Exercice 3: LINEAR SUPPORT VECTOR MACHINES

```
A workaround on SVM
```

Contents

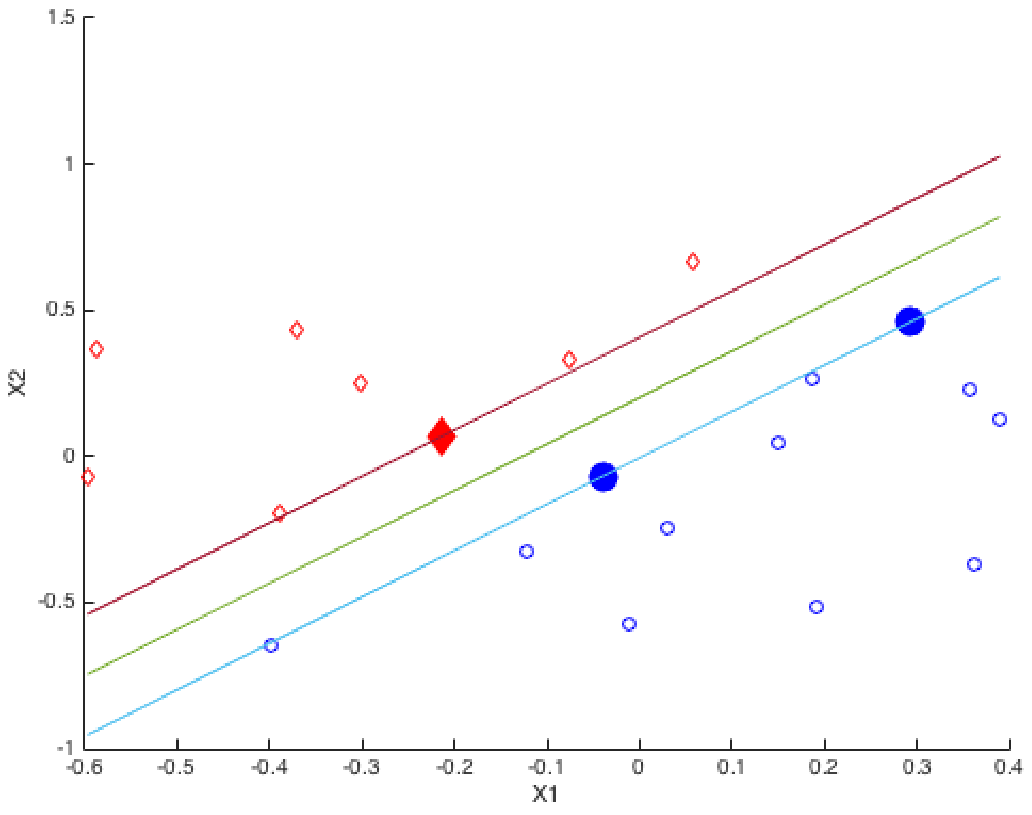
- Block 1
- Block 2
- Block 3
- Block 4
- Block 5
- Block 6

Block 1

The following code shows the Support Vector machine hyperplane, the lower and upper boundary, and the support vectors remarked over the others.

```
clear
load('data/example_dataset_1.mat');

[w, b, upp, low] = svm_linear_primal(data, labels);
svm_draw(data,labels,w,b, upp, low)
```

```
upp =

    12    16


low =

     4
```



In order to identify the different SVs, we considered that if an instance was at a projection distance of 1 from the hyperplane it was a Support Vector.

We've considered this approach, because if we consider the ?dual solution? approach, only the instances with weigthed values would be considered as Support Vetors.

