MVA 2017/2018 - Graphical models - HWK 1 Report

Xiao CHU, Yifan WANG

October 2017

1 Learning in discrete graphical models

To estimate π and θ, we consider maximizing the likelyhood of the joint distribution of the two variables:

$$\begin{aligned}
l(\theta,\pi) &= log(\prod_{i=1}^{n}\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}p(x_i=k,z_i=m)) \\
&= \sum_{i=1}^{n} log(\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}\pi_m)
\end{aligned}$$

And we have the following optimization problem:

$$\begin{aligned}
max \quad & \sum_{i=1}^{n} log(\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}\pi_m) \\
\text{s.t.} \quad & \sum_{m=1}^{M}\pi_m = 1 \\
& \forall m = 1...M, \sum_{k=1}^{K}\theta_{mk} = 1
\end{aligned}$$

We write the Lagrangian:

$$\begin{aligned}
L(\theta,\pi,\lambda,\mu) &= \sum_{i=1}^{n} log(\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}\pi_m) \\
&+ \lambda(1-\sum_{m=1}^{M}\pi_m) + \sum_{m=1}^{M}\mu_m(1-\sum_{k=1}^{K}\theta_{mk})
\end{aligned}$$

We have

$$\begin{aligned}
\frac{\partial L}{\partial \theta_{mk}}(\theta,\pi,\lambda,\mu) &= \sum_{i=1}^{n}\frac{\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\pi_m}{\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}\pi_m} - \mu_m, \\
&= \frac{N_{mk}\pi_m}{\theta_{mk}\pi_m} - \mu_m, \forall m = 1...M, k = 1...K \\
\frac{\partial L}{\partial \pi_m}(\theta,\pi,\lambda,\mu) &= \sum_{i=1}^{n}\frac{\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}}{\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{1}_{z_i=m,x_i=k}\theta_{mk}\pi_m} - \lambda \\
&= \frac{N_{mk}\theta_{mk}}{\theta_{mk}\pi_m} - \lambda, \forall m = 1...M \\
\frac{\partial L}{\partial \lambda}(\theta,\pi,\lambda,\mu) &= 1 - \sum_{m=1}^{M}\pi_m \\
\frac{\partial L}{\partial \mu_m}(\theta,\pi,\lambda,\mu) &= 1 - \sum_{k=1}^{K}\theta_{mk}, \forall m = 1...M
\end{aligned}$$

Where N_{mk} is the number of samples verifying $Z_i = m, X_i = k$.

By setting $\frac{\partial L}{\partial \theta_{mk}}(\theta, \pi, \lambda, \mu) = 0$ and $\frac{\partial L}{\partial \mu_m}(\theta, \pi, \lambda, \mu) = 0$ for all $m = 1...M, k = 1...K$, we have $\mu_m = N_m$ where N_m is the number of samples verifying $Z_i = m$, thus we have $\boxed{\hat{\theta}_{mk} = \frac{N_{mk}}{N_m}}$.

By $\frac{\partial L}{\partial \pi_m}(\theta, \pi, \lambda, \mu) = 0$ and $\frac{\partial L}{\partial \lambda}(\theta, \pi, \lambda, \mu) = 0$, we have $\lambda = N_k$ where N_k is the number of samples verifying $X_i = k$, thus we have $\hat{\pi}_m = \frac{N_{mk}}{N_k}$. Theoretically we can estimate π_m with any value of k, but to make the estimation more robust, we choose to estimate it with $\boxed{\hat{\pi}_m = \frac{\sum_{k=1}^K N_{mk}}{\sum_{k=1}^K N_k} = \frac{N_m}{N}}$, where N is the number of all the samples.

2 Linear classification

2.1 Generative model (LDA)

(a) *Derive the form of the maximum likelihood estimator for this model.*

As $y \sim Bernoulli(\pi)$, we have:

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

Consider n observations $y_1, y_2, ..., y_n$, we have:

$$\begin{aligned} l(\pi) &= \sum_{i=1}^n logp(y_i, \pi) \\ &= \sum_{i=1}^n log(\pi^{y_i}(1-\pi)^{1-y_i}) \\ &= Nlog(\pi) + (n - N)log(1 - \pi) \end{aligned}$$

where $N = \sum_{i=1}^n y_i$

As $l(\pi)$ is strictly concave, it has a unique maximizer, and since the function is in addition differentiable, its maximizer $\hat{\pi}$ is the zero of its gradient $\nabla l(\pi)$:

$$\nabla l(\pi) = \frac{N}{\pi} - \frac{n - N}{1 - \pi}$$

Thus from $\nabla l(\hat{\pi}) = 0$, we have $\hat{\pi} = \frac{N}{n}$

Therefore,

$$\boxed{\hat{\pi} = \frac{N}{n} = \frac{\sum_{i=1}^n y_i}{n}}$$

.

