Linear support vector machines

Introduction to Machine Learning

University of Barcelona

December 13, 2015

Authors:
Kazancli, Erol
Leyva, María

Contents

1 Understanding the primal

1.1 Hard margin

Using the dataset "example_dataset_1.mat" and training a hard margin linear support vector machine, we get the following hyperplane.

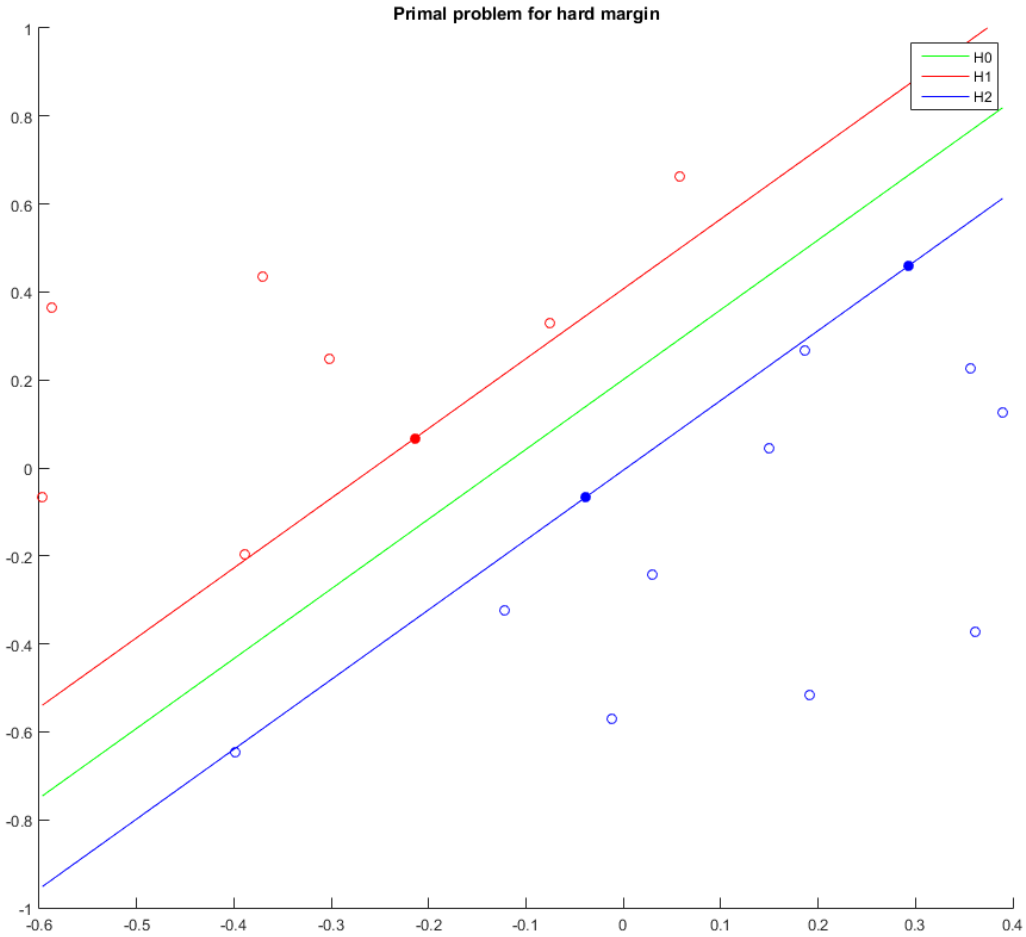

Figure 1: Hyperplane obtained by a svm with hard margin

Model obtained:

```
w=
     7.6950
    -4.8511

b=
     0.9752
```

$$H_0 = \frac{-b - w_1 * x}{w_2}$$

$$H_1 = \frac{-b + 1 - w_1 * x}{w_2}$$

$$H_2 = \frac{-b - 1 - w_1 * x}{w_2}$$

1.1.1 Obtaning the support vectors

The support vectors are the closest datapoints to H_0. They're the decisive datapoints in order to find the hyperplane. So the support vectors would be the points of each class that don't satisfy the constraint of being greater than 1 or smaller than -1 (for each class). Or, which is the same, the datapoints located inside the space defined between H_1 and H_2. They can be seen in the previous figure as filled circles.