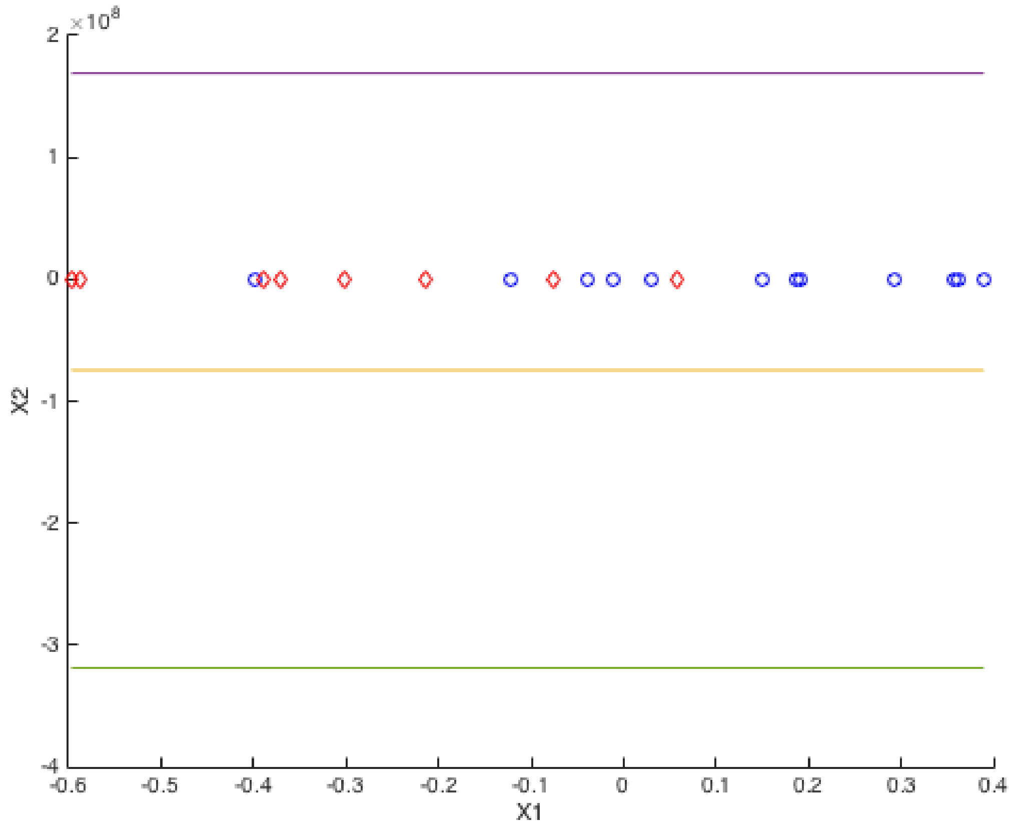
Block 2

In the soft margin approach, when we use a small λ, the margin size increase, with a lambda equals to 0, the boundary size increase. That's caused because the following function:

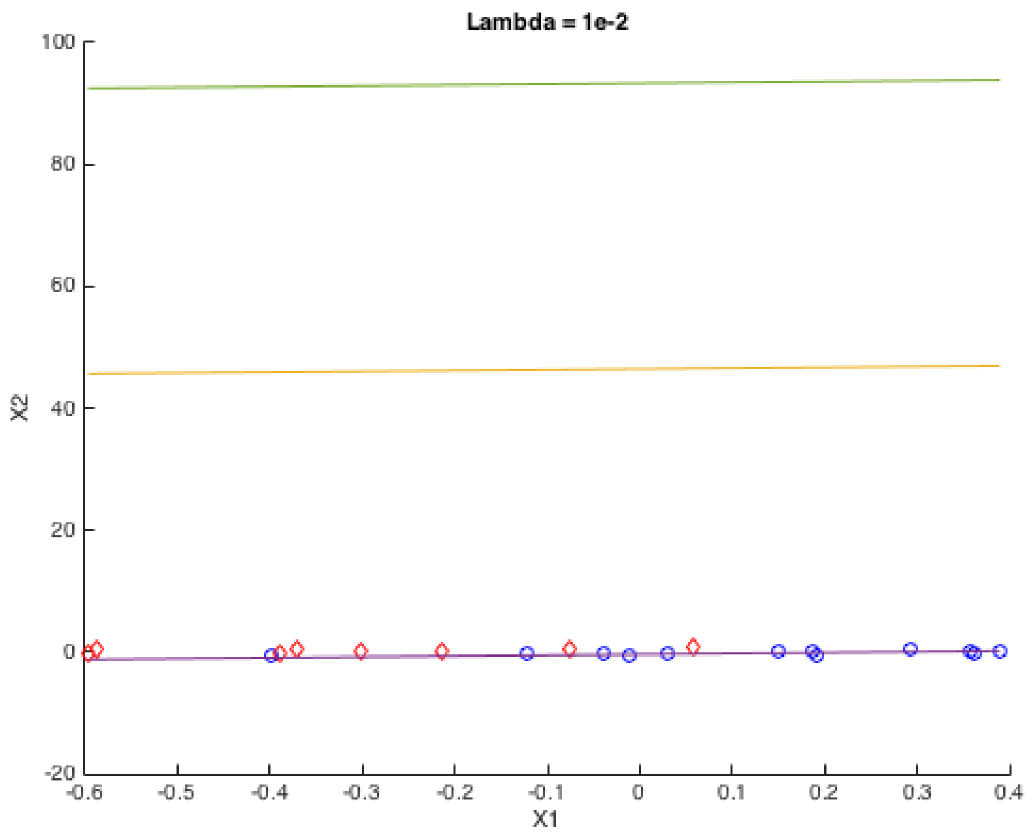
$$argmin\left\{||w|| + \lambda * \sum \xi\right\}$$

