As usual, please hand in on paper form your derivations and answers to the questions. You can use any programming language for your source code (submitted on Studium as per the website instructions). All the requested figures should be printed on paper with clear titles that indicate what the figures represent.

Linear classification

Please download the datasets from the website. You get 6 text files: three training sets (`.train` files) and three test sets (`.test` files). Each row of the text file represents a sample of data (x_i, y_i). There are three columns: the first two give the coordinates for $x_i \in \mathbb{R}^2$; the third column gives the class label $y_i \in \{0, 1\}$. There are three different types of datasets (`A`, `B` and `C`), all generated from some kind of mixture of Gaussians generative model. The train and test sets are generated from the same distribution for each types of dataset. To help your interpretation, we give you the actual generating process:

- Dataset `A`: the class-conditionals for this dataset are Gaussians with different means, but with a *shared* covariance matrix Σ.
- Dataset `B`: similar generating process but the covariance matrices are *different* for the two classes.
- Dataset `C`: here one class is a mixture of two Gaussians, while the other class is a single Gaussian (with no sharing).

Note that normally we would not know the information about the generating process. In this assignment, we will compare difference classification approaches.

1. **Generative model (Fisher LDA)**. We first consider the Fisher LDA model as seen in class: given the class variable, the data are assumed to be Gaussians with different means for different classes but with the same covariance matrix:

$$Y \sim \text{Bernoulli}(\pi), \quad X \mid Y = j \sim \mathcal{N}(\mu_j, \Sigma).$$

   (a) Derive the form of the maximum likelihood estimator for this model. *Hint*: you can re-use some of the tricks presented in class for the MLE of a multivariate Gaussian, but adapted to this setting. You can get inspiration from Section 7.2 in Mike's book (which covers the case where Σ is diagonal).

   Answer:

   We have $\theta = (\pi, \mu_1, \mu_0, \Sigma)$. Thus the likelihood function is given by (D is the data (X,Y)):

$$\begin{aligned}
L(\theta, D) &= \prod_{i=1}^{n} p(x_i|y_i,\theta)p(y_i|\pi) = \prod_{i=1}^{n} p(y_i|\pi)p(x_i|y_i,\theta) \\
&= \prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i} \cdot (\frac{1}{(2\pi|\Sigma|)^{\frac{1}{2}}} \exp\{\frac{-1}{2}(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)\})^{y_i} \\
&\cdot (\frac{1}{(2\pi|\Sigma|)^{\frac{1}{2}}} \exp\{\frac{-1}{2}(x_i-\mu_0)^T\Sigma^{-1}(x_i-\mu_0)\})^{1-y_i}
\end{aligned}$$

and the log-likelihood is thus :

$$\begin{aligned}
l(\theta, D) &= \sum_{i=1}^{n} \log(p(y_i|\pi)p(x_i|y_i,\theta)) \\
&= \sum_{i=1}^{n} y_i \log(\pi) + \sum_{i=1}^{n} (1-y_i)\log(1-\pi) + \sum_{i=1}^{n} \frac{-yi}{2}(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) \\
&+ \sum_{i=1}^{n} (\frac{-(1-yi)}{2}(x_i-\mu_0)^T\Sigma^{-1}(x_i-\mu_0)) \cdot -\frac{n}{2}\log(2\pi) \cdot -\frac{n}{2}\log|\Sigma|
\end{aligned}$$

Given the log-likelihood, we find the MLE for each parameter in θ by maximizing $l(\theta, D)$ with respect to these parameters.

$\hat{\pi}_{MLE}$:

$$\begin{aligned}
\frac{dl}{d\pi} &= \frac{\sum_{i=1}^{n} y_i}{\pi} - \frac{\sum_{i=1}^{n}(1-y_i)}{1-\pi} = \frac{(1-\pi)\sum_{i=1}^{n} y_i - \pi\sum_{i=1}^{n} 1-y_i}{\pi(1-\pi)} \\
&= \frac{\sum_{i=1}^{n} y_i - n\pi}{\pi(1-\pi)}
\end{aligned}$$

And setting equal to 0 gives us :

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

This is the same MLE as seen in class, and is indeed a maximum since the second derivative of $l(\theta, D)$ w.r.t. π is clearly < 0 along its domain, so the function is concave.

$\hat{\mu_1}_{MLE}$:

$$\begin{aligned}
\nabla l_{\mu_1} &= \nabla_{\mu_1} \frac{-1}{2} \sum_{i=1}^{n} y_i(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) && (1) \\
&= \sum_{i=1}^{n} \Sigma^{-1} y_i(x_i-\mu_1) && (2)
\end{aligned}$$

Where (2) was obtained directly following the derivation tricks shown in class for multivariate gaussians. Now setting this equal to 0 and ignoring the constant matrix we get:

$$\sum_{i=1}^{n} y_i(x_i-\mu_1)=0 \implies \sum_{i=1}^{n} y_i x_i = \sum_{i=1}^{n} y_i\mu_1 \implies \hat{\mu_1}_{MLE} = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} y_i}$$

We obtain the MLE for μ_0 with an almost equivalent derivation.

$\hat{\mu_0}_{MLE}$:

$$\begin{aligned}\nabla l_{\mu_0} &= \nabla_{\mu_0}\frac{-1}{2}\sum_{i=1}^{n}(1-y_i)(x_i-\mu_0)^T\Sigma^{-1}(x_i-\mu_0)\\ &= \sum_{i=1}^{n}\Sigma^{-1}(1-y_i)(x_i-\mu_0)\end{aligned}$$

Setting to 0 gives:

$$\sum_{i=1}^{n}(1-y_i)(x_i-\mu_0)=0 \implies \sum_{i=1}^{n}(1-y_i)x_i = \sum_{i=1}^{n}(1-y_i)\mu_0 \implies \hat{\mu_0}_{MLE} = \frac{\sum_{i=1}^{n}(1-y_i)x_i}{\sum_{i=1}^{n} 1-y_i}$$

These MLEs are indeed intuitive as they correspond to the MLE for μ in a multivariate gaussian within the points of their respective classes y_i. We now turn our attention to the MLE for Σ. Since the MLE is invariant, we can equivalently find the MLE for Σ^{-1}.

