Master MVA
Probabilistic Graphical Models

Homework 1: Classification models

Victor Busa
victor.busa@ens-paris-saclay.fr

January 16, 2018

1 Learning in discrete graphical models

Let z and x be discrete variables taking respectively M and K different values with $p(z = m) = \pi_m$ and $p(x = k|z = m) = \theta_{mk}$

Goal: Compute the maximum likelihood estimator for π and θ based on an i.i.d sample of observations

We can reformulate the probability of x knowing z as follow:

$$p(x = k|z = m) = p(X_k = 1|Z_m = 1)$$

Where

$$X = [X_1, X_2, \dots, X_K],\ Z = [Z_1, Z_2, \dots, Z_M]$$
$$X \in \{0,1\}^K \text{ s.t. } \sum_{k=1}^{K} X_k = 1 \text{ and } Z \in \{0,1\}^M \text{ s.t. } \sum_{m=1}^{M} Z_m = 1$$
$$\{x = k\} = \{X_k = 1\} \text{ and } \{z = m\} = \{Z_m = 1\}$$

With these notations, we can rewrite:

$$p(z; \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_k}$$
$$p(x|z; \boldsymbol{\theta}) = \prod_{k=1}^{K} \prod_{m=1}^{M} \theta_{mk}^{z_m x_k}$$

And, using conditional probability, we can write:

$$p(x, z; \boldsymbol{\pi}, \boldsymbol{\theta}) = p(x|z; \boldsymbol{\theta})p(z; \boldsymbol{\pi}) \quad = \prod_{k=1}^{K} \prod_{m=1}^{M} \theta_{mk}^{z_m x_k} \prod_{k=1}^{K} \pi_k^{z_k}$$

Then, considering that we have N observations, and using the fact that the sample of observation x_i, z_i is i.i.d, we can write:

$$p(x, z; \boldsymbol{\pi}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(x_i, z_i; \boldsymbol{\pi}, \boldsymbol{\theta})$$

and, we can compute the log-likelihood:

$$\begin{aligned}
\ell(\boldsymbol{\pi}, \boldsymbol{\theta}|x, z) &= log\left(\prod_{i=1}^{N} p(x_i, z_i; \boldsymbol{\pi}, \boldsymbol{\theta})\right) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M} z_m^{(i)} x_k^{(i)} log(\theta_{mk}) + \sum_{i=1}^{N}\sum_{m=1}^{M} z_k^{(i)} log(\pi_k) \\
&= \sum_{k=1}^{K}\sum_{m=1}^{M} N_{mk} log(\theta_{mk}) + \sum_{m=1}^{M} N_m log(\pi_m)
\end{aligned}$$

where we used the notation:

$$N_{mk} \triangleq \sum_{i=1}^{N} z_m^{(i)} x_k^{(i)} \ \{\text{number of times we observed } \{x = k \wedge z = m\}\}$$

$$N_m \triangleq \sum_{i=1}^{N} z_m^{(i)} \ \{\text{number of times we observed } \{z = m\}\}$$

$(\boldsymbol{\pi}, \boldsymbol{\theta}) \to \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}|x, z)$ is strictly concave (linear combination of logarithm functions that are strictly concave on their domains) and differentiable on its domain:
$\mathcal{D} = \{\theta_{mk} \geq 0 \ \forall (m, k) \in [1, M] \times [1, K], \ \pi_m \geq 0 \ \forall m \in [1, M], \text{ s.t. } \sum_k \theta_{mk} = \sum_m \pi_m = 1\}$
We want to maximize this function. Maximizing $\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}|x, z)$ is equivalent to minimize -($\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}|x, z)$) which is a convex optimization problem on its domain $\mathcal{D}$, since the objective function is convex and the equality constraints: $\mathbf{1}^\intercal \pi = 1$ and $\theta \mathbf{1} = \mathbf{1}$ are affine constraints.

Hence, we want to minimize the following convex optimization problem on its domain $\mathcal{D}$:

$$\begin{aligned}
&\text{minimize} \ \ -\ell(\boldsymbol{\pi}, \boldsymbol{\theta}|x, z) \\
&\text{subject to} \ \sum_{m=1}^{M} \pi_m = 1 \qquad (1) \\
&\qquad\qquad \sum_{k=1}^{K} \theta_{mk} = 1 \text{ for m} = 1, \ldots, M \qquad (2)
\end{aligned}$$

Moreover, we can notice that this convex problem satisfies Slater's condition (for example the vector $\boldsymbol{\pi} = \frac{1}{M}\mathbf{1}$ and the matrix $\boldsymbol{\theta} = \frac{1}{MK}\mathbf{1}$ are in the interior of the constrained problem), so strong duality holds and we can retrieve the optimal solution of the problem by maximizing the dual problem.