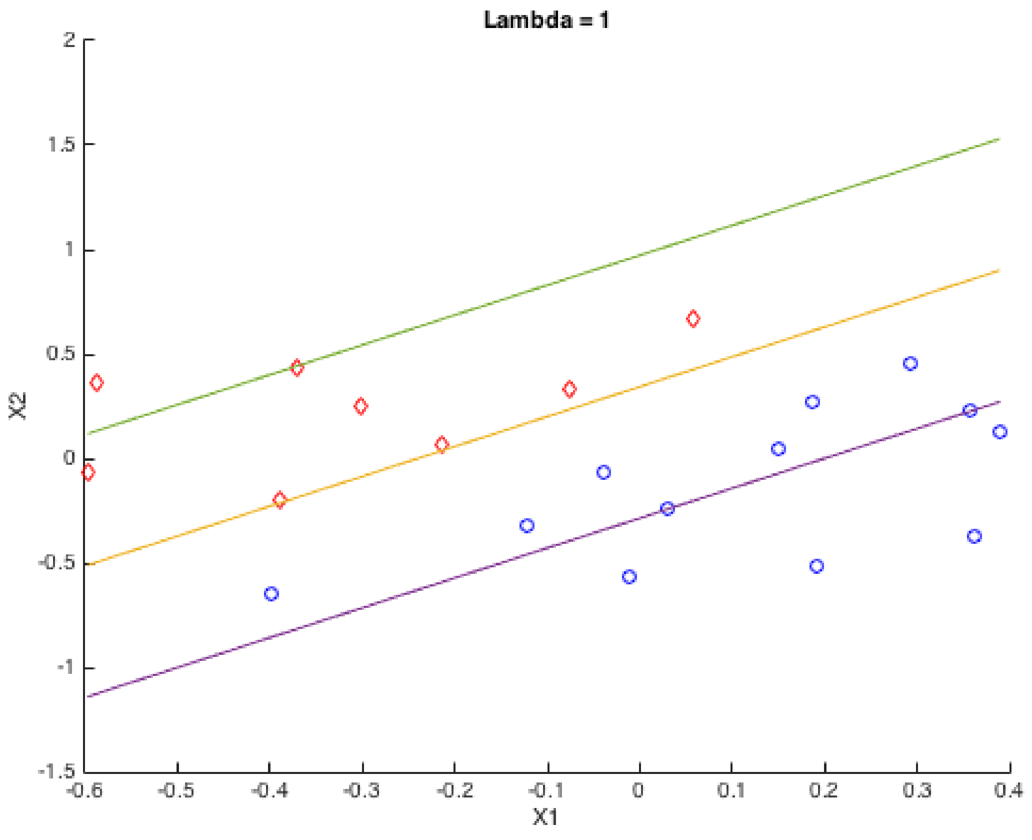
So if λ coefficient is zero, ξ is not taken into account in the minimization function, so the missclassificated samples aren't taken into account.