$\hat{\Sigma}^{-1}_{MLE}$:

$$\nabla_{\Sigma^{-1}} l = \nabla_{\Sigma^{-1}}\{\frac{n}{2}\log|\Sigma^{-1}| - \frac{n_1}{2}\langle\Sigma^{-1},\tilde{\Sigma}_1\rangle - \frac{n_0}{2}\langle\Sigma^{-1},\tilde{\Sigma}_0\rangle\} \tag{3}$$

Where $\tilde{\Sigma}_1 = \frac{\sum_{i=1}^{n} y_i(x_i-u_1)(x_i-u_1)^T}{n_1}, \tilde{\Sigma}_0 = \frac{\sum_{i=1}^{n}(1-y_i)(x_i-u_0)(x_i-u_0)^T}{n_0}, n_1 = \sum_{i=1}^{n} y_i, n_0 = \sum_{i=1}^{n}(1-y_i)$, as was seen in class using the trace trick. The only difference here is that we added the $y_i, 1-y_i$ terms, however these do not affect the derivation as they are simply scalar multiples. Thus, using the derived gradient for the log determinant from class, the overall gradient is:

$$\nabla_{\Sigma^{-1}} l = \frac{n}{2}\Sigma - \frac{n_1}{2}\tilde{\Sigma}_1 - \frac{n_0}{2}\tilde{\Sigma}_0$$

It should be noted that the coefficients of the $\tilde{\Sigma}$ terms are n_1, n_0 to account for the divisions in these terms, thus it is equivalent to the relevant terms in the log-likelihood. Setting to 0, we get:

$$\begin{aligned}\hat{\Sigma}_{MLE} &= \frac{n_1\tilde{\Sigma}_1 + n_0\tilde{\Sigma}_0}{n}\\ &= \frac{\sum_{i=1}^{n} y_i(x_i-u_1)(x_i-u_1)^T + \sum_{i=1}^{n}(1-y_i)(x_i-u_0)(x_i-u_0)^T}{n}\end{aligned}$$

This MLE corresponds to a pooling of the variances for the respective classes.

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

Answer:

Using Bayes' rule, we obtain the conditional distribution as follows:

$$p(y=1|x) = \frac{p(x|y=1,\theta)p(y=1)}{p(x)} \tag{4}$$

$$= \frac{\exp\{\frac{-1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}\pi}{\pi\exp\{\frac{-1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\} + (1-\pi)\exp\{\frac{-1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}} \tag{5}$$

$$= \frac{1}{1+\frac{(1-\pi)\exp\{\frac{-1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}}{\pi\exp\{\frac{-1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}}} \tag{6}$$

$$= \frac{1}{1+\exp\{-\log(\frac{\pi\exp\{\frac{-1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}}{(1-\pi)\exp\{\frac{-1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}}+)\}} \tag{7}$$

$$= \frac{1}{1+\exp\{-\log(\frac{\pi}{1-\pi})+\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}} \tag{8}$$

$$= \frac{1}{1+\exp\{-(\mu_1-\mu_0)^T\Sigma^{-1}x+\frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0)-\log(\frac{\pi}{1-\pi})\}} \tag{9}$$

Where equation 9 was re-derived but inspired by the analogous simplification in Mike Jordan's book (section 7.2).

Now setting $w = \Sigma^{-1}(\mu_1 - \mu_0)$ and $b = \frac{-1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0) + \log(\frac{\pi}{1-\pi})$, we can see that the above conditional distribution has the form:

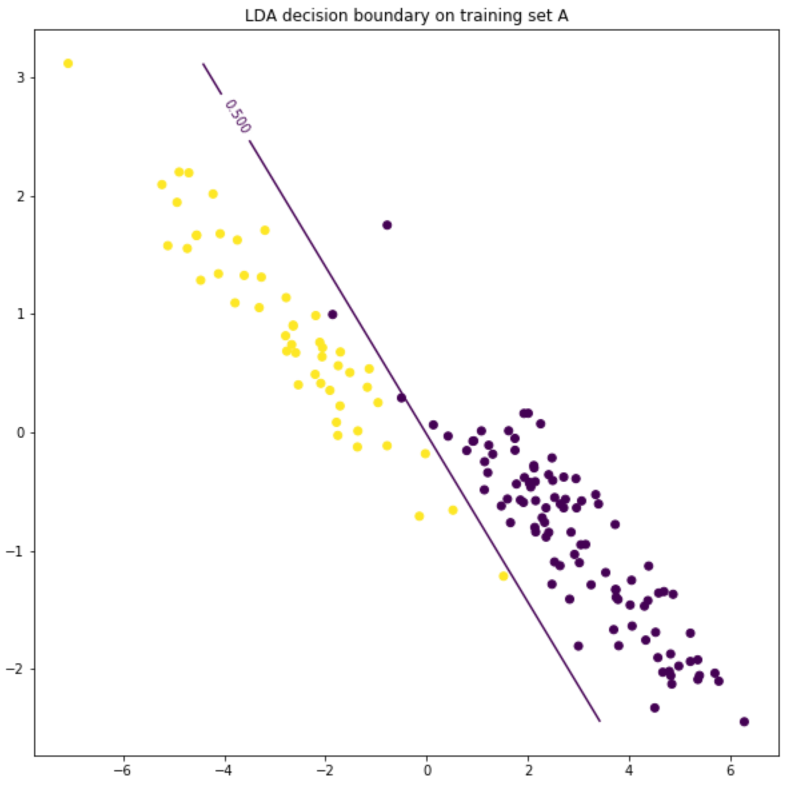
$$\frac{1}{1+\exp\{-w^Tx-b\}} = \sigma(w^Tx+b)$$