The support vectors have to fulfil:

$$w * x_{+1} + b \leq 1$$

$$w * x_{\text{-}1} + b \geq -1$$

$$H_2 = \frac{-0.9752 - 1 - 7.6950 * x}{-4.8511}$$

1.2 Soft margin

1.2.1 Example dataset 1

With $\lambda = 0$ the minimized function looks exactly the same with the one in hard-margin problem, however the constraints do not, because the slack variables in the constraints do not disappear and even though the problem is a linearly separable one, they are not forced to be zeros, they have some positive values. As a result the weights are too small in absolute value, which causes the margin lines to be far apart(because the line functions have huge bias terms - (-b $\pm$ 1)/w(2)). Therefore it is not reasonable to think that the hyperplane will be similar obtained in the hard-margin case. The separation obtained can be seen in the figure 2. The model with $\lambda = 0$ does not make much sense anyway, as long as the slacks are in the constraints. If we want to get a model similar to the hard-margin we need to increase λ which will force the slack variables to zero.

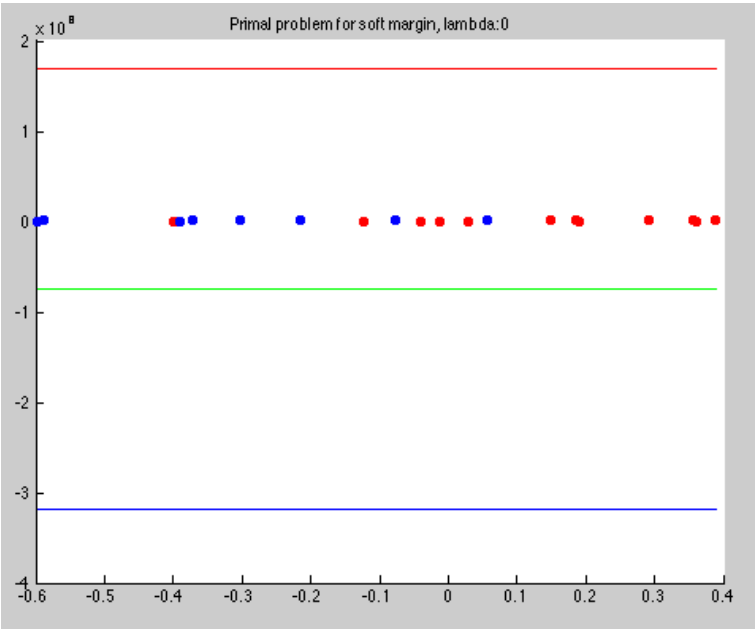

Figure 2: Hyperplane obtained by a svm with soft margin, λ= 0

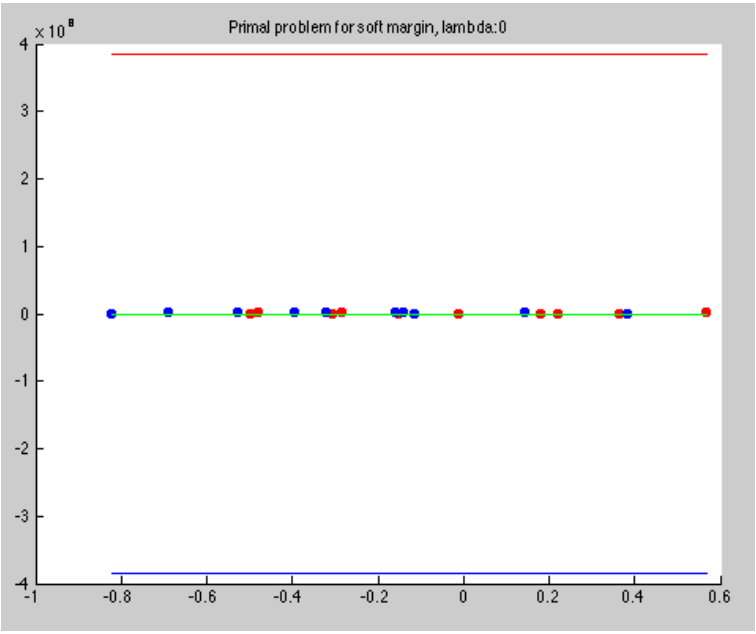

Figure 3: Soft margin with non separable data, $\lambda = 0$

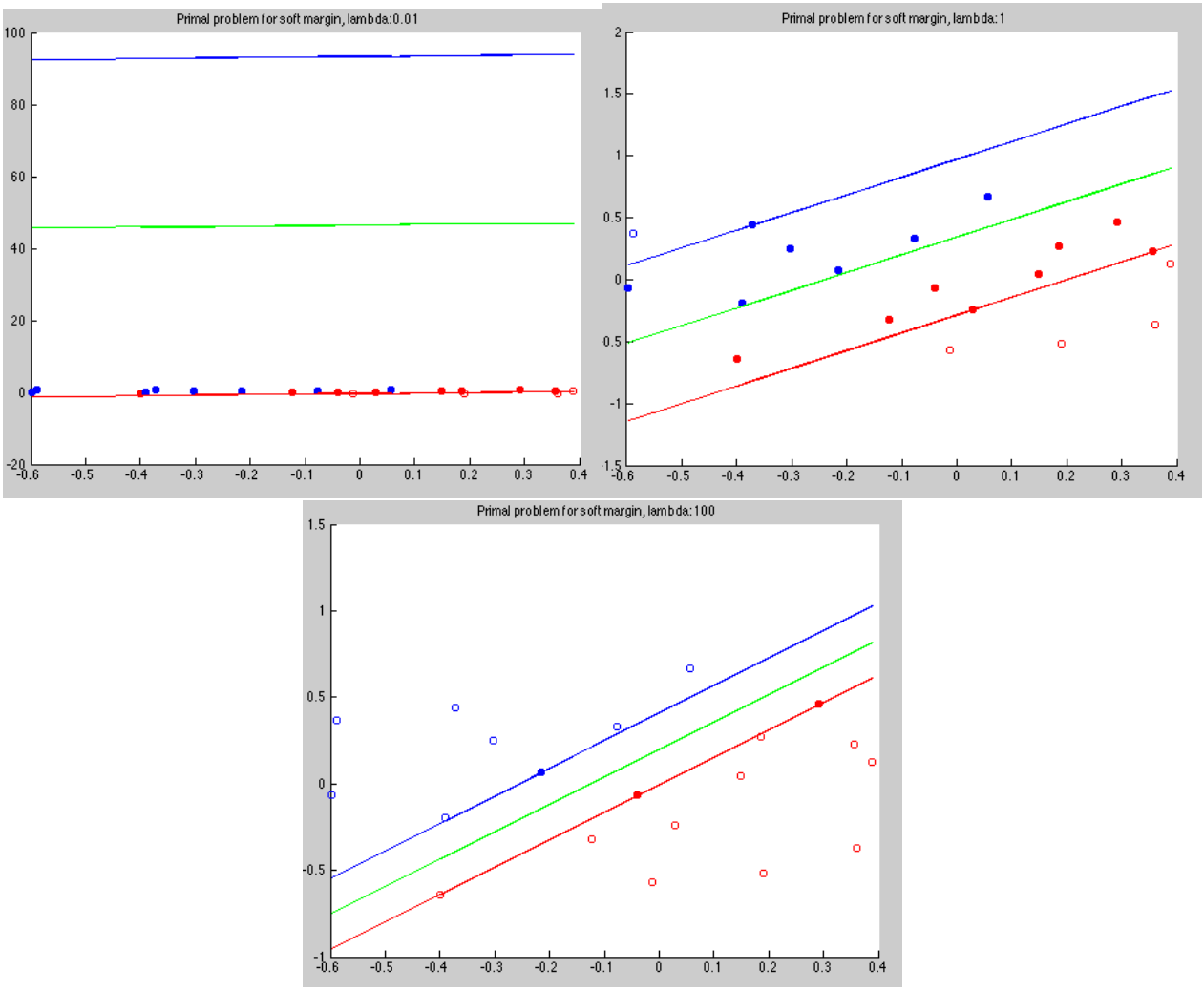

Figure 4: Soft margin with separable data, $\lambda = 0.01$, 1, 100

The model obtained using the non-separable data("toy_dataset.mat") with $\lambda = 0$ can be seen in figure 3. As can be seen the margins are too far apart. This is caused by the fact that the weights obtained are extremely small in absolute value (because of the 0-valued λ and the slack terms in conditions, which have no pressure to be low), which leads the bias term((-b $\pm$ 1)/w(2))) in the separating line functions of the margins to be too high in absolute value pushing the lines to extremes.