We will maximize the Lagragian $\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}, \lambda, \mu_1, \mu_2, \ldots, \mu_M | x, z)$ w.r.t π and θ:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\theta}, \lambda, \mu_1, \mu_2, \ldots, \mu_M | x, z) = &- \sum_{k=1}^{K}\sum_{m=1}^{M} N_{mk} log(\theta_{mk}) - \sum_{m=1}^{M} N_m log(\pi_m) \\ &+\lambda(\sum_{m=1}^{M}\pi_m - 1) + \sum_{m=1}^{M}\mu_m(\sum_{k=1}^{K}\theta_{mk} - 1)\end{aligned}$$

The Lagragian of a convex optimization problem is convex, so to retrieve the maximum, we need to solve:

$$\frac{\partial \mathcal{L}}{\partial \pi_m} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \theta_{mk}} = 0$$

We have:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \pi_m} &= -\frac{N_m}{\pi_m} + \lambda = 0 \Rightarrow \pi_m = \frac{N_m}{\lambda} \\ \frac{\partial \mathcal{L}}{\partial \theta_{mk}} &= -\frac{N_{mk}}{\theta_{mk}} + \mu_m = 0 \Rightarrow \theta_{mk} = \frac{N_{mk}}{\mu_m}\end{aligned}$$

Using the constraints (1) and (2) we can write:

$$\frac{\sum_{m=1}^{M} N_m}{\lambda} = \sum_{m=1}^{M}\pi_m = 1 \Rightarrow \lambda = \sum_{m=1}^{M} N_m = N \quad \{\text{number of observations}\}$$

$$\frac{\sum_{k=1}^{K} N_{mk}}{\mu_m} = \sum_{k=1}^{K}\theta_{mk} = 1 \Rightarrow \mu_m = \sum_{k=1}^{K}\theta_{mk} = N_m \quad \{\text{number of times we observed } \{z = m\}\}$$

Conclusion:

The maximum likelihood estimators $\widehat{\pi_m}$ and $\widehat{\theta_{mk}}$ for this problem are:

$$\begin{aligned}\widehat{\pi_m} = \frac{N_m}{N} &\triangleq \frac{\text{number of times we observed } \{z = m\}}{\text{number of observations}} \\ \widehat{\theta_{mk}} = \frac{N_{mk}}{N_m} &\triangleq \frac{\text{number of times we observed } \{x = k \wedge z = m\}}{\text{number of times we observed } \{z = m\}}\end{aligned}$$

2 Linear classification

2.1 Generative model (LDA)

(a) Let $y \sim Ber(\pi)$, and $x|\{y=i\} \sim \mathcal{N}(\mu_i, \Sigma)$. We will derive the form of the maximum likelihood estimator for this model.

We have:

$$p(y,\pi) = \pi^y(1-\pi)^{1-y}$$
$$p(x|y;\boldsymbol{\mu_i},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(x-\mu_{y_i})^\intercal \Sigma^{-1}(x-\mu_{y_i}))$$

using the fact that the samples are i.i.d and the Bayes rule we can compute the log-likelihood as follow:

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= log\left(\prod_{i=1}^{N} p(X=x_i, Y=y_i;\boldsymbol{\theta})\right) = log\left(\prod_{i=1}^{N} p(X=x_i|Y=y_i;\boldsymbol{\theta})p(Y=y_i;\boldsymbol{\theta})\right) \\
&= \sum_{i=1}^{N} log(p(X=x_i|Y=y_i;\boldsymbol{\theta})) + log(p(Y=y_i;\boldsymbol{\theta})) \\
&= \sum_{i=1}^{N}\left(-log(2\pi) - \frac{1}{2}log|\Sigma| - \frac{1}{2}(x-\mu_{y_i})^\intercal\Sigma^{-1}(x-\mu_{y_i})\right) + \sum_{i=1}^{N}(y_i log(\pi) + (1-y_i)log(1-\pi)) \\
&= -\frac{N}{2}log|\Sigma| - Nlog(2\pi) - \frac{1}{2}\sum_{i=1}^{N}(x-\mu_{y_i})^\intercal\Sigma^{-1}(x-\mu_{y_i}) + N_1 log(\pi) + (N-N_1)log(1-\pi) \\
&= -\frac{N}{2}log|\Sigma| - Nlog(2\pi) - \frac{1}{2}\sum_{i=N_1}^{N}(x-\mu_0)^\intercal\Sigma^{-1}(x-\mu_0) \\
&- \frac{1}{2}\sum_{i=1}^{N_1}(x-\mu_1)^\intercal\Sigma^{-1}(x-\mu_1) + N_1 log(\pi) + (N-N_1)log(1-\pi)
\end{aligned}$$