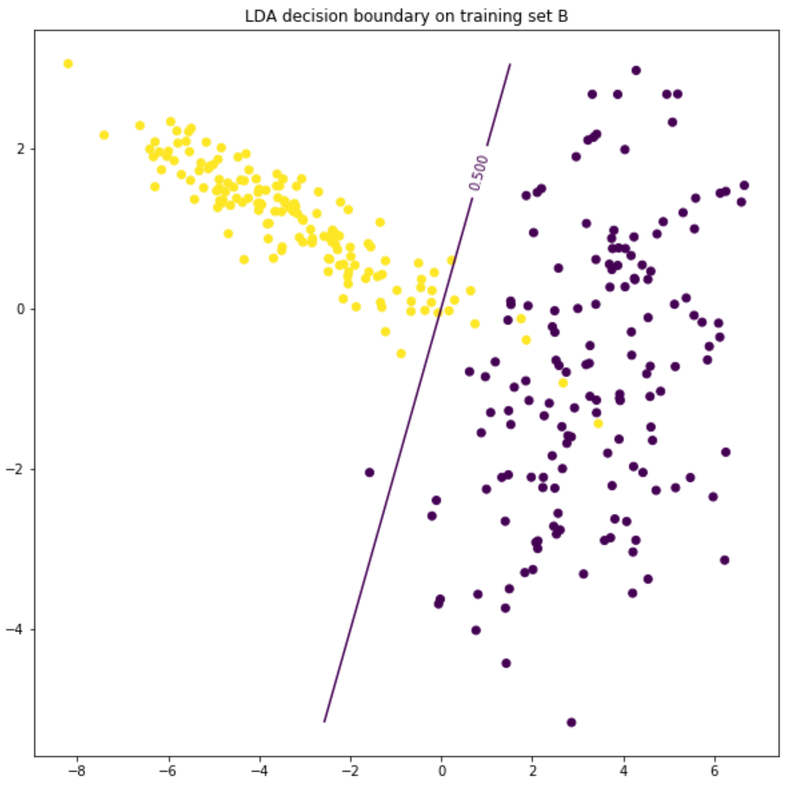
which indeed corresponds with the form of logistic regression, and makes evident the fact that the learned parameters act linearly upon the training data to produce a decision boundary. Clearly, the distinction with logistic regression comes from the fact that for Fisher LDA we are enforcing a gaussian assumption on the class-conditional likelihood of the data.

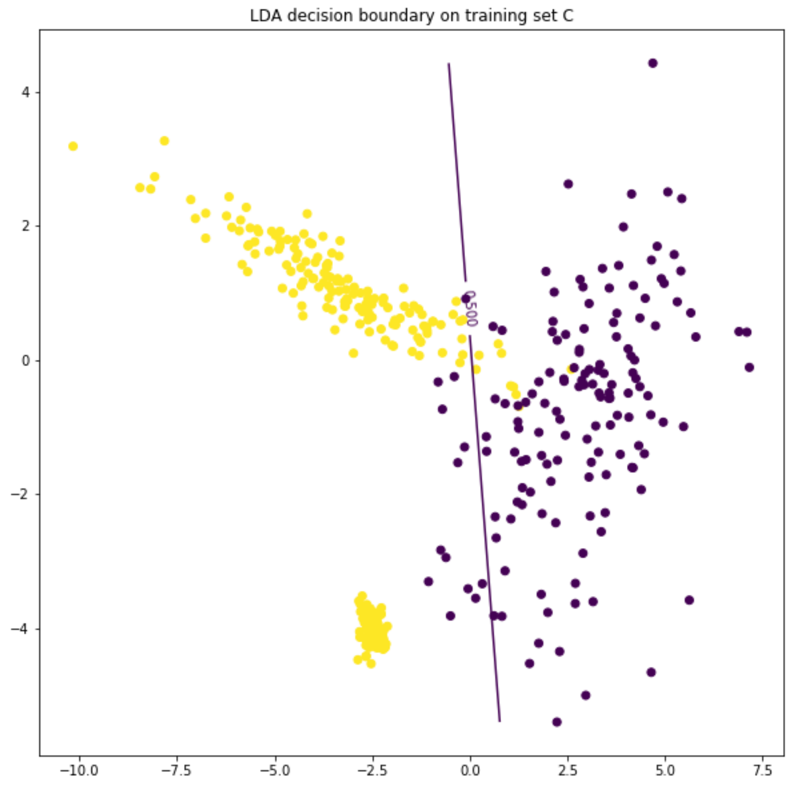
(c) Implement the MLE for this model and apply it to each training dataset. For each dataset, represent graphically the data as a point cloud in $\mathbb{R}^2$ and the line defined by the equation

$$p(y=1|x) = 0.5\,.$$

Answer: Refer to the Jupyter notebook for the relevant code (here and for all questions). The resulting figures are presented here :


LDA decision boundary on training set A
0.500


LDA decision boundary on training set B
0.500



2. **Logistic regression:** now implement logistic regression for an affine function $f(x) = w^\top x + b$ (do not forget the constant term – you can use the bias feature trick) using the IRLS algorithm (Newton's method) which was described in class. Hint: never compute the matrix inverse by itself – this is not numerically stable when the Hessian might become ill-conditioned...

