Simulation Problem

Mohamad Lakkis

March, 2024

This section is to explain some of the coding part in the *Simulation Problem*

Question 1

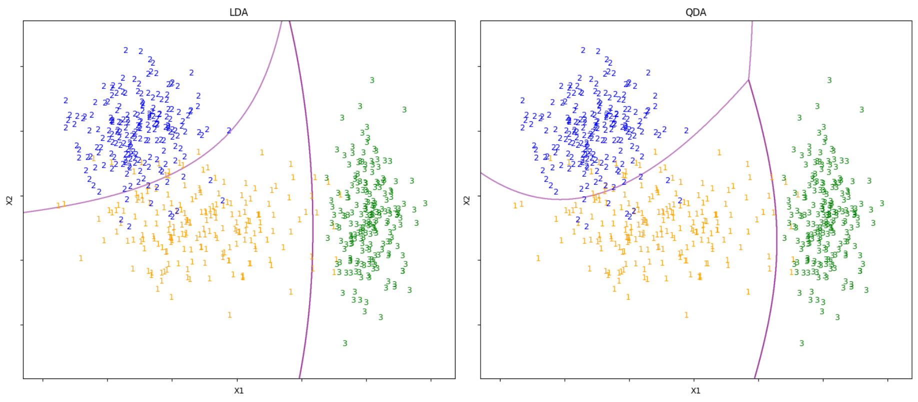

Figure 1: The two methods for fitting a quadratic boundaries(Figure 4.6)

So we can clearly see that both pictures are very similar in their boundaries, which gives us an alternative to the QDA, which is LDA but by adding $\frac{p(p+1)}{2}$ variables consisting of X_1^2, X_1X_2... So notice that this boundary would still be linear in the augmented space $\mathbb{R}^q$ ($q = p+ \frac{p(p+1)}{2}$), but if we map down to $\mathbb{R}^p$ we will get quadratic boundaries since we will be expressing them in terms of our initial (X_1, ..., X_p), and simply said we would have terms like X_1^2 which is quadratic, but in the augmented space $\mathbb{R}^q$ we could refer to them as (X_1,...,X_q) which would be linear since we are using LDA !

Question 2

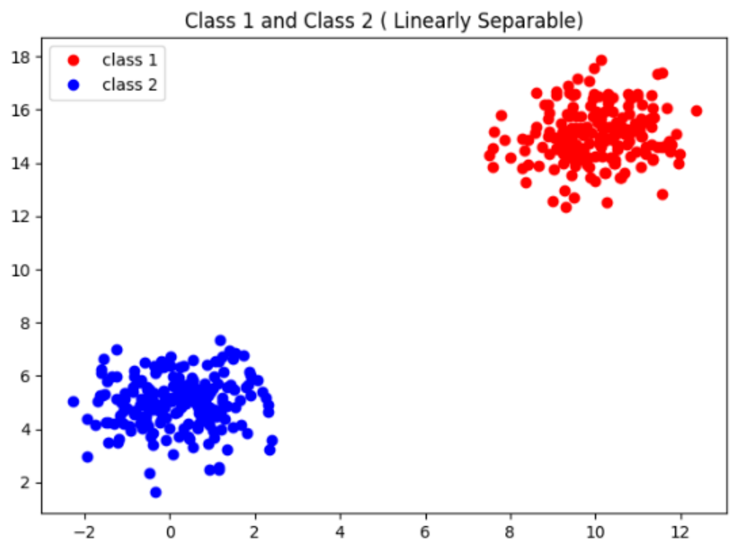

Figure 2: The two classes are linearly separable with the same covariance matrix

Comments(related to Problem 6 in theoretical part)

- We can see that for large C the boundary lines becomes very close to the boundry line of Fig 3, where we didn't use a penalty term, which is logical. Note that as we decrease C, the stronger the penalty becomes which might lead to some coefficients being exactly 0.

- We used the saga solver which has its own algorithm, which even for separable data it might converge (since they add a penalty term of their own but even then this doesn't guarantee that the algorithm should converge), if the number of max iteration is large enough. In our implementation we use $max_{iter} = 10^7$, if we lower it, the algorithm might not converge. Note that $\underline{\text{theoretically}}$ the algorithm shouldn't converge since we can keep increasing the β towards ∞. And so adding a penalty term can prevent that from happening, as we have seen in figure 4.