Where we assumed that the samples are of size N and we introduced the notation $\boldsymbol{\theta} = (\pi, \mu_0, \mu_1, \Sigma)$
For the third equality we used the fact that $d = 2$ (Indeed: $x_i \in \mathbb{R}^2$, $\Sigma \in \mathbb{R}^{2\times 2}$), and we introduced the notation: $N_1 \triangleq \sum_{i=1}^{N} y_i$ (number of points belonging to class 1). Hence: $N - N_1 \triangleq N_0$ represents the number of points belonging to class 0.
In the fourth equality, we use the fact that there are N_1 points belonging to class 1 and $N - N_1$ points belonging to class 0. We also assumed, without loss of generality, that the first N_1 points are in class 1 and that the remaining points (from N_1 to N) are in class 0.

We want to maximize the log-likelihood. As, here, the log-likelihood isn't a function strictly concave, computing the estimators for which the gradient is null is not enough to conclude that the

point is a maximum. Yet, computing the estimators for which the gradient is null will provide us a stationary point and we can conclude that the point is a maximum by studying the sign of the Hessian.

$f : \pi \to \mathcal{L}(\pi, \mu_0, \mu_1, \Sigma)$ is differentiable and strictly concave on its domain, as a weighted positive sum of function strictly concave (the logarithm function is strictly concave on its domain). From computing the derivatives of f w.r.t to π and setting it to zero, we obtain the following necessary condition for π to be an extremum of f:

$$\frac{N_1}{\pi} - \frac{N - N_1}{1 - \pi} = 0 \Rightarrow \pi = \frac{N_1}{N} \tag{1}$$

the function $g : \mu_0 \to \mathcal{L}(\pi, \mu_0, \mu_1, \Sigma)$ is differentiable and strictly concave on its domain as it is the opposite of a quadratic form (Σ is **positive definite**). From computing the derivatives of g w.r.t to μ_0 (as seen in class) and setting it to zero, we obtain the following necessary condition for μ_0 to be an extremum of g:

$$-\sum_{i=1}^{N-N_1} \Sigma^{-1}(\mu_0 - x_i) = 0$$

And, as Σ is positive definite, Σ^{-1} is positive definite and hence invertible, so we have:

$$\begin{aligned} -\sum_{i=N_1}^{N} \Sigma^{-1}(\mu_0 - x_i) &= 0 \\ \Leftrightarrow \sum_{i=N_1}^{N} (\mu_0 - x_i) &= 0 \\ \Leftrightarrow \mu_0 = \frac{1}{N - N_1} \sum_{i=N_1}^{N} x_i \triangleq \frac{1}{N_0} \sum_{i=N_1}^{N} x_i & \end{aligned}$$

Using the same procedure, the maximum of $\mu_1 \to \mathcal{L}(\pi, \mu_0, \mu_1, \Sigma)$ is reached for

$$\mu_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} x_i$$

Finally, we maximize the function $h : \Sigma \to \mathcal{L}(\pi, \mu_0, \mu_1, \Sigma)$. As seen in class, this function is not strictly concave. Yet, we can retrieve a stationary point by computing the derivatives of h w.r.t to Σ and setting it to zero. We obtain the following **necessary** condition for Σ to be an extremum of h:

$$\nabla_\Sigma \left(\sum_{i=N_1}^{N} (x-\mu_0)^\intercal \Sigma^{-1}(x-\mu_0) + \sum_{i=1}^{N_1} (x-\mu_1)^\intercal \Sigma^{-1}(x-\mu_1) - Nlog\left|\Sigma^{-1}\right| \right) = 0$$

$$(N-N_1)\frac{1}{N-N_1}\sum_{i=N_1}^{N}(x-\mu_0)(x-\mu_0)^\intercal + N_1\frac{1}{N_1}\sum_{i=1}^{N_1}(x-\mu_1)(x-\mu_1)^\intercal - N\Sigma = 0$$

$$\Rightarrow \Sigma = \frac{N_1}{N}\Sigma_1 + \frac{N_0}{N}\Sigma_0 = \frac{1}{N}(N_1\Sigma_1 + N_0\Sigma_0) \tag{2}$$

Where we used the fact that $N_0 \triangleq N - N_1$ and we used the notations:

$$\begin{aligned} \Sigma_0 &= \frac{1}{N_0}\sum_{i=N_1}^{N}(x-\mu_0)(x-\mu_0)^\intercal \\ \Sigma_1 &= \frac{1}{N_1}\sum_{i=1}^{N_1}(x-\mu_1)(x-\mu_1)^\intercal \end{aligned} \tag{3}$$