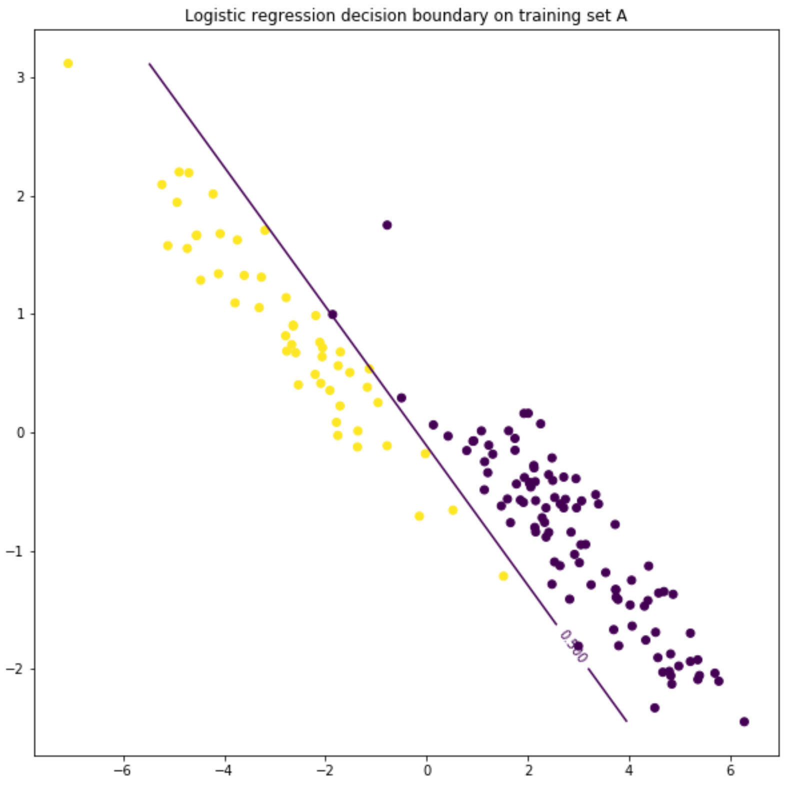
   (a) Give the numerical values of the learnt parameters for each training dataset.

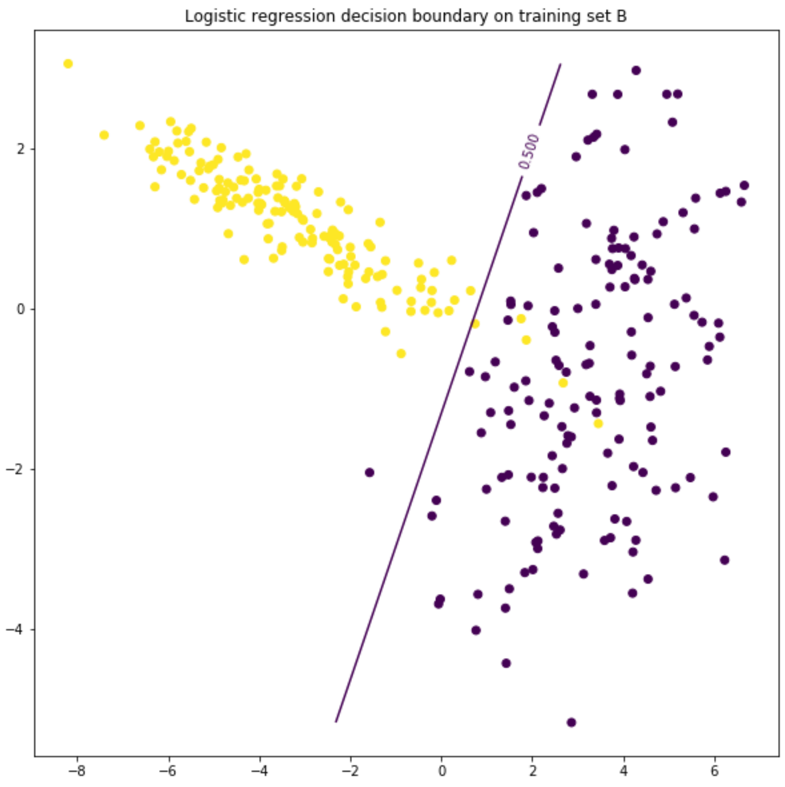
   Answer: We obtain the following parameter values for each training set, where the parameter vector corresponds to the weights and bias learned on the training set (bias is last coordinate).

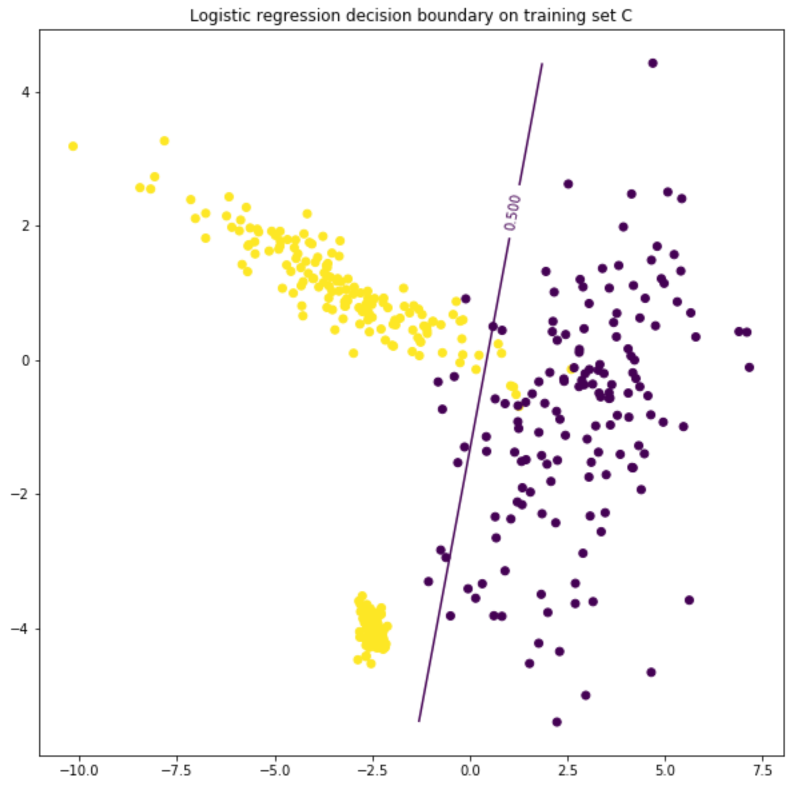
   Training set A: $w = (-25.16250945, -42.7151088, -4.81188626)$
   Training set B: $w = (-1.70518586, 1.02378538, 1.34959157)$
   Training set C: $w = (-2.2032324, 0.70926562, 0.95918885)$