As $x|y = i \sim Normal(\mu_i, \Sigma)$, we have:

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

Consider n observations $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, the log-likelihood is given by:

$$\begin{aligned} l(\mu_i, \Sigma) &= \sum_{j=1}^n 1_{y_j = i} log(\frac{1}{2\pi\sqrt{det(\Sigma)}} exp(\frac{-(x_j - \mu_i)^T \Sigma^{-1}(x_j - \mu_i)}{2})) \\ &= -(n_i log(2\pi) + \frac{n_i}{2} log(det(\Sigma)) + \frac{1}{2}\sum_{j=1}^n 1_{y_j=i}(x_j - \mu_i)^T \Sigma^{-1}(x_j - \mu_i)) \end{aligned}$$

where $n_i = \sum_{j=1}^n 1_{y_j=i}$ is the number of observations of $y_j = i$.

$l(\mu_i, \Sigma)$ is concave with respect to μ_i when Σ is fixed but is not even concave with respect to Σ when μ_i is fixed.

According to the results of differentiable function, we have:

$$\begin{aligned}\nabla_{\mu_i} l(\mu_i, \Sigma^{-1}) &= \sum_{j=1}^{n} \Sigma^{-1}(\mu_i - x_j)1_{y_j=i} \\ &= \Sigma^{-1}(n_i\mu_i - \sum_{j=1}^{n} x_j 1_{y_j=i}) \\ &= \Sigma^{-1} n_i(\mu_i - \bar{x_i})\end{aligned}$$

where $\bar{x_i} = \frac{1}{n_i}\sum_{j=1}^{n} x_j 1_{y_j=i}$ is the mean value of x whose $y_i = i$.
Thus from $\bigtriangledown_{\mu_i} l(\mu_i, \Sigma^{-1}) = 0$, we have:

$$\boxed{\hat{\mu_i} = \bar{x_i} = \frac{1}{\sum_{j=1}^{n} 1_{y_j=i}} \sum_{j=1}^{n} x_j 1_{y_j=i}}$$

Now we differentiate with respect to Σ^{-1}.
Suppose that $A = \Sigma_i^{-1}$, so we have:

$$l(\mu_i, \Sigma_i) = -(n_i log(2\pi) + \frac{n_i}{2} log(det(A)) + \frac{1}{2}\sum_{j=1}^{n} 1_{y_j=i}(x_j - \mu_i)^T A(x_j - \mu_i))$$

The last term is a real number, so it equal to its trace. Thus :

$$\begin{aligned}l(\mu_i, \Sigma_i) &= -(n_i log(2\pi) + \frac{n_i}{2} log(det(A)) + \frac{1}{2}\sum_{j=1}^{n} Trace(1_{y_j=i}(x_j - \mu_i)^T A(x_j - \mu_i))) \\ &= -(n_i log(2\pi) + \frac{n_i}{2} log(det(A)) + \frac{n_i}{2} Trace(A\tilde{\Sigma})\end{aligned}$$

where $\tilde{\Sigma} = \frac{1}{n_i}\sum_{j=1}^{n} 1_{y_j=i}(x_j - \mu_i)(x_j - \mu_i)^T$ is the empirical covariance matrix.
As $\nabla log(det(A)) = A^{-1}$, the gradient of with respect to A is :

$$\nabla_A(l) = -\frac{n_i}{2}A^{-1} + \frac{n_i}{2}\tilde{\Sigma}$$

Thus from $\bigtriangledown_A(l) = 0$, we have:

$$\hat{\Sigma} = \hat{\Sigma_i} = \tilde{\Sigma} = \frac{1}{n_i}\sum_{j=1}^{n} 1_{y_j=i}(x_j - \bar{x_i})(x_j - \bar{x_i})^T$$

when $\tilde{\Sigma}$ is invertible, where $\bar{x_i} = \frac{1}{n_i}\sum_{j=1}^{n} x_j 1_{y_j=i}$ is the mean value of x whose $y_i = i$.