Finally, using the fact that $\forall i \in [1, N]$, $y_i \in \{0, 1\}$ with $y_i = 0$ if x_i is in class 0 and $y_i = 1$ if x_i belongs to class 1, we can rewrite:

$$\begin{aligned} \mu_0 &= \frac{1}{N_0}\sum_{i=1}^{N_0} x_i = \sum_{i=1}^{N}(1-y_i)x_i \\ \mu_1 &= \frac{1}{N_1}\sum_{i=N_1}^{N} x_i = \sum_{i=1}^{N} y_i x_i \end{aligned} \tag{4}$$

Conclusion:

Using relations (1), (2), (3) and (4), the maximum likelihood estimators are:

$$\pi = \frac{N_1}{N}$$
$$\mu_0 = \sum_{i=1}^{N}(1-y_i)x_i$$
$$\mu_1 = \sum_{i=1}^{N} y_i x_i$$
$$\Sigma = \frac{1}{N}(N_1\Sigma_1 + N_0\Sigma_0)$$

where:

$$\Sigma_0 = \frac{1}{N_0}\sum_{i=N_1}^{N}(x-\mu_0)(x-\mu_0)^\intercal$$
$$\Sigma_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}(x-\mu_1)(x-\mu_1)^\intercal$$
$$N_1 \triangleq \{\text{number of points belonging to class 1}\}$$
$$N_0 \triangleq \{\text{number of points belonging to class 0}\}$$

(b) What is the form of the conditional distribution $p(y=1|x)$?
Using Bayes formula we can compute $p(y=1|x)$ as follow:

$$\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} \\
&= \frac{p(x|y=1)p(y=1)}{\sum_y p(x,y)} \\
&= \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)+p(x|y=1)p(y=1)} \\
&= \frac{1}{1+\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}} \\
&= \frac{1}{1+\frac{\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}exp(-\frac{1}{2}(x-\mu_0)^\intercal\Sigma^{-1}(x-\mu_0)}{\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}exp(-\frac{1}{2}(x-\mu_1)^\intercal\Sigma^{-1}(x-\mu_1)}} \\
&= \frac{1}{1+exp\left(\underbrace{\frac{1}{2}((x-\mu_1)^\intercal\Sigma^{-1}(x-\mu_1)-(x-\mu_0)^\intercal\Sigma^{-1}(x-\mu_0))}_{f(\Sigma,\mu_1,\mu_0,x)}\right)\frac{1-\pi}{\pi}}
\end{aligned} \tag{5}$$

We can rewrite $f(\Sigma, \mu_1, \mu_0, x)$ as follow:

$$\begin{aligned} f(\Sigma, \mu_1, \mu_0, x) &= \frac{1}{2}((x-\mu_1)^\intercal \Sigma^{-1}(x-\mu_1) - (x-\mu_0)^\intercal \Sigma^{-1}(x-\mu_0)) \\ &= \frac{1}{2}\left(x^\intercal \Sigma^{-1} x + \mu_1^\intercal \Sigma^{-1}\mu_1 - 2\mu_1^\intercal \Sigma^{-1} x - x^\intercal \Sigma^{-1} x - \mu_0^\intercal \Sigma^{-1}\mu_0 + 2\mu_0^\intercal \Sigma^{-1} x\right) \\ &= (\mu_0 - \mu_1)^\intercal \Sigma^{-1} x + \frac{1}{2}\left(\mu_1^\intercal \Sigma^{-1}\mu_1 - \mu_0^\intercal \Sigma^{-1}\mu_0\right) \end{aligned}$$

Also, we can rewrite $\frac{1-\pi}{\pi} = exp(log(\frac{1-\pi}{\pi}))$, so that, finally we have:

$$\begin{aligned} p(y=1|x) &= \frac{1}{1+exp(-(\beta^\intercal x + \gamma))} \triangleq \sigma(\beta^\intercal + \gamma) \\ &\text{where:} \\ \beta &= (\mu_1 - \mu_0)^\intercal \Sigma^{-1} \\ \gamma &= \frac{1}{2}\left(\mu_0^\intercal \Sigma^{-1}\mu_0 - \mu_1^\intercal \Sigma^{-1}\mu_1\right) - log\left(\frac{1-\pi}{\pi}\right) \end{aligned}$$

(c) Implementation
See the python code and the comments for a brief explanation.