   (b) Represent graphically the data as a point cloud in $\mathbb{R}^2$ and the line defined by the equation:

$$p(y = 1|x) = 0.5\,.$$


Logistic regression decision boundary on training set A
0.500
3
2
1
0
−1
−2
−6
−4
−2
0
2
4
6


Logistic regression decision boundary on training set B
0.500
2
0
−2
−4
−8
−6
−4
−2
0
2
4
6



3. **Linear regression:** as mentioned in class, we can forget that the class y can only take the two values 0 or 1 and think of it as a real-valued variable on which we can do standard linear regression (least-squares). Here, the Gaussian noise model on y does not make any sense from a generative point of view; but we can still do least-squares to estimate the parameters of a linear decision boundary (you'll be surprised by its performance despite coming from a "bad" generative model!). Implement linear regression (for an affine function $f(x) = w^\top x + b$) by solving the normal equations on each dataset (with no regularization).

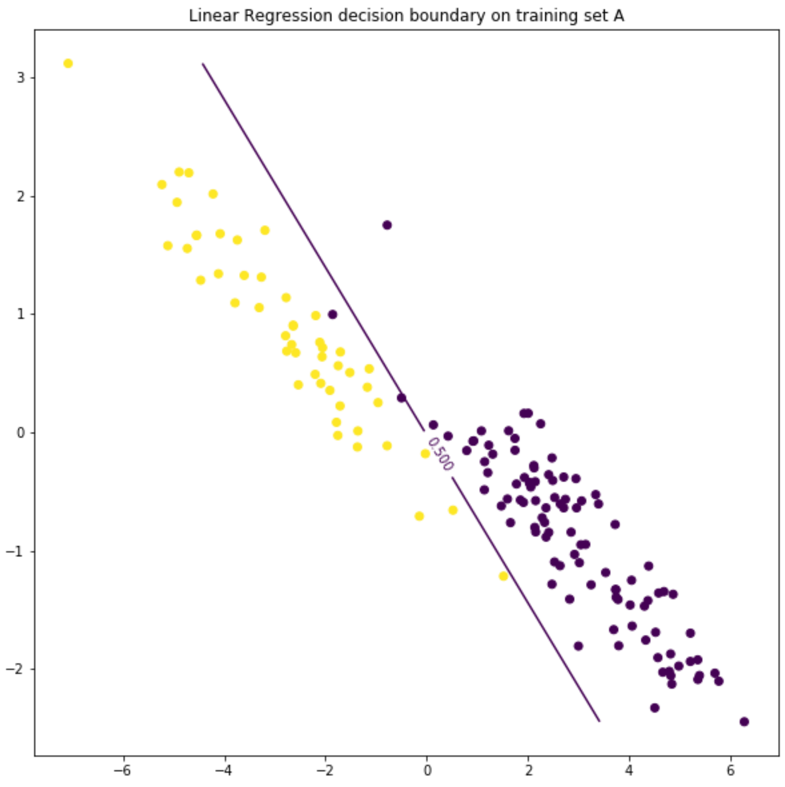
   (a) Provide the numerical values of the learnt parameters.

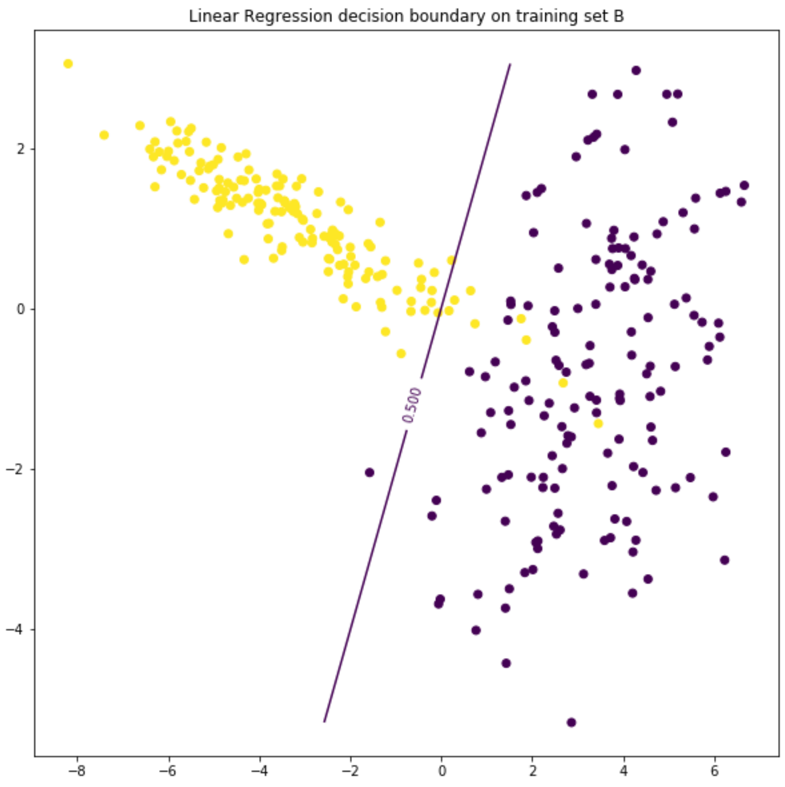
   Answer: We obtain the following parameter values for each training set, where the parameter vector corresponds to the weights and bias learned on the training set (bias is last coordinate).