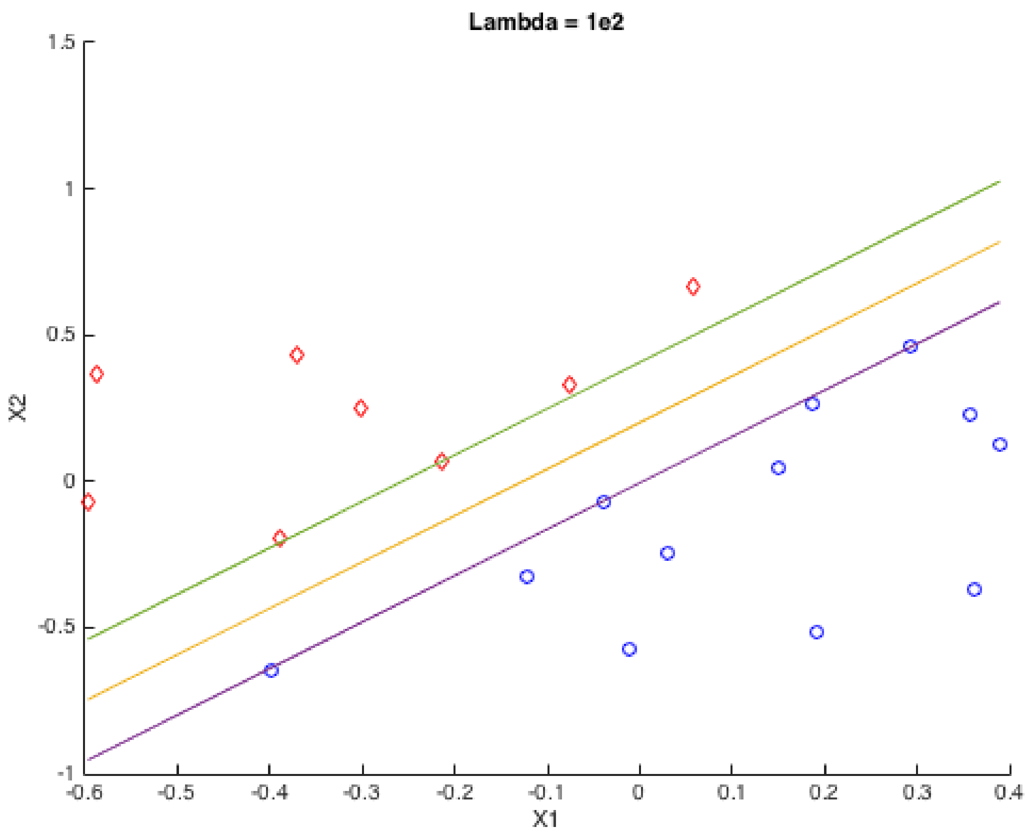
```
clear
load('data/example_dataset_1.mat');
[w, b, upp, low] = svm_soft(data, labels, 0); % We use 1e-8 instead of 0
svm_draw(data,labels,w,b);
```



```
[w, b, upp, low] = svm_soft(data, labels, 1e-2);
svm_draw(data,labels,w,b);
title('Lambda = 1e-2');
[w, b, upp, low] = svm_soft(data, labels, 1);
svm_draw(data,labels,w,b);
title('Lambda = 1');
[w, b, upp, low] = svm_soft(data, labels, 1e2);
svm_draw(data,labels,w,b);
title('Lambda = 1e2');
```


Lambda = 1e-2
X2
X1
100
80
60
40
20
0
-20
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4


Lambda = 1
X2
X1
2
1.5
1
0.5
0
-0.5
-1
-1.5
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4

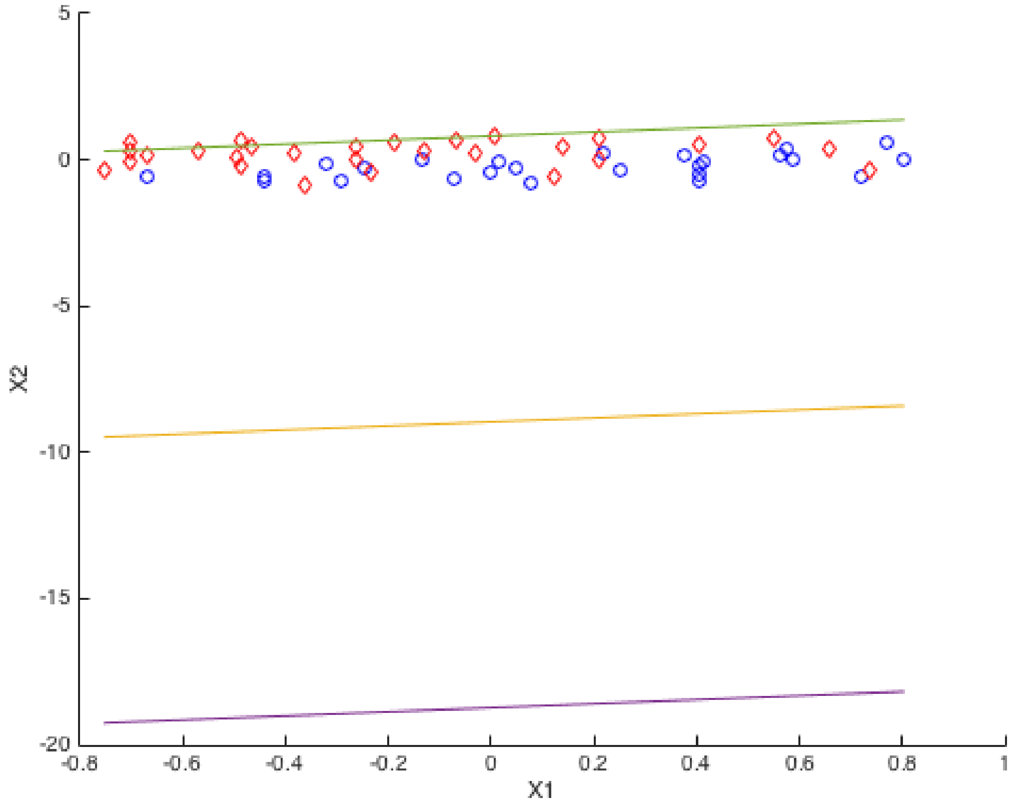


The expected value for the support vectors with margin equals 1 is 0.