- For the **training dataset A** that contains 150 points, the values of the parameters are:
  - μ_0 = [2.89970947 -0.893874]
  - μ_1 = [-2.69232004 0.866042]
  - Σ = [[2.44190897 -1.13194024][-1.13194024 0.61375465]]
  - π = 0.33
  - β = [-6.62245258 -9.3462503]
  - γ = -0.136496290948
- For the **training dataset B** that contains 300 points, the values of the parameters are:
  - μ_0 = [3.34068896 -0.83546333]
  - μ_1 = [-3.21670734 1.08306733]
  - Σ = [[3.34623467 -0.13516489][-0.13516489 1.73807475]]
  - π = 0.5
  - β = [-1.92108197 0.95442836]
  - γ = 0.000929288716541
- For the **training dataset C** that contains 400 points, the values of the parameters are:
  - μ_0 = [2.79304824 -0.83838667]
  - μ_1 = [-2.94232885 -0.9578284]

- Σ = [[2.88039225 -0.63405081][-0.63405081 5.19952435]]
- π = 0.625
- β = [-2.05129911 -0.27311529]
- γ = 0.112429132177

→ See graphs in Annexe 3.1

2.2 Logistic regression

See the python code and the comments for a brief explanation.

(a) Numerical values of the parameters

- For the **training dataset A** that contains 150 points, the values of the parameters are:
  - β = [-1448.78155254 -2513.63096139]
  - γ = -253.264128442
- For the **training dataset B** that contains 300 points, the values of the parameters are:
  - β = [-1.70518585956 1.02378537713]
  - γ = 1.34959157315
- For the **training dataset C** that contains 400 points, the values of the parameters are:
  - β = [-2.20323239693 0.709265621318]
  - γ = 0.959188854105

Note: For the **training dataset A**, the data is perfectly separable. Hence, the values of the parameter retrieve by the algorithm doesn't make sense because the minimum of the objective function is not attained and correspond to $|\beta| \to +\infty$. Indeed, the algorithm stops before because the number for the logarithm is too small and Python tell us: 'divide by zero encountered in log'. To avoid this situation, we could have implemented a criterion to stop the algorithm instead of using a number of iterations. We could have also implemented the Tikhonov regularization.

(b) → See graphs in Annexe 3.1

2.3 Linear regression

See the python code and the comments for a brief explanation.

(a) Numerical values of the parameters

- For the **training dataset A** that contains 150 points, the values of the parameters are:
  - β = [-0.264007502359 -0.372593109145]
  - γ = -0.00770796243524

- For the **training dataset B** that contains 300 points, the values of the parameters are:
  - $\beta =$ [-0.104245751042 0.0517911796205]
  - $\gamma =$ 5.0426999757e-05
- For the **training dataset C** that contains 400 points, the values of the parameters are:
  - $\beta =$ [-0.127693330499 -0.0170014214318]
  - $\gamma =$ 0.00839981582635

(b) → See graphs in Annexe 3.1

2.4 Models performance

See python code and the comments for a brief explanation.

(a) Misclassification errors for LDA, Logistic regression and Linear regression

Algorithms	**LDA**	**Logistic regression**	**Linear regression**
Training set A	1.33	**0.0**	1.33
Testing set A	**2.0**	3.53	2.06
Training set B	3.0	**2.0**	3.0
Testing set B	**4.15**	4.3	4.15
Training set C	5.5	**4.0**	5.5
Testing set C	4.23	**2.26**	4.23

Table 1: Misclassification errors in percentage (%)

(b) the models performance depend upon the dataset. So I will analyze the performance of each algorithms with respect to each dataset.

- **Dataset A**: The misclassification error is low for all three methods on both the training set and the testing set. Yet, we can notice that the Logistic Regression achieve 100 % accuracy on the training set. This is due to the fact that the data is linearly separable (Figure 3.1). The fact that the data is linearly separable cause a higher misclassification errors for the testing sets and in particular we can clearly see that Logistic Regression overfits. Also, LDA leads to the slightest error rate. This is due to the fact that the data is drawn from 2 Gaussians having exactly the same covariance matrix and that the LDA actually models this situation.
- **Dataset B**: The data seems to come from 2 Gaussian with different covariance matrices. Here, we can notice (Figure 3.2) that all the algorithms struggle to find a **linear** bound between the points at the extremities of the 2 different Gaussians. The misclassification errors are quite similar for this dataset (around 4.2 for all three models). We could also expect that QDA model will outperform all the linear models because QDA exactly model 2 Gaussian distributions with different covariances.

- **Dataset C**: Here, the presence of a third little cluster (Figure 3.3) cause the misclassification error for all the linear classifiers to increase. Also, we can notice that logistic regression didn't suffer from the fact that there is one more 'cluster'. Finally, we can see that all 3 models doesn't overfit the data as the testing error is smaller than the training error, which is quite unusual.