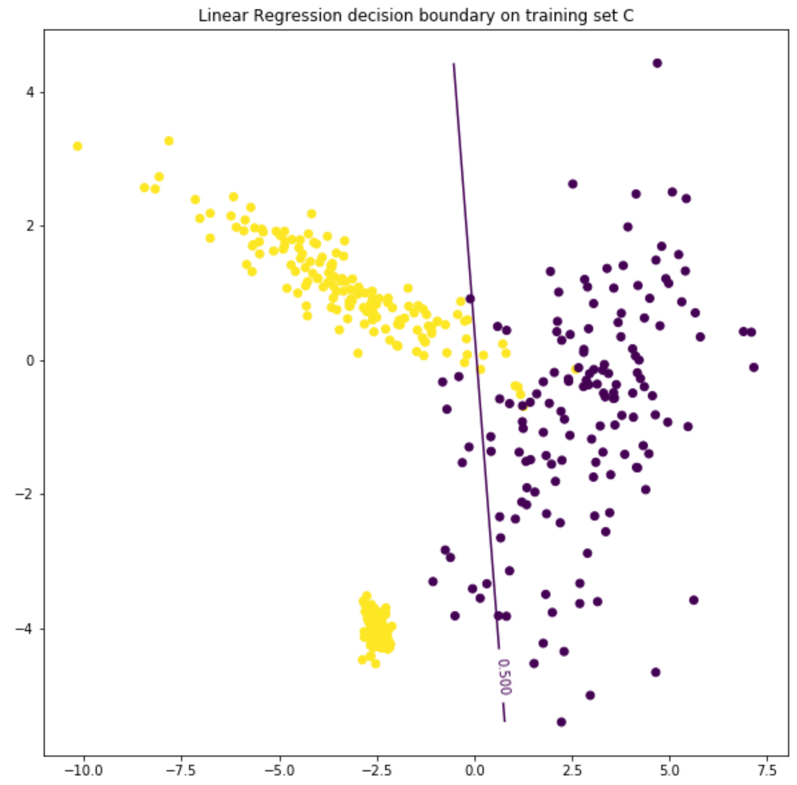
   Training set A: $w = (-0.2640075, -0.37259311, 0.49229204)$
   Training set B: $w = (-0.10424575, 0.05179118, 0.50005043)$
   Training set C: $w = (-0.12769333, -0.01700142, 0.50839982)$

   (b) Represent graphically the data as a point cloud in $\mathbb{R}^2$ and the line defined by the equation

$$f(x) = 0.5\,.$$


Linear Regression decision boundary on training set A
0.500
3
2
1
0
−1
−2
−6
−4
−2
0
2
4
6


Linear Regression decision boundary on training set B
0.500
2
0
−2
−4
−8
−6
−4
−2
0
2
4
6



4. Data in the files `classificationA.test`, `classificationB.test` and `classificationC.test` are respectively drawn from the same distribution as the data in the files `classificationA.train`, `classificationB.train` and `classificationC.train`. Test the different models learnt from the corresponding training data on these test data.

   (a) Compute for each model the misclassification error (i.e. the fraction of the data misclassified) on the training data and compute it as well on the test data.

   Answer:

	Dataset A		Dataset B		Dataset C	
	Train	Test	Train	Test	Train	Test
Fisher LDA	1.33 %	2.00 %	3.00 %	4.15 %	5.50 %	4.23 %
Logistic Regression	0.00 %	3.53 %	2.00 %	4.30 %	4.00 %	2.27 %
Linear Regression	1.33 %	2.07 %	3.00 %	4.15 %	5.50 %	4.23 %

Table 1: Error rate on each data set for each model

(b) Compare the performances of the different methods on the three datasets. Is the misclassification error larger, smaller, or similar on the training and test data? Why? Which methods yield very similar/dissimilar results ? Which methods yield the best results on the different datasets ? Provide an interpretation.

Answer:

An initially surprising result from these performances is that the test error is not always higher than the training error. Although this would generally be the expected result from real-world problems, since the training data does not fully encapsulate the true underlying distribution, here it makes sense because both sets of data are drawn from the exact same distribution. Additionally, smaller samples in the test sets mean there is a lower likelihood of outliers within the gaussian samples, thus explaining how a lower test error is possible. Beyond that, since the sampling process is stochastic, a higher test error is also possible. This is observed for all models on dataset B for example.

LDA and Linear Regression have essentially identical results. After reading upon this topic, it would indeed seem that for binary classification, LDA and Linear Regression are equivalent in that they produce proportional parameter values with respect to each other. Logistic Regression clearly differs in its results, as we would expect considering it does not make the same distributional assumptions as LDA.

In that vein of thought, Logistic Regression is the best overall model on datasets B and C, which makes sense as it is more robust to varying assumptions about the distribution of the data than LDA. However, LDA outperforms Logistic Regression on the test error for dataset A, which is logical as this dataset is built according to the assumptions of the LDA model. The lower training error of Logistic Regression may be due to overfitting.

5. **QDA model**. We finally relax the assumption that the covariance matrices for the two classes are the same. So, given the class label, the data are now assumed to be Gaussian with means and covariance matrices which are a priori different:

$$Y \sim \text{Bernoulli}(\pi), \quad X \,|\, Y = j \sim \mathcal{N}(\mu_j, \Sigma_j).$$

Implement the maximum likelihood estimator and apply it to the data.