Here, as we have $\Sigma = \Sigma_0 = \Sigma_1$, we take the weighted average of $\hat{\Sigma_0}$ and $\hat{\Sigma_1}$ to compute the $\hat{\Sigma}$. So we have:

$$\boxed{\hat{\Sigma} = \sum_{i=0}^{1} \frac{n_i}{n}\hat{\Sigma_i} = \frac{1}{n}\sum_{i=0}^{1}\sum_{j=1}^{n} 1_{y_j=i}(x_j - \bar{x_i})(x_j - \bar{x_i})^T}$$

Remark: It is easy to prove that the Maximum Likelyhood Estimator for covariance matrix is biased. If we would like it to be unbiased, we need to modify it, replace n by $n - 1$.

(b) *What is the form of the conditional distribution p(y = 1—x) ? Compare with the form of logistic regression.*

According to the Bayes Rule, we know that

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Besides,

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

Thus we have:

$$\begin{aligned}
P(y=1|x) &= \frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1)+P(x|y=0)P(y=0)} \\
&= \frac{1}{1+\frac{P(y=0)}{P(y=1)}\frac{P(x|y=0)}{P(x|y=1)}} \\
&= \frac{1}{1+\frac{1-\pi}{\pi}exp(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)+\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1))} \\
&= \frac{1}{1+\frac{1-\pi}{\pi}exp(-\frac{1}{2}x^T\Sigma^{-1}x+\frac{1}{2}x^T\Sigma^{-1}x+\mu_0^T\Sigma^{-1}x-\mu_1^T\Sigma^{-1}x-\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0+\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1)} \\
&= \frac{1}{1+\frac{1-\pi}{\pi}exp(-(\Sigma^{-1}(\mu_1-\mu_0))^Tx+\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1)))} \\
&= \boxed{\frac{1}{1+\frac{1-\pi}{\pi}exp(-(w^Tx+b))}}
\end{aligned}$$

where $w=\Sigma^{-1}(\mu_1-\mu_0)$ and $b=\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1$.

Comparison with the form of logistic regression:

For logistic regression, we have:

$$P(y=1|x)=\sigma(w^Tx),\ where\ \sigma(z)=\frac{1}{1+exp(-z)}$$

For LDA, as we deduce above, we have:

$$P(y=1|x)=\frac{1}{1+\frac{1-\pi}{\pi}exp(-(w^Tx+b))}$$

So when $\pi=0.5$, these two $P(y=1|x)$ are in the same form.

For both logistic regression and LDA, the decision boundary defined by the equation $p(y=1|x)=0.5$ is a line.

(c)*Implement the MLE for this model and apply it to the data. Represent graphically the data as a point cloud in R2 and the line defined by the equation p(y = 1—x) = 0.5*

After appling the MLE for this model, we have:

For classificationA training data:

π=0.333333333333,

μ_0=(2.89970947, -0.893874), μ_1= (-2.69232004, 0.866042)

$$\Sigma=\begin{bmatrix}2.44190897 & -1.13194024\\ -1.13194024 & 0.61375465\end{bmatrix}$$

For classificationB training data:

π=0.5,

μ_0=(3.34068896, -0.83546333), μ_1= (-3.21670734, 1.08306733)

$$\Sigma_0=\begin{bmatrix}3.34623467 & -0.13516489\\ -0.13516489 & 1.73807475\end{bmatrix}$$

For classificationC training data:

π=0.625,

μ_0=(2.79304824, -0.83838667), μ_1= (-2.94232885, -0.9578284)

$$\Sigma_0=\begin{bmatrix}2.88039225 & -0.63405081\\ -0.63405081 & 5.19952435\end{bmatrix}$$

For $p(y=1|x)=0.5$, we should have:

$$\frac{1-\pi}{\pi}exp(-(\Sigma^{-1}(\mu_1-\mu_0))^Tx+\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1)))=1$$

which means that x satisfies that

$$w^Tx+b=-log(\frac{\pi}{1-\pi})$$

where $w = \Sigma^{-1}(\mu_1 - \mu_0)$ and $b = \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1$.

So according to this equation, we can get the line that we want.

We implement the MLE for this model and apply it respectively for different training data. Results that we get are showed in the figure 1.