The models obtained using the "example_dataset_1.mat" data with $\lambda = 0.01$, 1, 100 can be seen in figure 4.

The expected values of slack variables on the margins are 0 or very close to zero, because they are correctly classified. When we observe the slack variables for the support vectors for λ= 100 we see that they are close to zero, however since all the slack variables in this case are close to zero it is difficult to distinguish the support vectors by only looking at them. We can only say they are the ones with the biggest values in this case. This could be due to the fact that the data set is linearly separable and the the lambda value is quite high. Therefore the support vectors are only on the margins, which leave them with the highest slack variables because the rest of the points are all correctly specified.

Slack variables for λ= 100:

```
Values of u
   1.0e-10 *
    0.0810
    0.0810
    0.0810
    0.1382 - margin, SPV
    0.0810
    0.0810
    0.0810
    0.0810
    0.0810
    0.0810
    0.0810
    0.1351 - margin, SPV
    0.0810
    0.0810
    0.0810
    0.0821 - margin, SPV
    0.0810
    0.0810
    0.0810
    0.0810
```

1.2.2 Example dataset 2

The models obtained using the "example_dataset_2.mat" data with primal SVM and λ= 0.01, 1, 100 can be seen in figure5.

When we observe the values of slack variables for λ= 10 we see that there are many zeroes (which correspond to the correctly classified ones) and positive values(which correspond to the ones in the margin or the misclassified ones). We can say that the points with the positive slack variables are the support vectors, which is quite logical since we expect the points outside the margin to be zero(if not misclassified). However they should not exceed the value 2 because in that case they are the misclassified ones. Points with u's greater than 0 and smaller than 2 are the support vectors.