Answer:

We can inspire ourselves from the results for LDA to get the MLE in the case of QDA. Indeed, in order to get $p(y = 1|x)$ here, all we need to do is plug in the different covariance matrices Σ_1, Σ_0 into equation (8) in the derivation of question 1.b). Thus, :

$$p(y=1|x) = \frac{1}{1+\exp\{-\log(\frac{\pi}{1-\pi}) + \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\}}$$

Since the quadratic forms no longer cancel out as they did for LDA, we will get a non-linear decision boundary. This also means we need an MLE estimator for both Σ_1, Σ_0. The estimators for μ_1, μ_0 and π, however, clearly do not change. That said, we can use the form of equation (3) in question 1.a) to obtain the gradients for Σ_1, Σ_0 by treating the irrelevant covariance matrix as a constant in each case. In addition, the log determinant term will be split between the two covariance matrices for this log likelihood.Thus the log-likelihood here is:

$$\begin{aligned}
l(\theta, D) &= \sum_{i=1}^{n} \log(p(y_i|\pi)p(x_i|y_i,\theta)) \\
&= \sum_{i=1}^{n} y_i \log(\pi) + \sum_{i=1}^{n}(1-y_i)\log(1-\pi) + \sum_{i=1}^{n} \frac{-yi}{2}(x_i-\mu_1)^T\Sigma_1^{-1}(x_i-\mu_1) \\
&+ \sum_{i=1}^{n}(\frac{-(1-yi)}{2}(x_i-\mu_0)^T\Sigma_0^{-1}(x_i-\mu_0)) \cdot -\frac{n}{2}\log(2\pi) \cdot -\frac{1}{2}\sum_{i=1}^{n} y_i \log|\Sigma_1| \\
&- \frac{1}{2}\sum_{i=1}^{n}(1-y_i)\log|\Sigma_0|
\end{aligned}$$

From this we can compute the gradient w.r.t each matrix as:

$$\nabla_{\Sigma_1^{-1}} l = \nabla_{\Sigma_1^{-1}}\{-\frac{1}{2}\sum_{i=1}^{n} y_i \log|\Sigma_1^{-1}| - \frac{n_1}{2}\langle\Sigma_1^{-1}, \tilde{\Sigma_1}\rangle\} \tag{10}$$

$$= \frac{1}{2}(\Sigma_1 \sum_{i=1}^{n} y_i - n_1\tilde{\Sigma_1}) \tag{11}$$

$$\nabla_{\Sigma_0^{-1}} l = \nabla_{\Sigma_0^{-1}}\{-\frac{1}{2}\sum_{i=1}^{n}(1-y_i)\log|\Sigma_0^{-1}| - \frac{n_1}{2}\langle\Sigma_0^{-1}, \tilde{\Sigma_0}\rangle\} \tag{12}$$

$$= \frac{1}{2}(\Sigma_0 \sum_{i=1}^{n}(1-y_i) - n_1\tilde{\Sigma_0}) \tag{13}$$

Where the notation is the same as for LDA. Thus setting to 0, we get :

$$\begin{aligned}
\hat{\Sigma_1}_{MLE} &= \frac{n_1 \cdot \tilde{\Sigma_1}}{\sum_{i=1}^{n} y_i} \\
&= \frac{\sum_{i=1}^{n} y_i(x_i-u_1)(x_i-u_1)^T}{\sum_{i=1}^{n} y_i}
\end{aligned}$$

$$\begin{aligned}\hat{\Sigma}_{0MLE} &= \frac{n_0 \cdot \tilde{\Sigma}_0}{\sum_{i=1}^{n}(1-y_i)} \\ &= \frac{\sum_{i=1}^{n}(1-y_i)(x_i-u_0)(x_i-u_0)^T}{\sum_{i=1}^{n}(1-y_i)}\end{aligned}$$

We now implement this model in the Jupyter notebook in order to provide the answers to the questions below.

(a) Provide the numerical values of the learnt parameters.
Answer:

Training set A ——————————
Sigma 1: [[2.70442172 -1.3008515] [-1.3008515 0.68969588]]
Sigma 0: [[2.31065259 -1.04748461] [-1.04748461 0.57578403]]
mu 1: [[-2.69232004] [0.866042]]
mu 0: [[2.89970947] [-0.893874]]
pi: 0.3333333333333333

Training set B ——————————
Sigma 1: [[4.15361075 -1.33454097] [-1.33454097 0.51607059]]
Sigma 0: [[2.53885859 1.0642112] [1.0642112 2.96007891]]
mu 1: [[-3.21670734] [1.08306733]]
mu 0: [[3.34068896] [-0.83546333]]
pi: 0.5

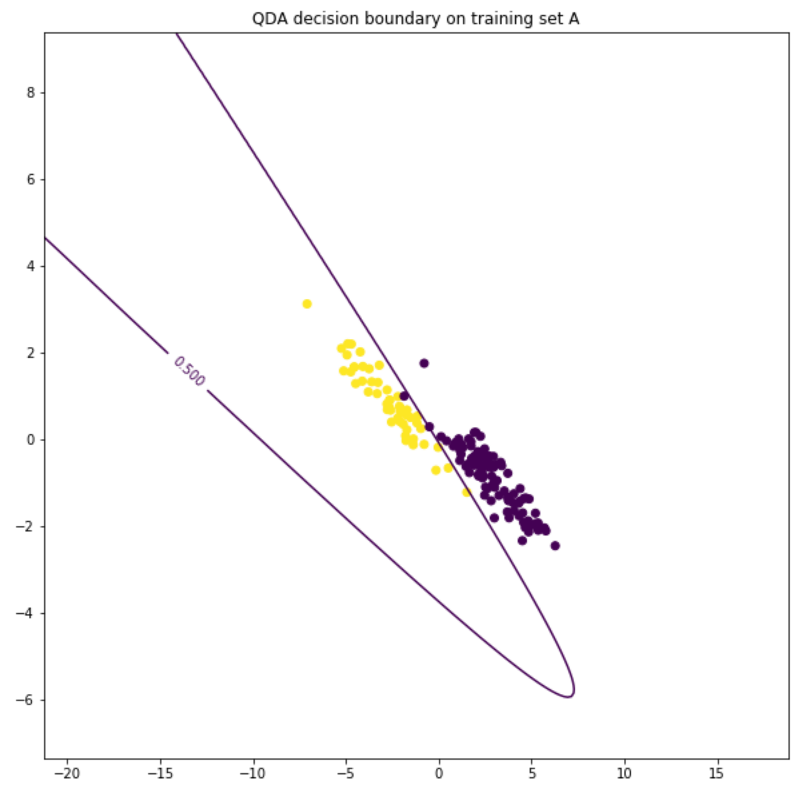
Training set C ——————————
Sigma 1: [[2.86914403 -1.76197061] [-1.76197061 6.56438626]]
Sigma 0: [[2.89913927 1.24581553] [1.24581553 2.92475448]]
mu 1: [[-2.94232885] [-0.9578284]]
mu 0: [[2.79304824] [-0.83838667]]
pi: 0.625