2.5 QDA model

(a) For QDA, we can redo the same computation that we derived for LDA. The log-likelihood for QDA can be written as follow:

$$\begin{aligned}\ell(\boldsymbol{\theta}) = N_1 log(\frac{\pi}{1-\pi}) + Nlog(1-\pi) - \frac{1}{2}\sum_{i=N_1}^{N}(x_i-\mu_0)^\intercal\Sigma_0^{-1}(x_i-\mu_0) \\ - \frac{1}{2}\sum_{i=1}^{N_1}(x_i-\mu_1)^\intercal\Sigma_1^{-1}(x_i-\mu_1) \\ - \frac{N}{2}log(2\pi) - \frac{(N-N_1)}{2}log|\Sigma_0| - \frac{(N_1)}{2}log|\Sigma_1|\end{aligned}$$

Then, as $y \sim Ber(\pi)$ we still have: $\pi = \frac{N_1}{N}$, also, according to the relation above the estimators for μ_0 and μ_1 doesn't change because here also, Σ_0^{-1} and Σ_1^{-1} are invertible. So, we have still: $\mu_0 = (1/N_0)\sum_{i=1}^{N}(1-y_i)x_i$ and $\mu_1 = (1/N_1)\sum_{i=1}^{N} y_i x_i$
Moreover, a similar calculation as the one we derived for LDA gives us Σ_0 and Σ_1, so finally the estimators for the QDA algorithm are:

Using relations (1), (2), (3) and (4), the maximum likelihood estimators are:

$$\begin{aligned}\pi &= \frac{N_1}{N} \\ \mu_0 &= \sum_{i=1}^{N}(1-y_i)x_i \\ \mu_1 &= \sum_{i=1}^{N} y_i x_i \\ \Sigma_0 &= \frac{1}{N_0}\sum_{i=N_1}^{N}(x-\mu_0)(x-\mu_0)^\intercal \\ \Sigma_1 &= \frac{1}{N_1}\sum_{i=1}^{N_1}(x-\mu_1)(x-\mu_1)^\intercal\end{aligned}$$

where:

$N_1 \triangleq \{\text{number of points belonging to class 1}\}$
$N_0 \triangleq \{\text{number of points belonging to class 0}\}$

We can also compute $p(y = 1|x)$:

$$p(y = 1|x) = \frac{1}{1 + exp\left(\underbrace{\frac{1}{2}((x-\mu_1)^\intercal \Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^\intercal \Sigma_0^{-1}(x-\mu_0))}_{f(\Sigma_1,\Sigma_0,\mu_1,\mu_0,x)}\right) \underbrace{\frac{1-\pi}{\pi}\frac{\left|\Sigma_1^{1/2}\right|}{\left|\Sigma_0^{1/2}\right|}}_{g(\Sigma_1,\Sigma_0,\pi)}}$$

and $f(\Sigma_1, \Sigma_0, \mu_1, \mu_0, x)$ can be rewritten:

$$f(\Sigma_1, \Sigma_0\mu_1, \mu_0, x) = \frac{1}{2}x^\intercal \underbrace{(\Sigma_1^{-1} - \Sigma_0^{-1})}_{-Q} x + \underbrace{(\mu_0\Sigma_0^{-1} - \mu_1\Sigma_1^{-1})}_{-\beta^\intercal} x + \frac{1}{2}(\mu_1^\intercal\Sigma_1^{-1}\mu_1 - \mu_0^\intercal\Sigma_0^{-1}\mu_0)$$

also, $g(\Sigma_1, \Sigma_0, \pi)$ can be rewritten:

$$g(\Sigma_1, \Sigma_0, \pi) = log\left[exp\left(\frac{1-\pi}{\pi}\frac{\left|\Sigma_1^{1/2}\right|}{\left|\Sigma_0^{1/2}\right|}\right)\right]$$

so finally, we have :

$$p(y = 1|x) = \frac{1}{1 + exp(-(0.5x^\intercal Qx + \beta^\intercal x + \gamma))} \triangleq \sigma(\frac{1}{2}x^\intercal Qx + \beta^\intercal x + \gamma)$$

where:

$$Q = -(\Sigma_1^{-1} - \Sigma_0^{-1})$$
$$\beta = -(\mu_0\Sigma_0^{-1} - \mu_1\Sigma_1^{-1})$$
$$\gamma = \frac{1}{2}\left(\mu_0^\intercal\Sigma^{-1}\mu_0 - \mu_1^\intercal\Sigma^{-1}\mu_1\right) - log\left(\frac{1-\pi}{\pi}\right) - \frac{1}{2}log\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right)$$

- For the **training dataset A** that contains 150 points, the values of the parameters are:
  - μ_0 = [2.89970947 -0.893874]
  - μ_1 = [-2.69232004 0.866042]
  - Σ_0 = [[2.31065259 -1.04748461][-1.04748461 0.57578403]]
  - Σ_1 = [[2.70442172 -1.3008515][-1.3008515 0.68969588]]
  - π = 0.33
  - Q = [[-1.51744039 -3.0272297][-3.0272297 -5.72332675]]
  - β = [-7.36527314 -10.87335416]