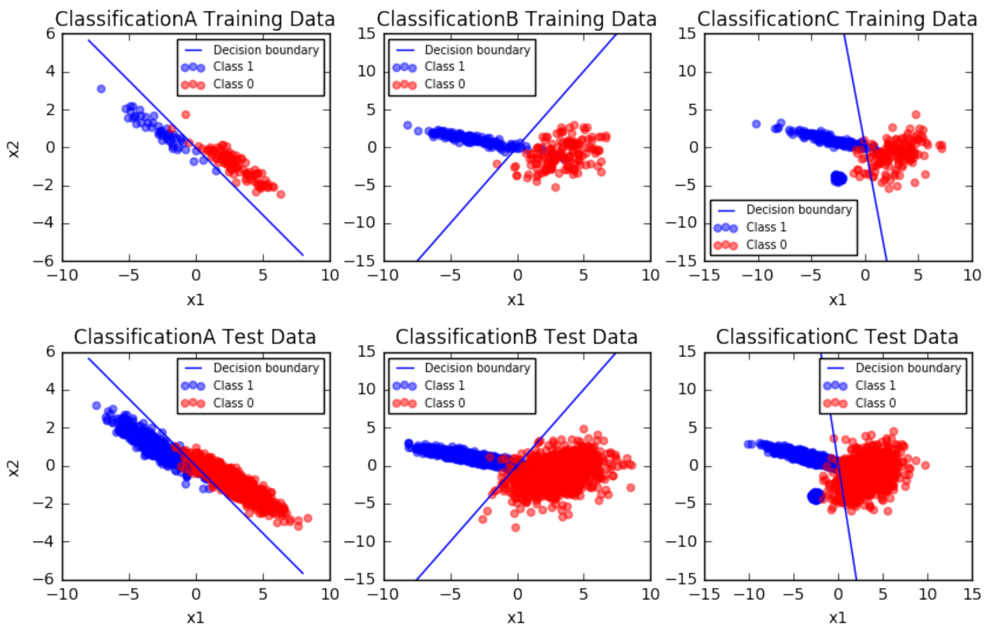

Figure 1: Training data, test data and decision boundary for LDA

From this figure, we can observe that the decision boundary for LDA is a line. It seems that LDA classifier works better on the dataset A and B compared with dataset C as the decision boundary separates well the data in test set. This may because that dataset C doesn't satisfy the assumption of LDA (LDA supposes that $\Sigma_0 = \Sigma_1$) and points in class 1 seems not follow exactly a gaussian distribution.

2.2 Logistic regression

We write the log-likelyhood that we want to maximize for logistic regression.

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

Note that in our implementation, the affine function $\boldsymbol{w}^T\boldsymbol{x} + b$ is written in form of $\boldsymbol{W}^T\tilde{x}$ with $\boldsymbol{W} = [\begin{smallmatrix}\boldsymbol{w}\\ b\end{smallmatrix}]$ and $\tilde{\boldsymbol{x}} = [\begin{smallmatrix}\boldsymbol{x}\\ 1\end{smallmatrix}]$. We will use this notation in the rest of this section. The log likelyhood, its gradient w.r.t. $\boldsymbol{W}$, and the Hessian matrix are:

$$l(\boldsymbol{W}) = \sum_{i=1}^{n} y_i log(\sigma(\boldsymbol{W}^T\tilde{\boldsymbol{x}}_{\boldsymbol{i}})) + (1 - y_i)log(\sigma(-\boldsymbol{W}^T\tilde{\boldsymbol{x}}_{\boldsymbol{i}}))$$

$$\nabla_{\boldsymbol{W}} l(\boldsymbol{W}) = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{\eta})$$

$$Hl(\boldsymbol{W}) = -\boldsymbol{X}^T Diag(\eta_i(1 - \eta_i))\boldsymbol{X}$$

where $\boldsymbol{X} = \begin{bmatrix}\tilde{\boldsymbol{x}}_{\boldsymbol{1}}^T\\ \dots\\ \tilde{\boldsymbol{x}}_{\boldsymbol{n}}^T\end{bmatrix}$ is the design matrix, $\eta_i = \sigma(\boldsymbol{W}^T\tilde{\boldsymbol{x}}_{\boldsymbol{i}})$, $\boldsymbol{\eta} = \begin{bmatrix}\eta_1\\ \dots\\ \eta_n\end{bmatrix}$.