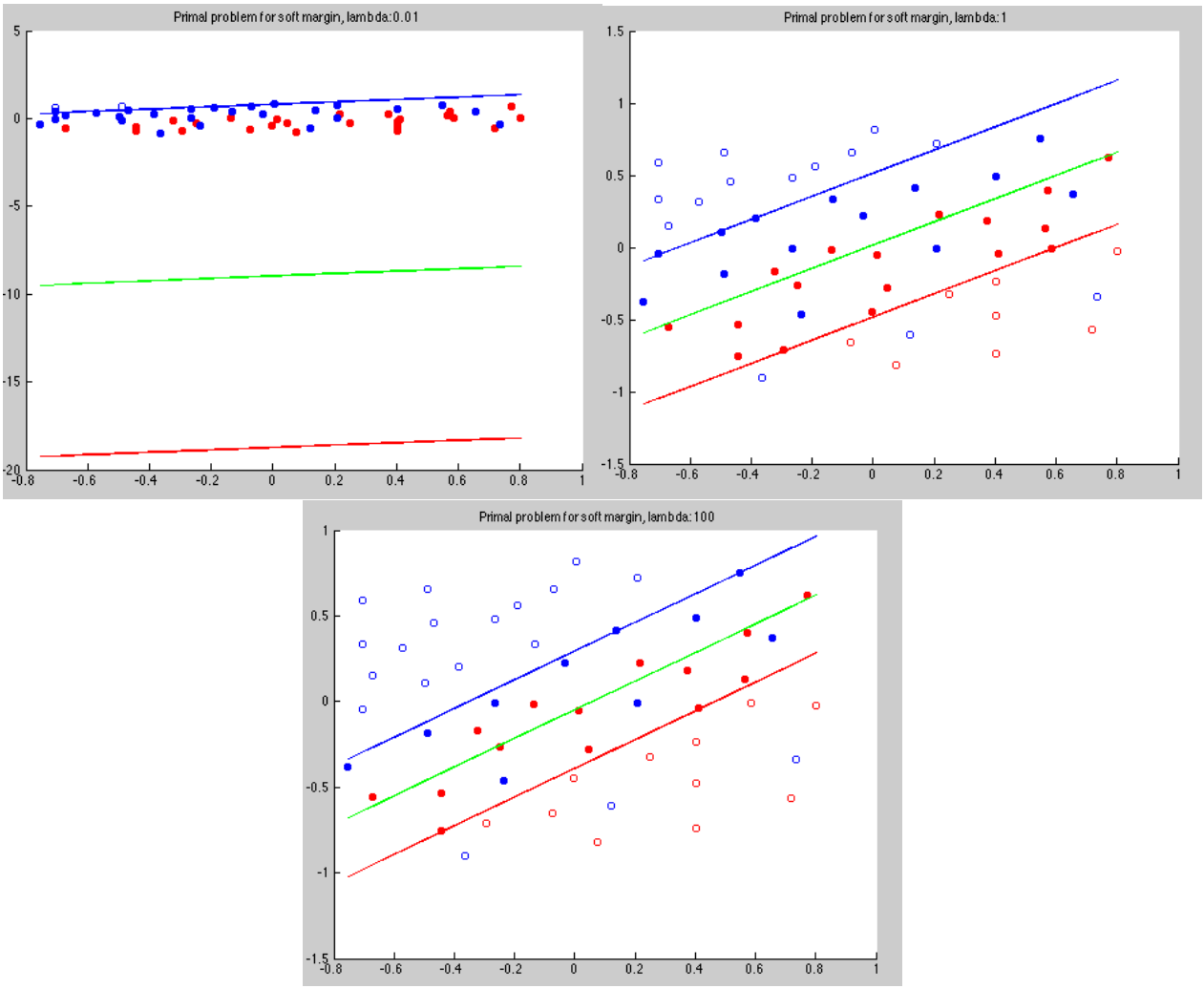

Figure 5: Soft margin with non-separable data, λ= 0.01, 1, 100

2 Understanding the dual

The models obtained using the "example_dataset_2.mat" data with dual SVM and λ= 0.01, 1, 100 can be seen below.

When we observe the values of slack variables for λ= 10 we see that there are many zeroes (which correspond to the correctly classified ones) and positive values(which correspond to the ones in the margin or the misclassified ones). This time it is more difficult to detect the support vectors. Even though they have slack values which are bigger than zero and smaller than or equal to λ, the misclassified points also have slack variables very close to λ value. Therefore it is difficult to distinguish between a support vector and a misclassified point.