- γ = -0.626271453018

- For the **training dataset B** that contains 300 points, the values of the parameters are:
  - μ_0 = [3.34068896 -0.83546333]
  - μ_1 = [-3.21670734 1.08306733]
  - Σ_0 = [[2.53885859 1.0642112][1.0642112 2.96007891]]
  - Σ_1 = [[4.15361075 -1.33454097][-1.33454097 0.51607059]]
  - π = 0.5
  - Q = [[-0.95965255 -3.84765056][-3.84765056 -11.05867014]]
  - β = [-2.28065009 1.45700199]
  - γ = 3.366501856

- For the **training dataset C** that contains 400 points, the values of the parameters are:
  - μ_0 = [2.79304824 -0.83838667]
  - μ_1 = [-2.94232885 -0.9578284]
  - Σ_0 = [[2.89913927 1.24581553][1.24581553 2.92475448]]
  - Σ_1 = [[2.86914403 -1.76197061][-1.76197061 6.56438626]]
  - π = 0.625
  - Q = [[0.00488603 -0.29185968][-0.29185968 0.23611066]]
  - β = [-2.66524064 0.34888942]
  - γ = 0.110042748892

(b) $\rightarrow$ See graphs in Annexe 3.1

(c) Misclassification errors for LDA, Logistic regression, Linear regression and QDA

Algorithms	**LDA**	**Logistic regression**	**Linear regression**	**QDA**
Training set A	1.33	**0.0**	1.33	0.66
Testing set A	**2.0**	3.53	2.06	**2.00**
Training set B	3.0	2.0	3.0	**1.33**
Testing set B	4.15	4.3	4.15	**2.00**
Training set C	5.5	**4.0**	5.5	5.25
Testing set C	4.23	**2.26**	4.23	3.83

Table 2: Misclassification errors in percentage (%)

(d) As we could have expected, QDA always outperforms LDA on all training and testing set. It is normal, because QDA doesn't make the assumption that $\Sigma_1 = \Sigma_0$ contrary to LDA. So QDA is naturally more powerful than LDA, as LDA is a particular case of QDA. Moreover, as expected and highlighted in the performance comparison, QDA outperforms all the other linear classifiers on **Dataset B**, as the Dataset B is drawn from 2 Gaussians with different covariances which is exactly what QDA models.

Conclusion This homework was interesting. Yet, I spent too much time on the report (I'm quite new to LaTeX). I've copied and pasted all the parameters with great accuracy in the report so you can verify easily the correctness of the result. My code is far from being perfect. actually I've realized I should have used OOP paradigm and create functions such as train and predict (as it is done in scikit-learn).