- If you look at Figure 5 you can clearly see that as we increase C (or similarly removing the penalty term), the sigmoid function becomes very close to a step-function, because the class are separable in this case. So it is consistent with what we have discussed in problem 6 !

- Note that if we change the initial point in the saga solver we might get different estimations and the algorithm might not converge then.

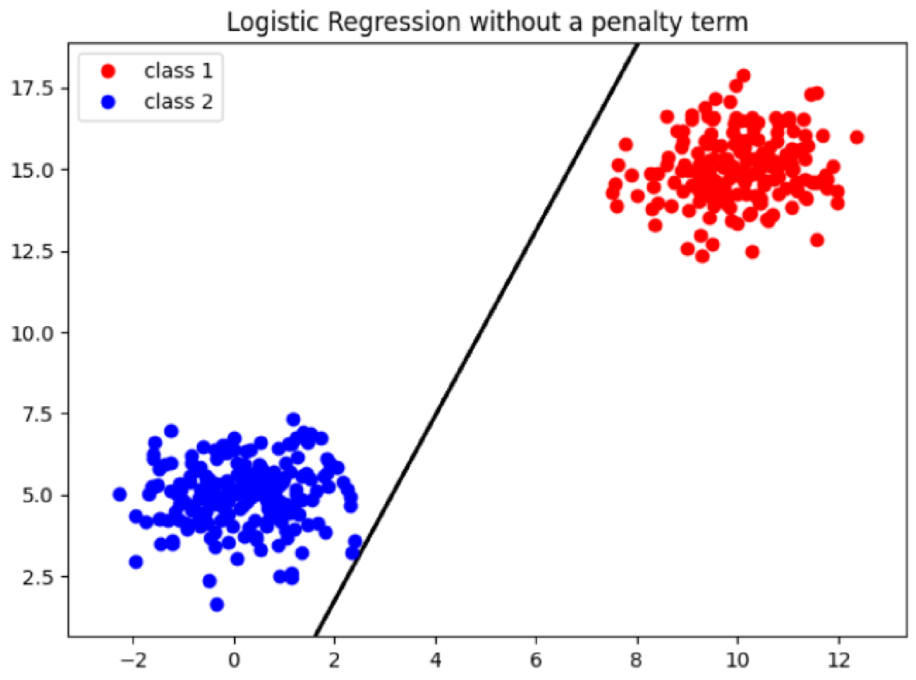

Figure 3: Logistic Regression boundary without a penalty term

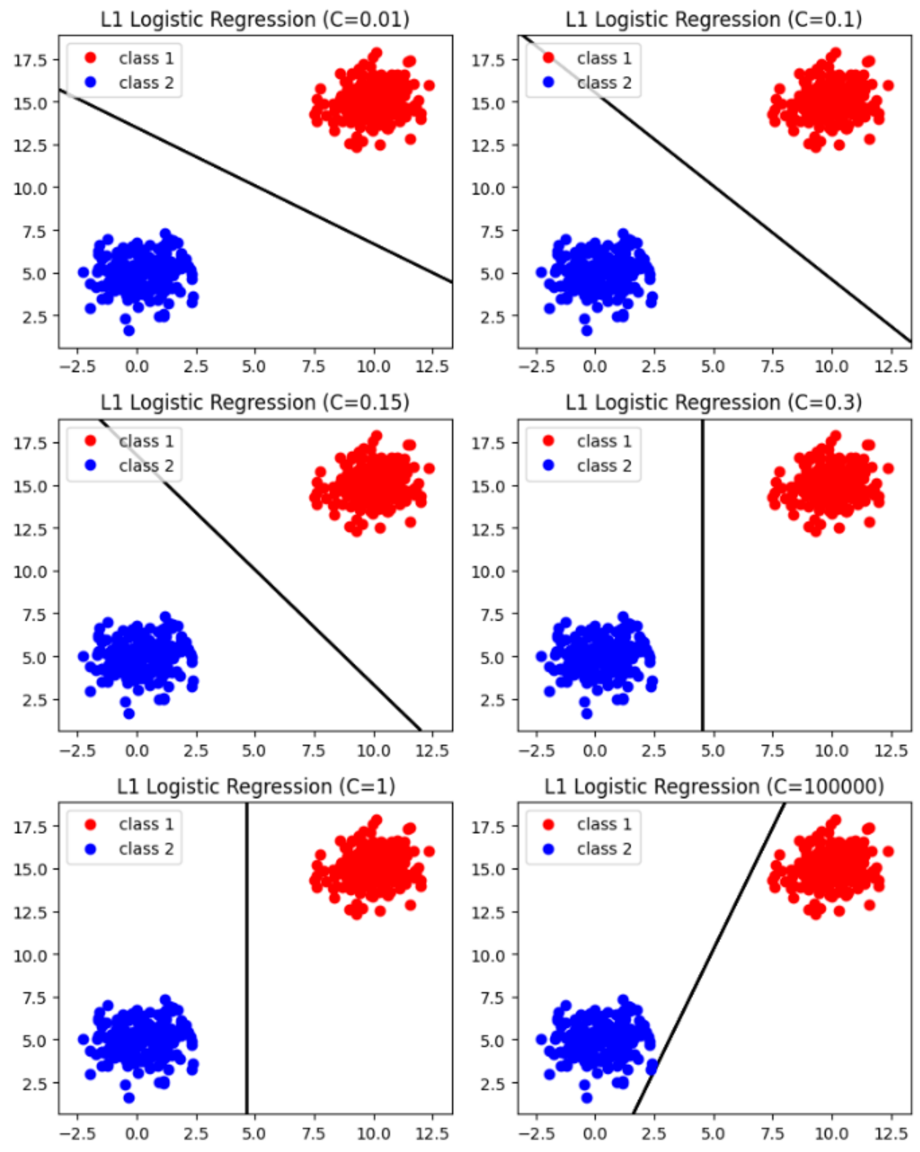

Figure 4: Logistic Regression boundary with an L1-penalty term for different C values