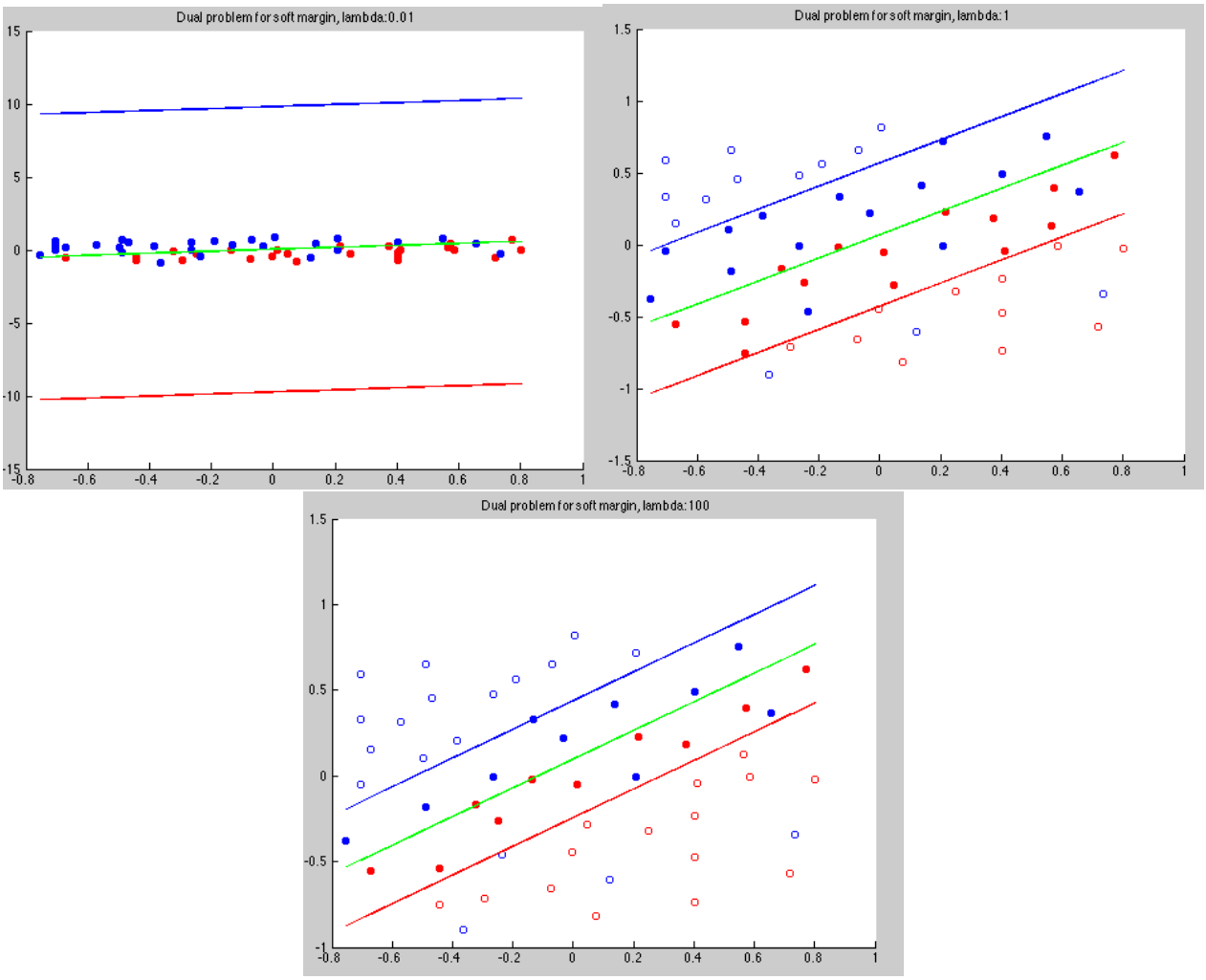

Figure 6: Dual soft margin with non-separable data, λ= 0.01, 1, 100

3 Unbalanced data

We are working with a dataset which have unbalanced data. The -1 class is determinant, so we will discuss different methods to get the minimun error rate in that.

Class 1	Class -1	Total
110	13	123

Table 1: Unbalanced dataset

3.1 Unweighted classes

3.1.1 Choosing the best λ

Lambda	0	1e-2	1	10	1e2	1e3
Error	0.1057	0.1057	0.0894	0.0650	0.0488	0.0569

Table 2: λ values vs training error rates

We get the lambda with lowest error rate. We get $\lambda = 1e2$, and plot the hyperplane.