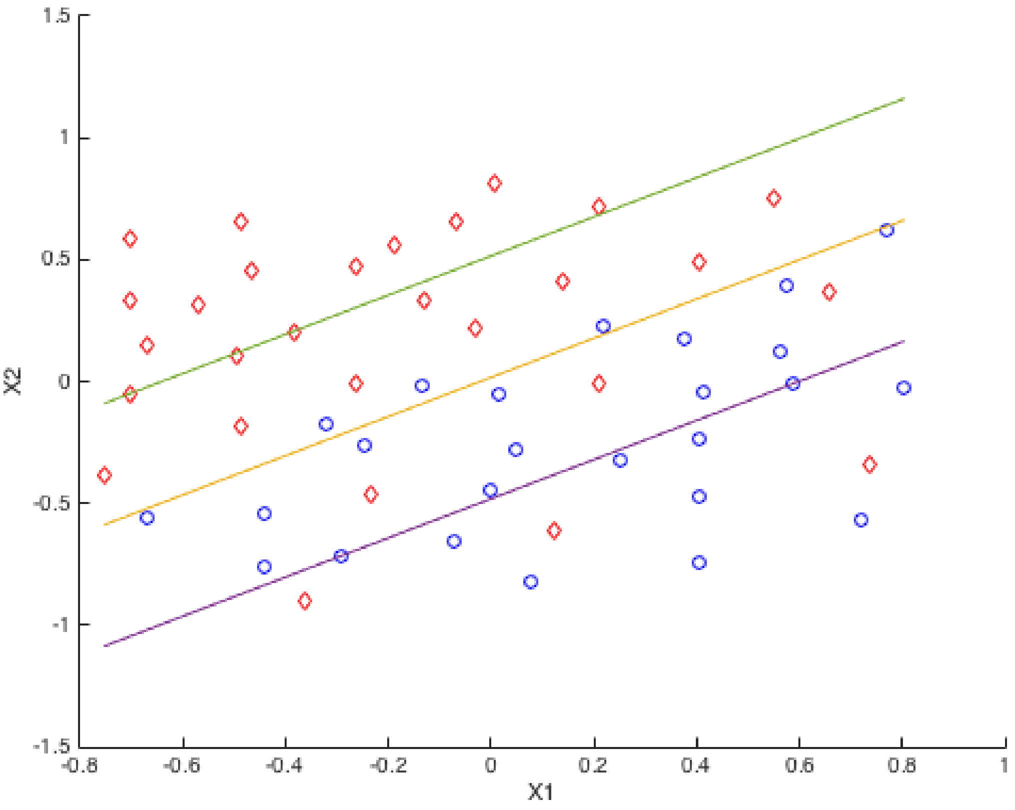
In order to identify the support vectors we have to find those samples that are into the margin function. In this case we have two samples in the upper bound data(49) and data(51) and one smaple in the lowe bound, data(9)

Block 3

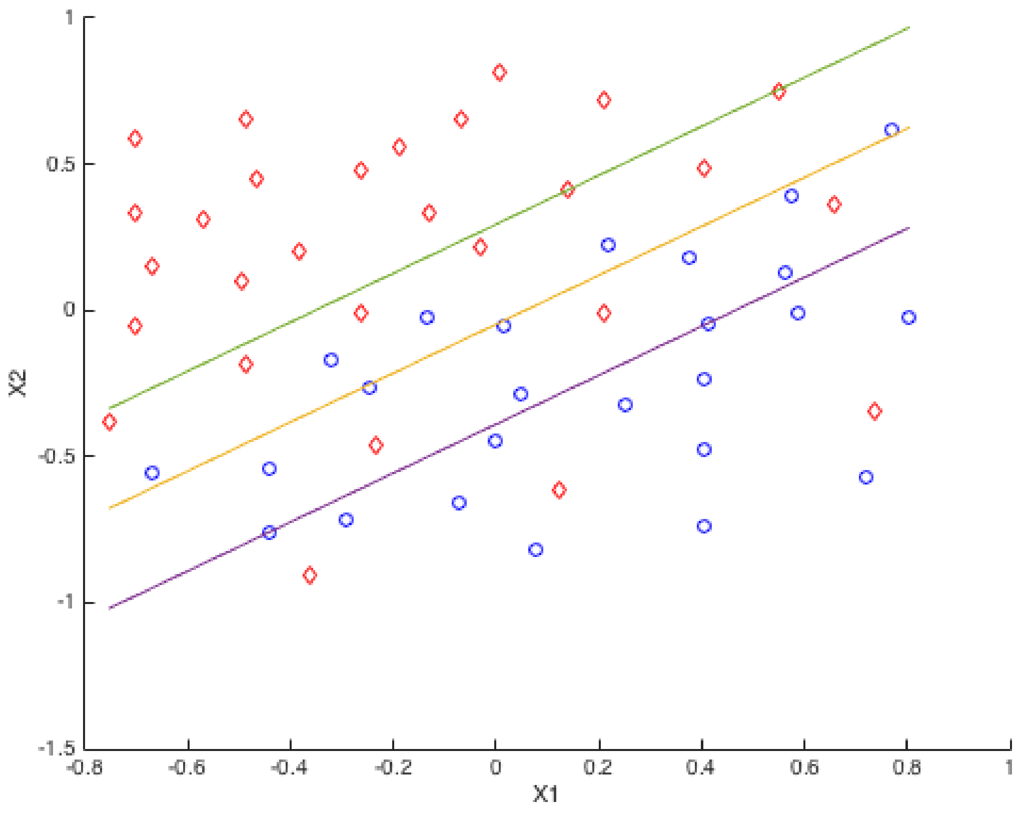
```
clear
load('data/example_dataset_2.mat');
[w, b, upp, low] = svm_soft(data, labels, 1e-2);
svm_draw(data,labels,w,b);
[w, b, upp, low] = svm_soft(data, labels, 1);
svm_draw(data,labels,w,b);
[w, b, upp, low] = svm_soft(data, labels, 1e2);
svm_draw(data,labels,w,b);


[w, b, upp, low] = svm_soft(data, labels, 10);
svm_draw(data,labels,w,b);
```