We apply IRLS algorithm to iteratively maximize $l(\boldsymbol{W})$ by setting $\boldsymbol{W}^{t+1} = \boldsymbol{W}^t - \epsilon^t(Hl(\boldsymbol{W}^t))^{-1}\nabla_{\boldsymbol{W}^t} l(\boldsymbol{W})$ where ϵ^t is set with Armijo Line Search as described below. The algorithm will stop when the norm of update of $\boldsymbol{W}$ is small enough. As for parameter initialization, we initialize $\boldsymbol{W}^0$ to zeros.

Algorithm 1: Armijo Line Search for Linear Regression

```
Hyper parameters: Decay rate $a$
$\boldsymbol{d} = (Hl(\boldsymbol{W^t}))^{-1}\nabla_{\boldsymbol{W}^t} l(\boldsymbol{W}))$
while $l(\boldsymbol{W}^t + \epsilon \boldsymbol{d}) < l(\boldsymbol{W}^t) + \frac{1}{2}\epsilon \nabla_{\boldsymbol{W}^t} l(\boldsymbol{W}^t))^T \boldsymbol{d}$ do
    $\epsilon = a\epsilon$
end
$\epsilon^t = \epsilon$
Return: $\epsilon^t$
```

(a) The numerical values of the parameters learnt for ClassificationA dataset:

$$\boldsymbol{w} = [-1019.4784, -1766.5203]^T$$
$$b = -168.8184$$

We scale them by dividing them by b and we get:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [6.0389, 10.4640]^T\\ b &= 1.0\end{aligned}}$$

ClassificationB dataset:

$$\boldsymbol{w} = [-1.7051, 1.0237]^T$$
$$b = 1.34959153$$

after scaling:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [-1.2635, 0.7586]^T\\ b &= 1.0\end{aligned}}$$

ClassificationC dataset:

$$\boldsymbol{w} = [-2.2032, 0.7093]^T$$
$$b = 0.95918885$$

after scaling:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [-2.2970, 0.7394]^T\\ b &= 1.0\end{aligned}}$$

The results might be slightly different due to different parameter initialization.

(b) We present the data and the decision boundary in $\mathbb{R}^2$. The points on the decision boundary satisfies $p(y = 1|x) = 0.5$, which gives us:

$$p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) = \frac{1}{1 + exp(-\boldsymbol{w}^T\boldsymbol{x} - b)} = \frac{1}{2}$$
$$exp(-\boldsymbol{w}^T\boldsymbol{x} - b) = 1$$

Finally we find that the decision boundary is an affine function.

$$\boldsymbol{w}^T\boldsymbol{x} = b$$

Intuitively, we notice that the linear regression classifier works better on the last two datasets than LDA classifier.