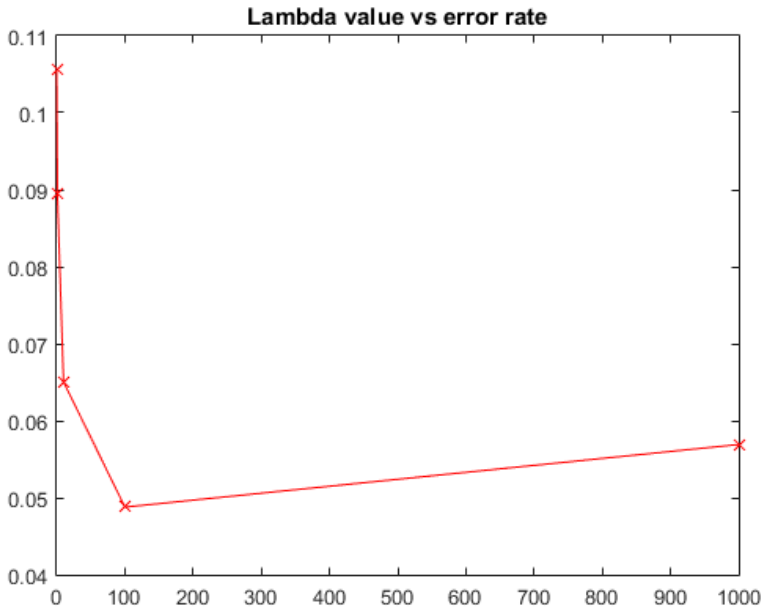

Figure 7: λ values vs error rates

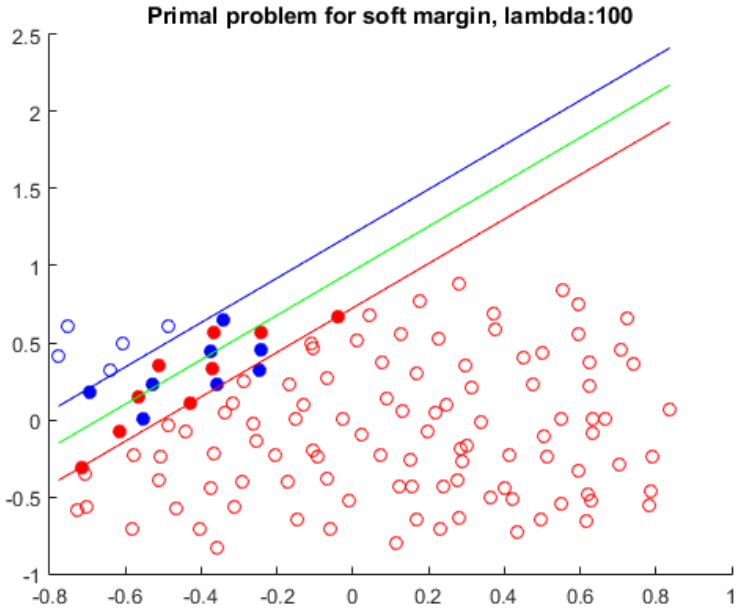

Figure 8: Hyperplane for λ= 1e2

This result is not satisfying: the critical class still gets examples misclassified, and this is not desirable at all.

3.2 Weighted classes

We are balancing the classes applying weights to the SVM algorithm, in order to consider more important to classify better the members of the -1 class, or, in this case, in order to consider less important to misclassify the members of the +1 class.