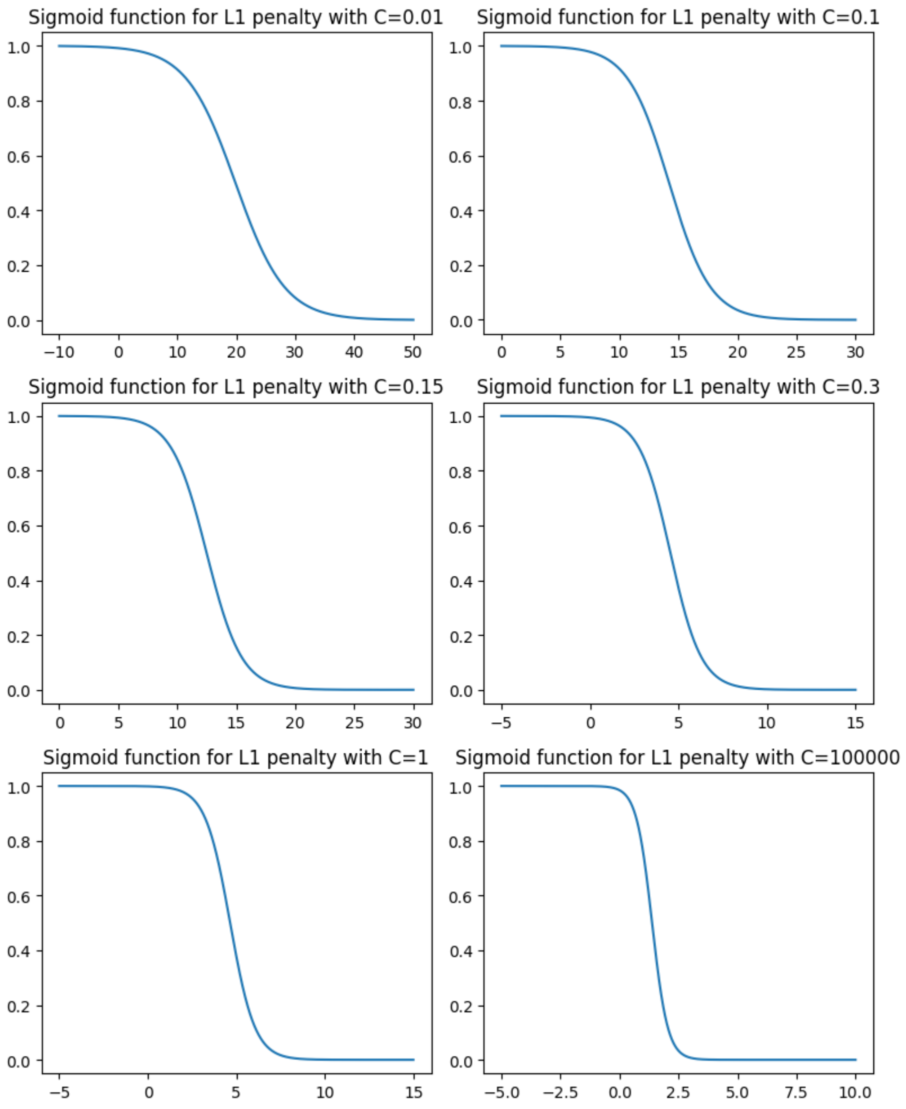

Figure 5: The Sigmoid Function for each C

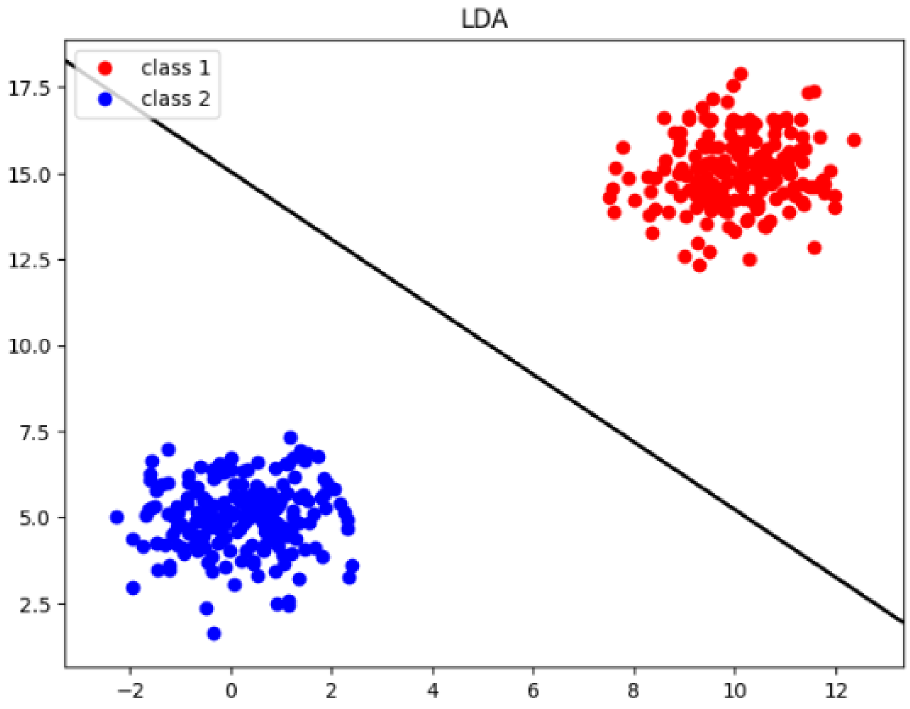

Figure 6: LDA boundary

Additional notes

- For Figure 3, this is how we are modeling the objective function which we need to minimize
$$\text{Loss} = -\text{log likelihood}$$
- For Figure 4, this is how we are modeling the objective function which we need to minimize(note that C is like λ in the lasso)
$$\text{Loss} = -\text{log likelihood} + C\sum_{j=1}^{n} |\beta_j|$$
- The way we got the sigmoid functions (figure 5) is by first fitting the Logistic Regression model with an L1-penalty with a particular C. Then getting the coefficients and the intercept of this particular model. Then we fixed X2 to a constant = 0 for simplicity(we could have taken the linear combination $\beta_0 + \beta_1 X_1 + \beta_2 X_2$ and in that case on the x-axis we would have this value, but we care about the shape of this curve which would be the same regardless). Then we applied our best fitted line to a range of values by multiplying each value by the first coefficient and adding to it the intercept. SO now we have an array of these values, which we then applied the sigmoid function on each one of them to get the shape that we have. On the x-axis, it represents the X_1 values !