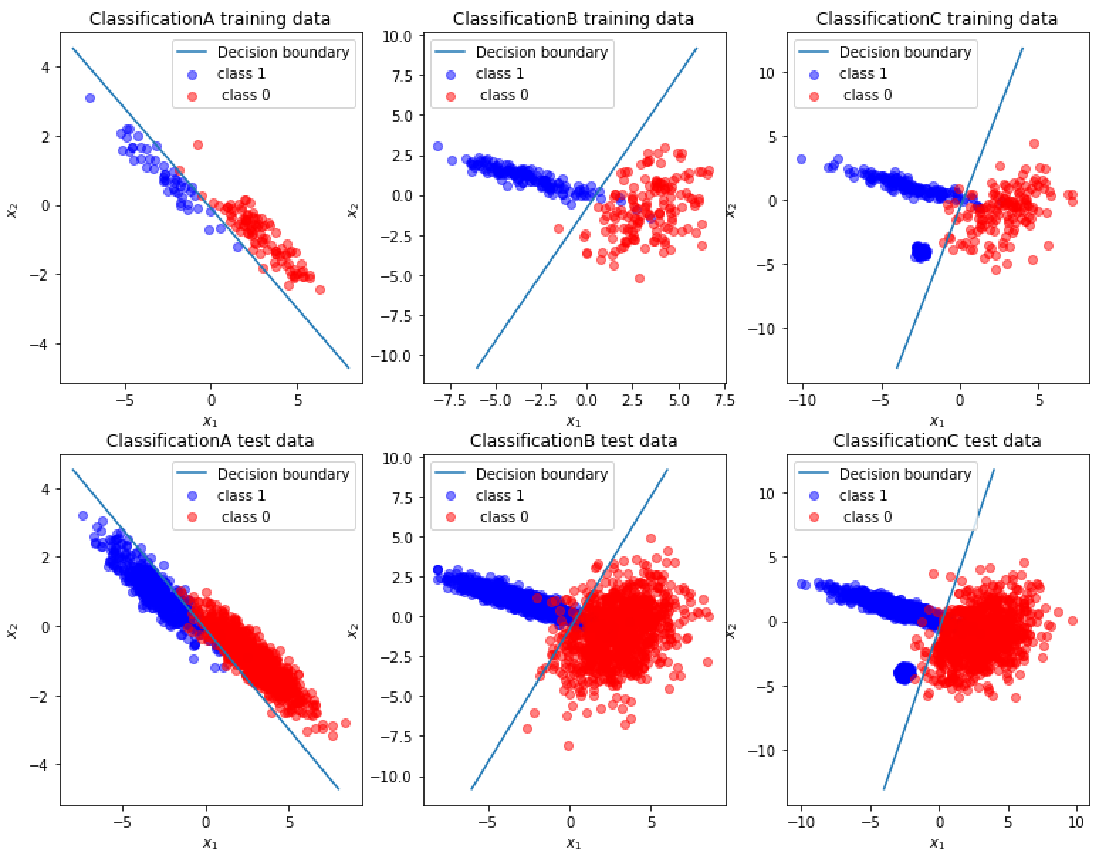

Figure 2: Training data, test data and decision boundary for logistic regression

2.3 Linear regression

To solve the linear regression problem, we need to solve the normal equation:

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} = \boldsymbol{X}^T\boldsymbol{y}$$

Here we also include the constant term in $\boldsymbol{W}$. If $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, which is the case for our datasets, we have the unique solution given by $\hat{\boldsymbol{W}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

(a) The numerical values of the parameters learnt for
ClassificationA dataset:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [-0.2640, -0.3726]^T \\ b &= 0.49229204\end{aligned}}$$

ClassificationB dataset:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [-0.1042, 0.0518]^T \\ b &= 0.50005043\end{aligned}}$$

ClassificationC dataset:

$$\boxed{\begin{aligned}\boldsymbol{w} &= [-0.1277, -0.0170]^T \\ b &= 0.50839982\end{aligned}}$$

(b) We present the data and the decision boundary in $\mathbb{R}^2$.