3 Annexe

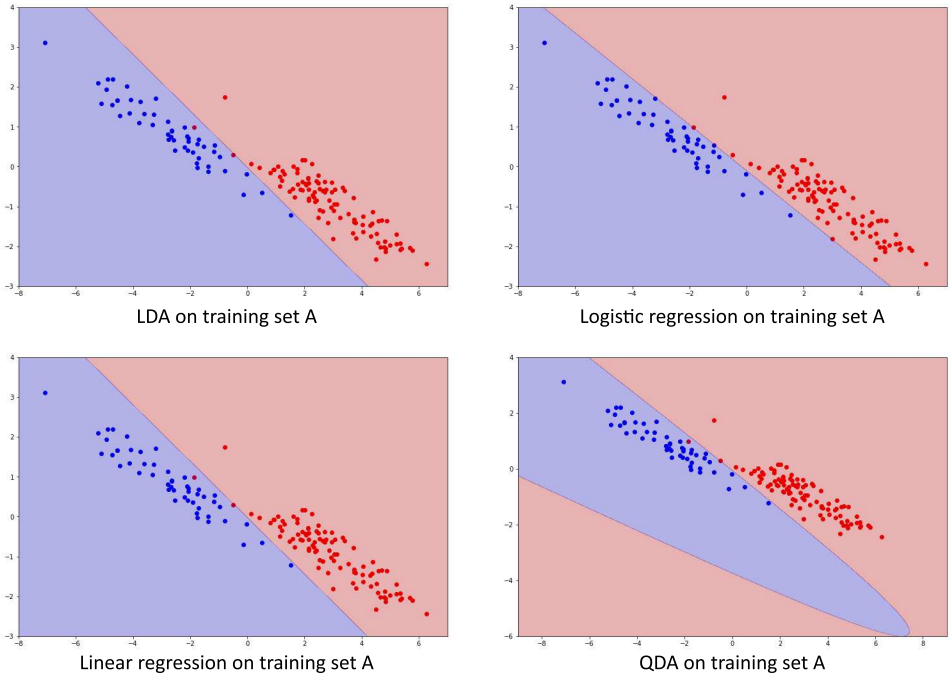

Figure 3.1: Decision boundary on training dataset A

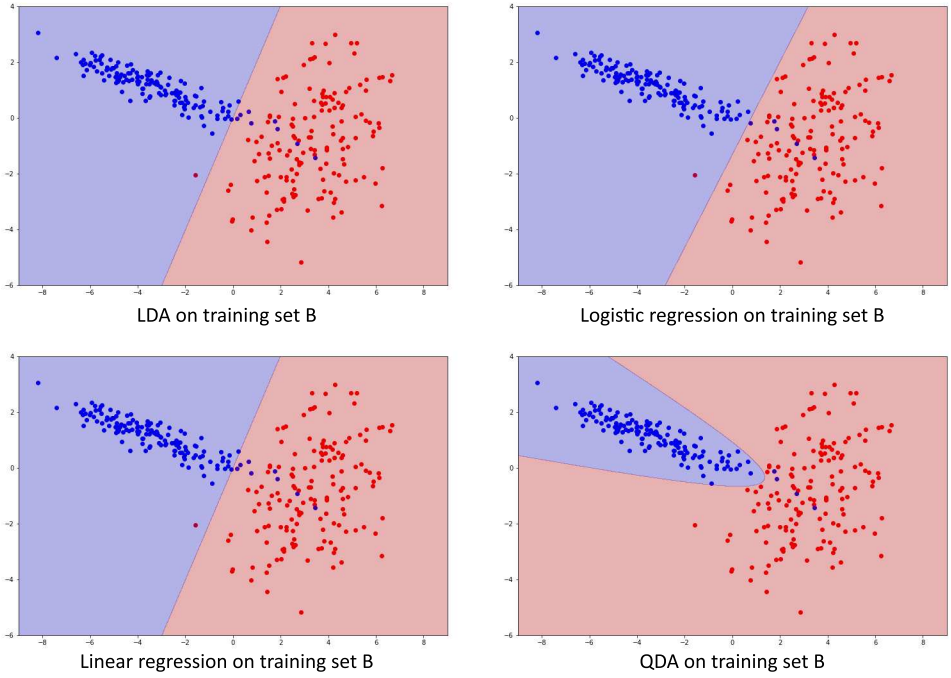

Figure 3.2: Decision boundary on training dataset B

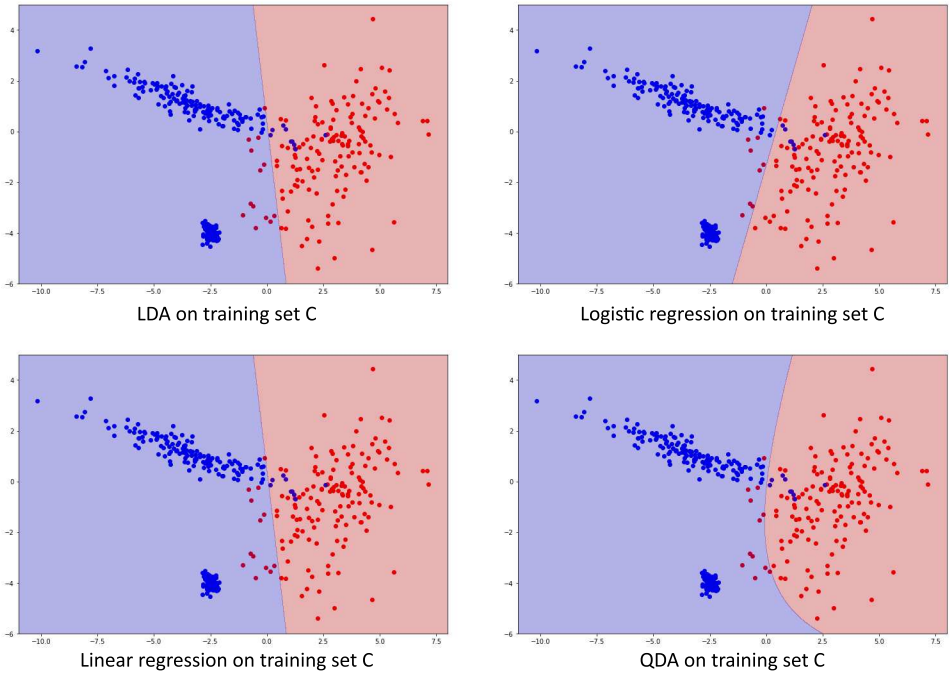

Figure 3.3: Decision boundary on training dataset C