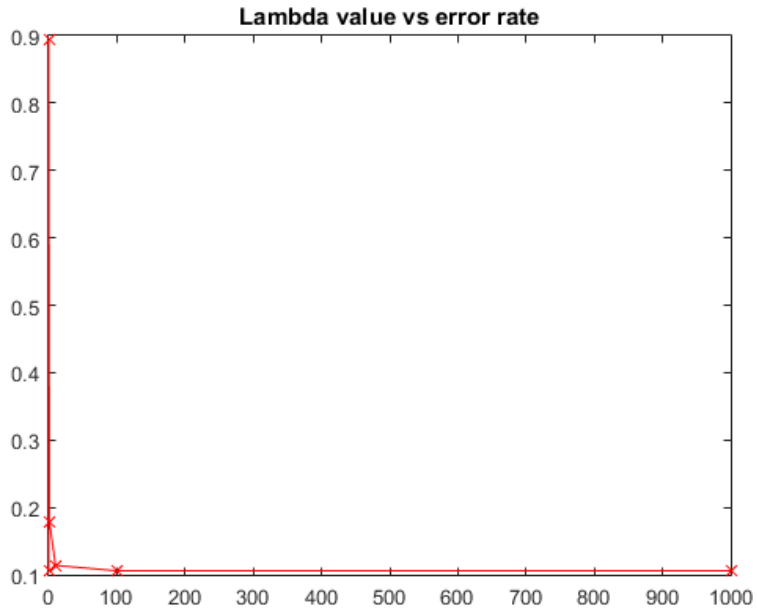

Figure 9: λ values vs error rates

Lambda	0	1e-2	1	10	1e2	1e3
Error	0.1057	0.8943	0.1789	0.1138	0.1057	0.1057

Table 3: λ values vs training error rates

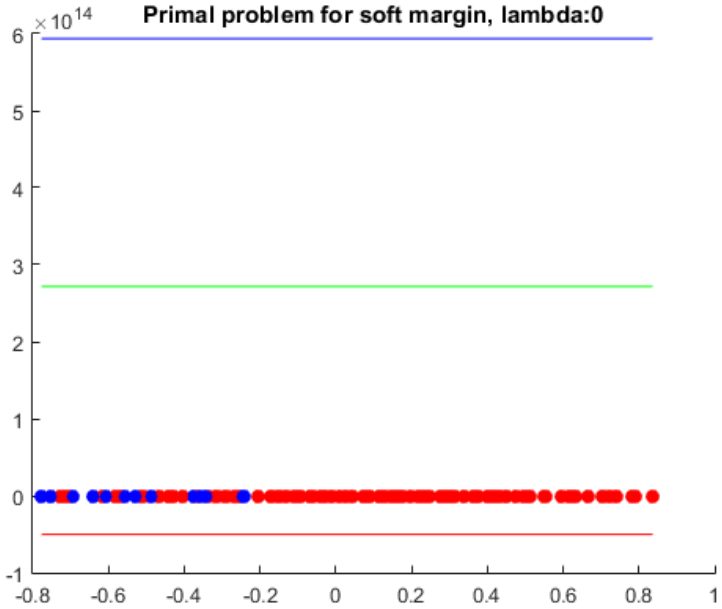

Figure 10: Hyperplane for λ= 0

3.2.1 Choosing the best λ

We get the lambda with lowest error rate. This time, we get $\lambda = 0$. We know this is a horrible classifier, and can be seen in the figure, but actually, it has less misclassified data than any of the others. The solution would be to use weighted error rates.

This result is not satisfying: we don't have any member of the critical class misclassified, but in

exchange, we have all members of the positive class misclassified.

3.2.2 Choosing the best λ with weighted error rates

Lambda	0	1e-2	1	10	1e2	1e3
Error	0.1057	0.1057	0.0211	0.0135	0.0125	0.0125

Table 4: λ values vs weighted training error rates

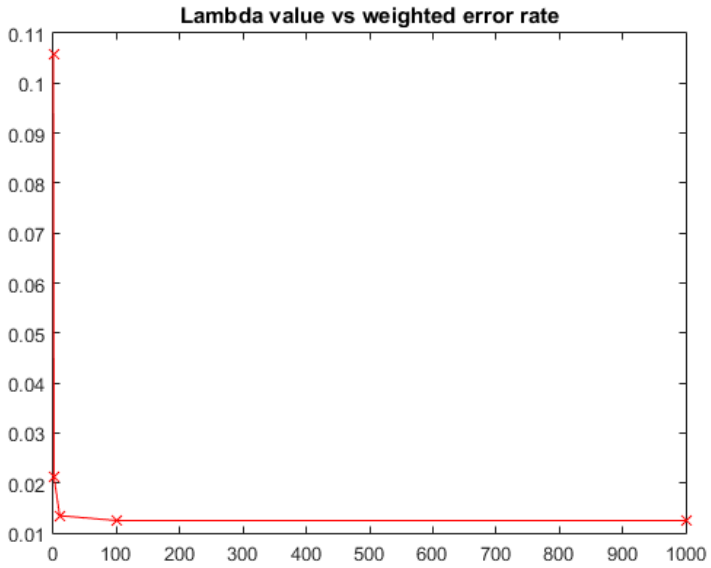

Figure 11: λ values vs weighted error rates

We get the lambda with lowest error rate. We have $\lambda = 100$. Now we can see that this classifier is way better.

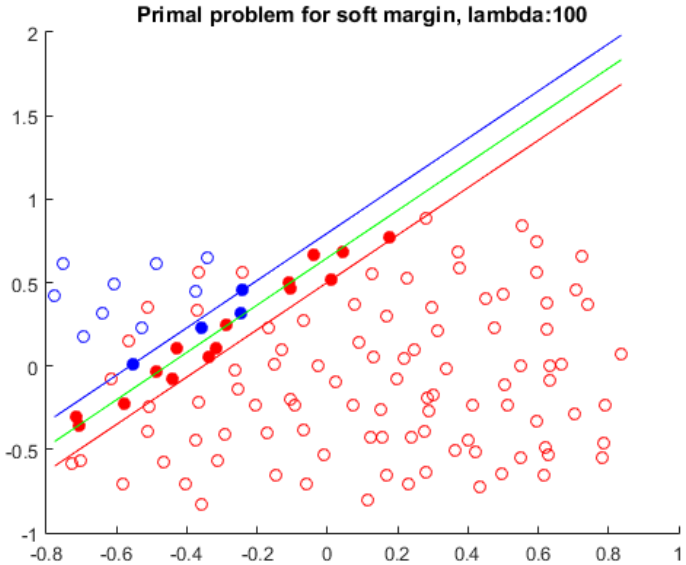

Figure 12: Hyperplane for λ= 100

This result is quite satisfying: the critical class does not get members misclassified, but also the

non critical class gets the minimum error possible.

4 Running the code

In order to run all our code, you can execute the W5 script included. This will interactively run all the scripts corresponding to all the exercises, plotting the figures and showing in console the data this assignment asked for.

Please note that after each exercise block, the program will close all the windows and clear the variables and the command line.