ_1 as follows

$$\hat{\Sigma}_j = \frac{1}{N_j} \sum_{i=0}^{N} \delta_{j,y^{(i)}} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T$$

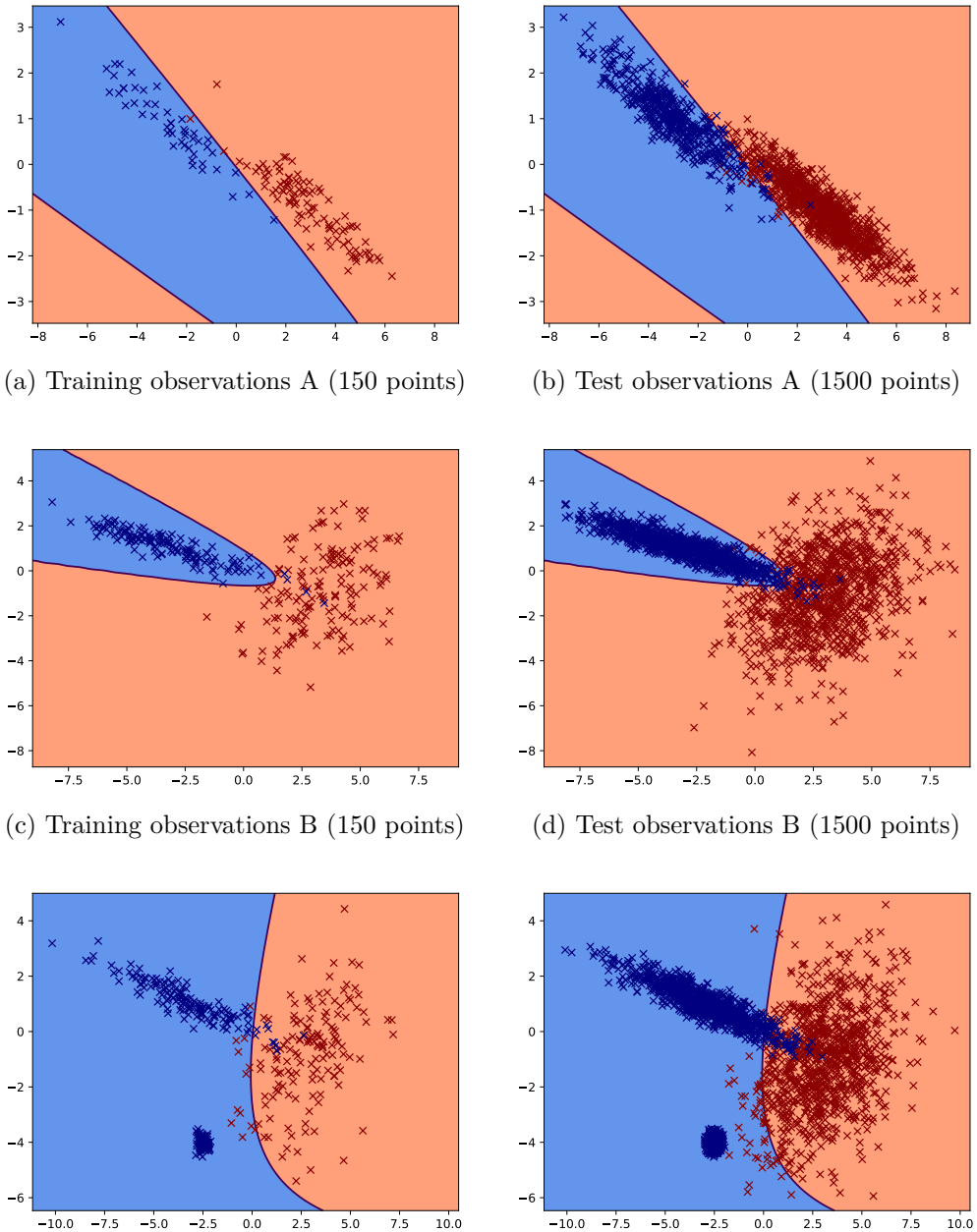
The values obtained from my implementation of the QDA are represented in the following table