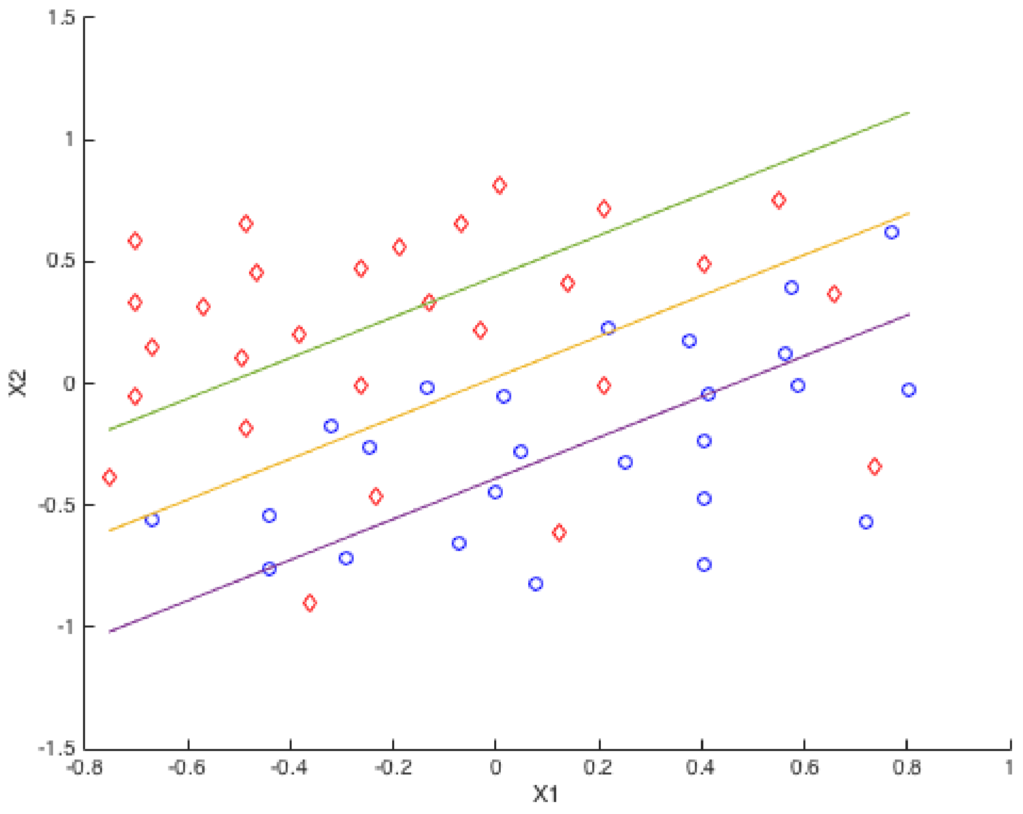


5
0
-5
-10
-15
-20
X2
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
X1



1.5
1
0.5
0
-0.5
-1
-1.5
X2
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
X1