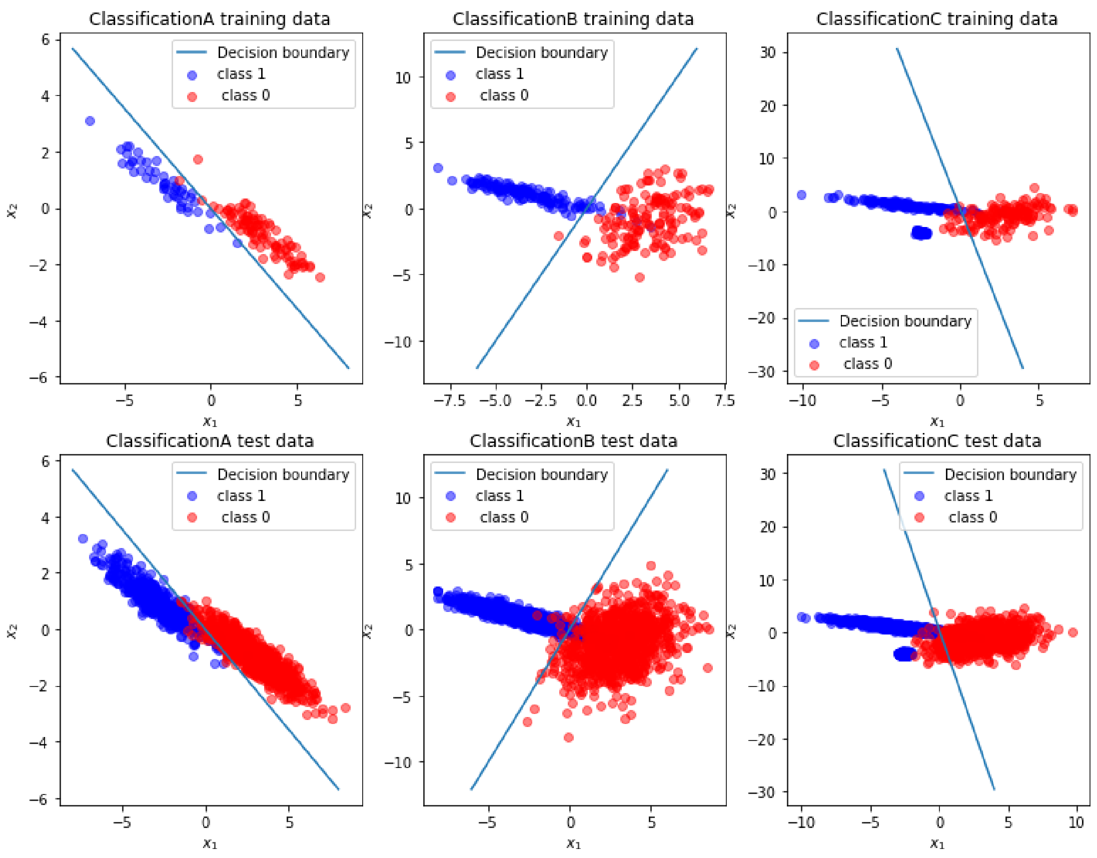

Figure 3: Training data, test data and decision boundary for linear regression

2.4 Comparison of different methods

(a) *Compute for each model the misclassification error on the training data and compute it as well on the test data*

	Training Error			Test Error		
	A	B	C	A	B	C
LDA	0.0133	0.03	0.055	**0.02**	**0.0415**	0.0423
Logistic Regression	0	0.02	0.04	0.034	0.043	**0.0227**
Linear Regression	0.0133	0.03	0.055	0.0206	**0.0415**	0.0423

(b) *Compare the performances of the different methods on the three datasets*

For dataset ClassificationA and ClassificationB, the misclassification error of the three classifiers are larger on test set than on training set, however, for ClassificationB dataset, the test errors are smaller than the training errors.

The reason why test error is higher than the training error on the first two datasets is that the classifier learns from the training data and try to minimize the error on training data, thus it might not be able to generalize well on test set. In our case, since the test set and training set are drawn from the same distribution, so the classifiers generalize quite well.

It seems that the logistic regression classifier yieds quite dissimilar results for training and test sets. LDA seems to have the best performance for these three datasets since it has the lowest error in two datasets.

For ClassificationA dataset, the three classifiers all work quite well. This might be because that the data is roughly linearly separable and the two classes have similar covariance, which is important for LDA.

For ClassificationB dataset, the two classes have a small overlap, which might influence the performance of classifiers. Moreover, here the covariance matrix of the two classes are quite different, which might influence the performance of LDA.

For ClassificationC dataset, we have a small dense cluster of points for class 1 besides the strip of points. From the previous figures, we find that this dense cluster is more important for the linear regression classifier since the decision boundary of linear regression for dataset A and B are quite different, while quite similar in the case of logistic regression. The linear regression classifier tries to keep the dense cluster far from the decision boundary, even if it will misclassify some class 0 points.

2.5 QDA model

In this case, we have:

$$x|y = i \sim Normal(\mu_i, \Sigma_i)$$

So compared with the LDA model, the maximum likelihood estimator of π, μ_0 and μ_1 is the same. For Σ_i, the method to get its MLE is the same. So we can easily get that:

$$\hat{\pi} = \frac{N}{n} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$\hat{\mu_i} = \bar{x_i} = \frac{1}{\sum_{j=1}^{n} 1_{y_j=i}} \sum_{j=1}^{n} x_j 1_{y_j=i}$$

$$\hat{\Sigma_i} = \frac{1}{n_i} \sum_{j=1}^{n} 1_{y_j=i}(x_j - \bar{x_i})(x_j - \bar{x_i})^T$$

(a)*Provide the numerical values of the learnt parameters.*

For classificationA training data:
π = 0.3333,
μ_0=(2.8997, -0.8939), μ_1= (-2.6923, 0.8660)
$\Sigma_0 = \begin{bmatrix} 2.3107 & -1.0475 \\ -1.0475 & 0.5758 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2.7044 & -1.3009 \\ -1.3009 & 0.6897 \end{bmatrix}$

For classificationB training data:
π=0.5,
μ_0=(3.3407, -0.8355), μ_1= (-3.2167, 1.0831)
$\Sigma_0 = \begin{bmatrix} 2.5389 & 1.0642 \\ 1.0642 & 2.9601 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 4.1536 & -1.3345 \\ -1.3345 & 0.5161 \end{bmatrix}$