sample file	π	μ_0	μ_1	Σ_0	Σ_1
classificationA	0.33	(2.90, -0.89)	(-2.69, 0.87)	$\begin{pmatrix} 2.33 & -1.06 \\ -1.06 & 0.58 \end{pmatrix}$	$\begin{pmatrix} 2.76 & -1.33 \\ -1.33 & 0.70 \end{pmatrix}$
classificationB	0.5	(3.34, -0.84)	(-3.22, 1.08)	$\begin{pmatrix} 2.56 & 1.07 \\ 1.07 & 2.98 \end{pmatrix}$	$\begin{pmatrix} 4.18 & -1.34 \\ -1.34 & 0.52 \end{pmatrix}$
classificationC	0.625	(2.79, -0.84)	(-2.94,-0.96)	$\begin{pmatrix} 2.92 & 1.25 \\ 1.25 & 2.94 \end{pmatrix}$	$\begin{pmatrix} 2.88 & -1.77 \\ -1.77 & 6.59 \end{pmatrix}$

TABLE 6 – Parameters learnt for the QDA method for the different training sample

(b) The sample data and decision boundary of this classifier are represented on Figure 6 p. 17

(c) Table 7 p. 17 presents the misclassification error for both training and test sample (it's also present in Table 4 p. 13, which regroups the misclassification error for all methods)



(a) Training observations A (150 points)

(b) Test observations A (1500 points)

(c) Training observations B (150 points)

(d) Test observations B (1500 points)

(e) Training observations C (150 points)

(f) Test observations C (1500 points)

FIGURE 6 – Sample data and decision boundary representation for the *QDA* classifier on the three files

Method	*classificationA*	*classificationB*	*classificationC*
Training error	0.67%	1.33%	5.25%
Test error	2.00%	2.00%	3.83%

TABLE 7 – Comparison of the training and test misclassification error for the QDA on the different training sample

(d) The QDA method provides the overall best results in particular for samples B and C. It's no suprise that it performs better than the LDA, since the LDA is a particular case of the QDA when $\Sigma_0 = \Sigma_1$, we can see that on the Table 6 p. 16, the 2 matrix found are very close $\|\Sigma_0 - \Sigma_1\|^2 \approx 0.341$, and we observe in the Figure 6a p. 17 and Figure 6b p. 17 that the decision

boundary is approximately a line.

3 Implementation details

I've implemented the 4 classifiers (LDA, QDA, Linear Regression and Logistic Regression) in Python. The script is located in the file *src/classify.py*, it trains all classifiers on the files *data/classificationA.train*, *data/classificationB.train*, *data/classificationC.train* compute and print the parameters, and test the trained classifier on the files *data/classificationA.test*, *data/classificationB.test*, *data/classificationC.test* it then computes and outputs the train and the test misclassification error It also save graphs under *imgs/* showing for each classifier and each file, the data sample and the decision boundary.

Each classifier corresponds to a class that inherit the class *ClassificationModel*. This class is responsible for classifying and plot the contour of data based on the subclasses overriding the function *probability*. Each classifier class has 3 methods : *train* which is used for training the classifier, *probability* which estimates the probability that a given point X belongs to class 1, and *print_params* which prints the parameters of the model.

There's also a script that generates data sample from two gaussian kernels *generate.py* that I used to check and to experiment the classification methods (it's not very user friendly but I let it in the archive nonetheless).

4 Conclusion

This homework was interesting and visual, I really enjoyed doing it. This was also the opportunity for me to learn Python, I've tried to make a code as clear as possible though there's not much comments in the code... I should also have implemented a stopping criterion for the Newton-Raphson method on Logistic regression, (I chose a fixed number of iterations) a decent criterion would be to stop when the gradient is sufficiently small.