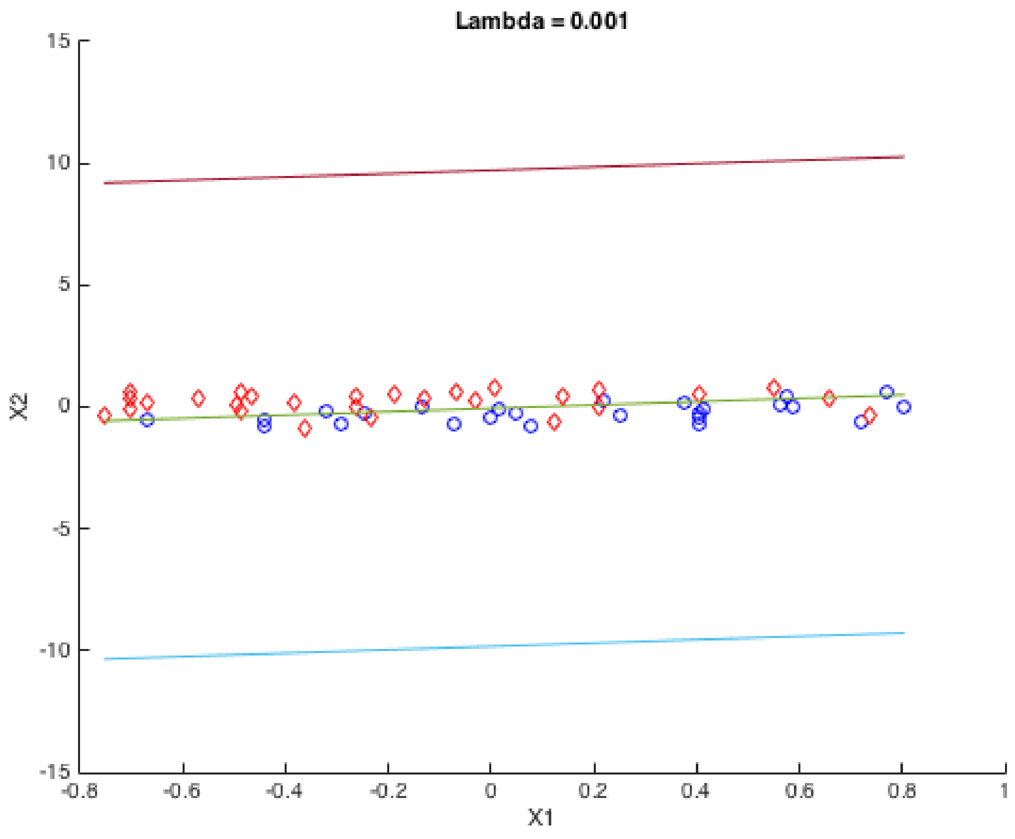


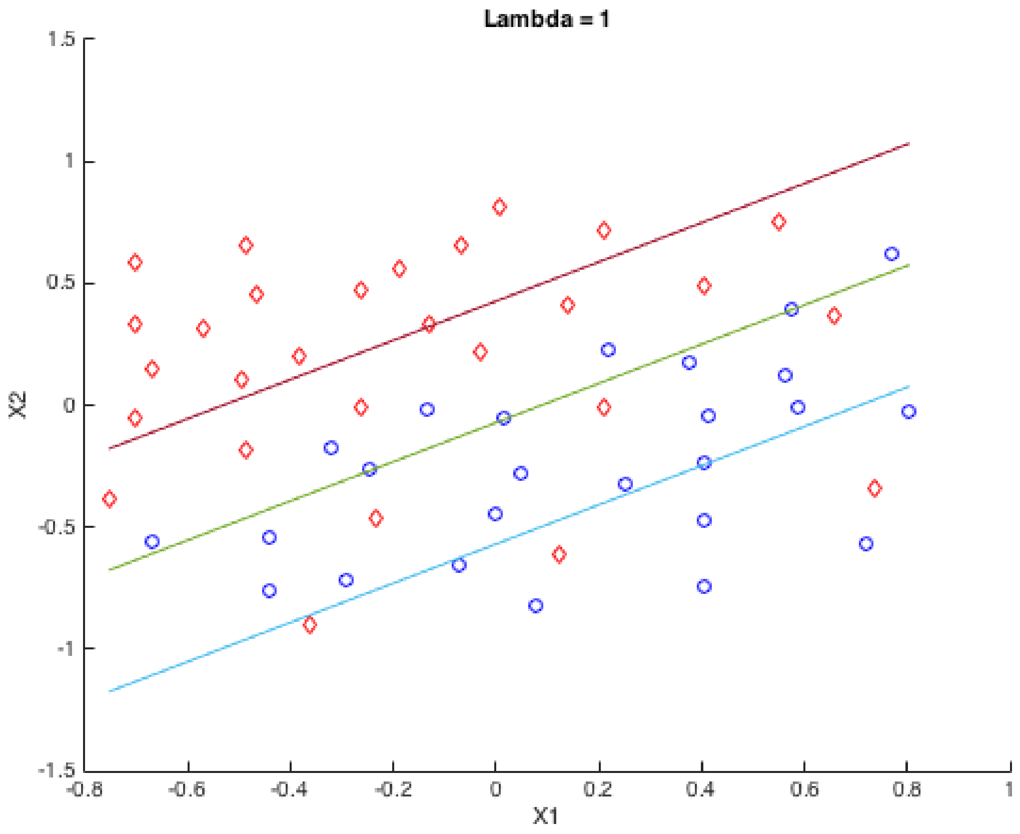
As we can see in the images, the values considered as SVs are the ones with a distance of 1 to the hyperplane.