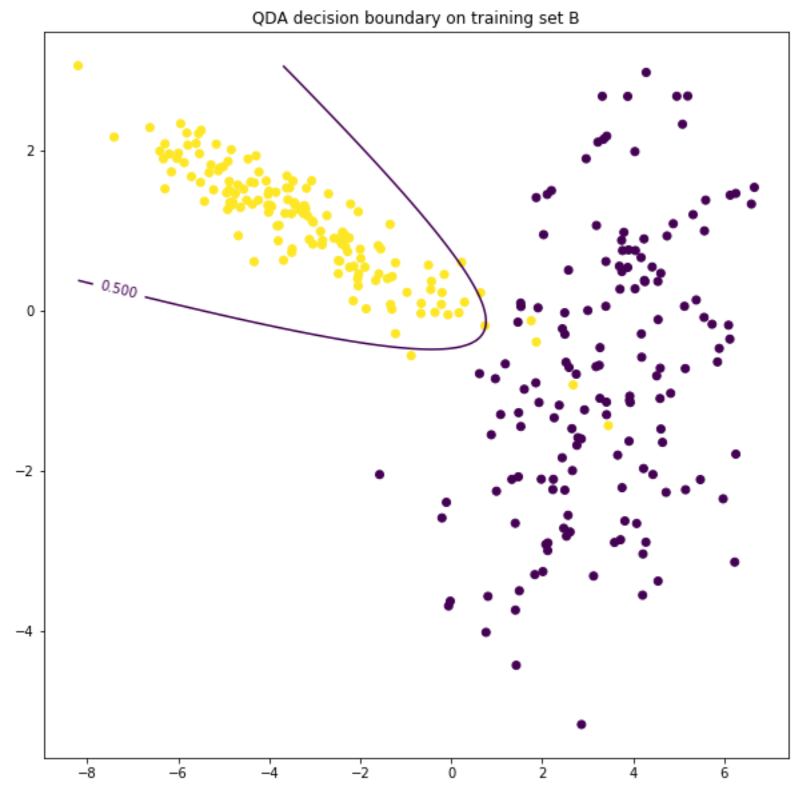
(b) Represent graphically the data as well as the conic defined by

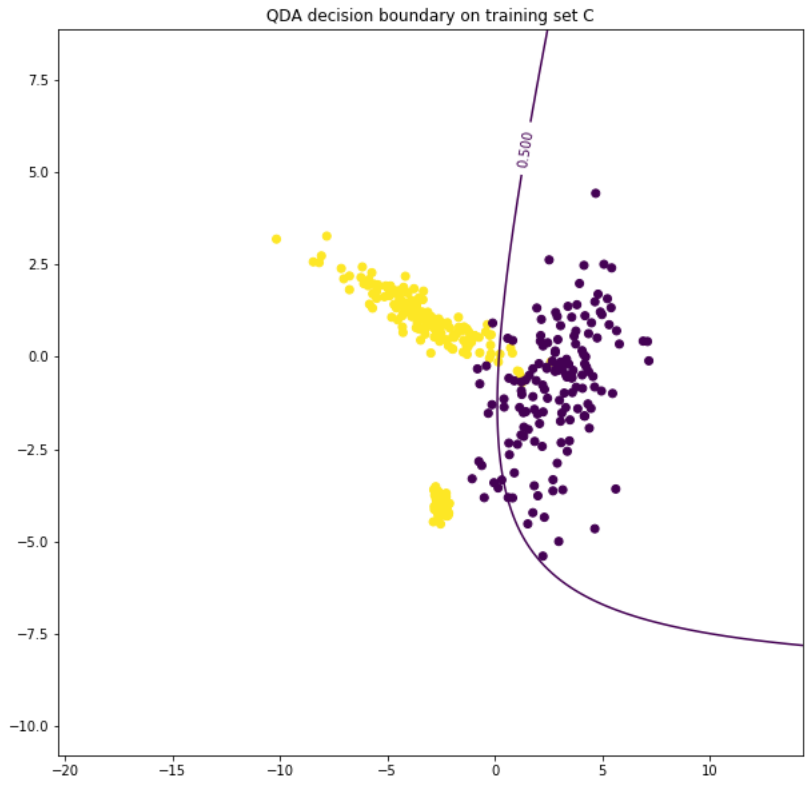
$$p(y=1|x) = 0.5\,.$$

Answer:

See below the plotted figures.


QDA decision boundary on training set A
0.500


QDA decision boundary on training set B
0.500



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

Answer:

	Dataset A		Dataset B		Dataset C	
	Train	Test	Train	Test	Train	Test
Fisher LDA	1.33 %	2.00 %	3.00 %	4.15 %	5.50 %	4.23 %
Logistic Regression	0.00 %	3.53 %	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 %	1.87 %	2.33 %	2.35 %	5.25 %	4.03 %

Table 2: Error rate on each data set for each model, with QDA

(d) Comment the results as previously.

Answer:

Looking at the new results, we can see that QDA mostly outperforms the other models. Indeed, given a quadratic decision boundary, it is natural that it has the capacity to better fit the data, especially if this data is non-linearly separable. That said, we wouldn't expect QDA to do much better than LDA on Dataset A, and although it slightly outperforms the difference is marginal. Otherwise, the only place QDA is outperformed is on dataset C by Logistic Regression. In this case it would seem the linear boundary obtained by Logistic Regression generalizes better to the test set.