For classificationC training data:
π=0.625,
μ_0=(2.7930, -0.8384), μ_1= (-2.9423, -0.9578)
$\Sigma_0 = \begin{bmatrix} 2.8991 & 1.2458 \\ 1.2458 & 2.9248 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2.8691 & -1.7620 \\ -1.7620 & 6.5644 \end{bmatrix}$

(b)*Represent graphically the data as well as the conic defined by p(y = 1—x) = 0.5*
As we have:

$$\begin{aligned} P(y=1|x) &= \frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1) + P(x|y=0)P(y=0)} \\ &= \frac{1}{1 + \frac{P(y=0)}{P(y=1)}\frac{P(x|y=0)}{P(x|y=1)}} \\ &= \frac{1}{1 + \frac{1-\pi}{\pi}\frac{\sqrt{det(\Sigma_1)}}{\sqrt{det(\Sigma_0)}} exp(-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1))} \end{aligned}$$

So we can define the predict probability according to this function and then we can draw the conic defined by $p(y = 1|x) = 0.5$ by using the matplotlib.pyplot.contour.

Results that we get are showed in the figure 4:

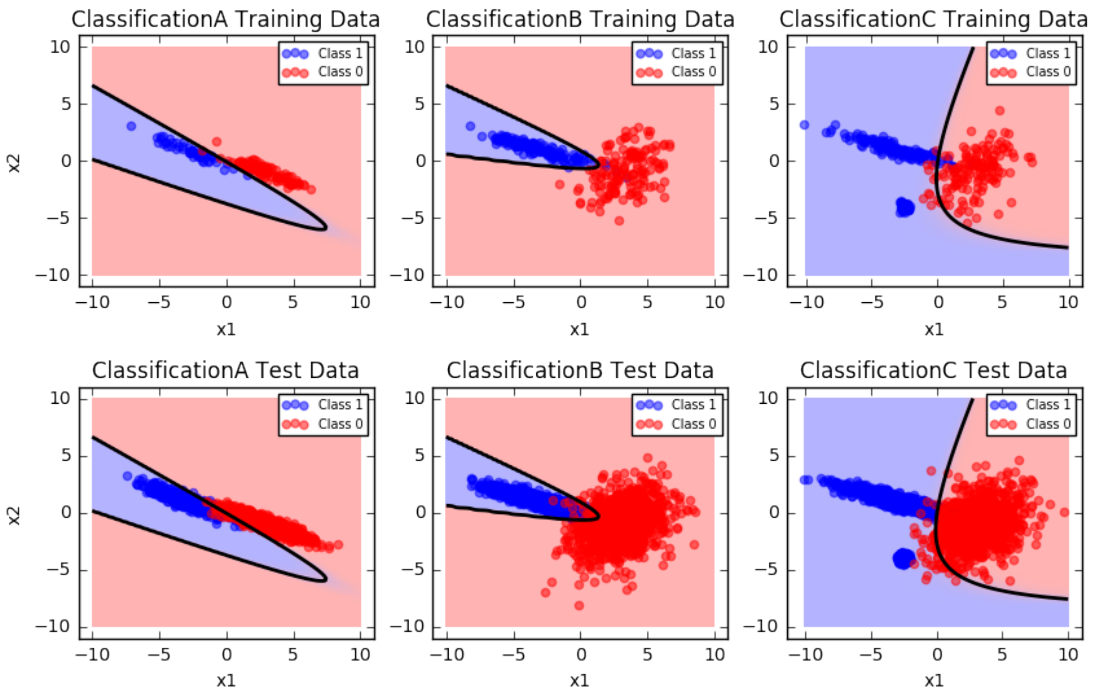

Figure 4: Training data, test data and decision boundary for QDA

(c) *Compute the misclassification error for QDA for both train and test data.*

The misclassification error is showed in the table below:

Misclassification Error for QDA	A	B	C
Training Data	0.0067	0.0133	0.0525
Test Data	0.02	0.02	0.0383

(d) *Comment the results as previously*

From the table in the question 2.5.c, we can see that the QDA classifier has a good performance on all datasets, especially on A and B.

Compared with LDA classifier, the QDA classifier has a lower misclassification error thus a better score for both train and test data, which is very reasonable. LDA supposes that $\Sigma_0 = \Sigma_1$, however this assumption is not satisfied by all datasets as we can easily observe from the data which is represented graphically. QDA doesn't have this assumption and that's why it has a better performance.