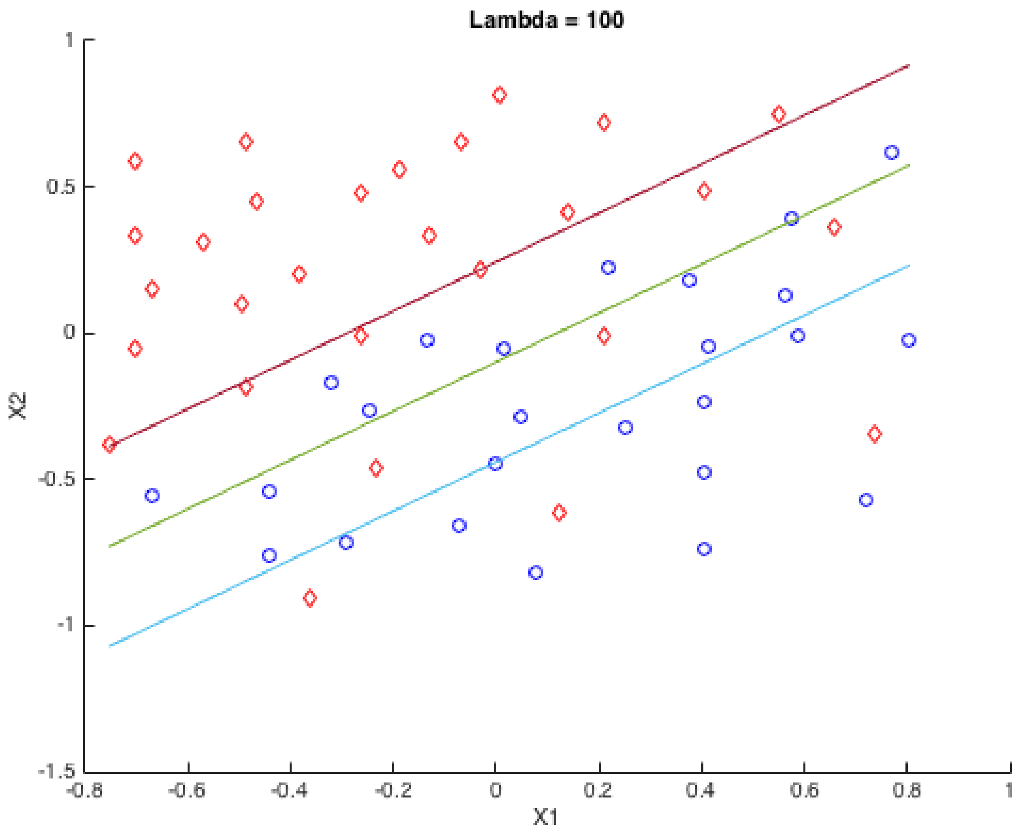
Block 4

```
clear
```

```
load('data/example_dataset_2.mat');
lbda = 0.5;

[ w, aa, b, upp, low ] = svm_soft_dual(data, labels, 1e-2);
svm_draw(data,labels,w,b, upp, low);
title('Lambda = 0.001');
[ w, aa, b, upp, low ] = svm_soft_dual(data, labels, 1);
svm_draw(data,labels,w,b, upp, low);
title('Lambda = 1');
[ w, aa, b, upp, low ] = svm_soft_dual(data, labels, 1e2);
svm_draw(data,labels,w,b, upp, low);
title('Lambda = 100');
```







The v of the support vectors are those which have an alpha different of 0. The rule to identify the SV in the dual is stronger.

We considered easier to identify the SVs in the dual. In the primal, we found some values that could be considered or not, so, it had a little ambiguity in some cases.

In dual, the weights obtained from the value V can be considered in order to show whether an instance should be considered or not in the SVM.

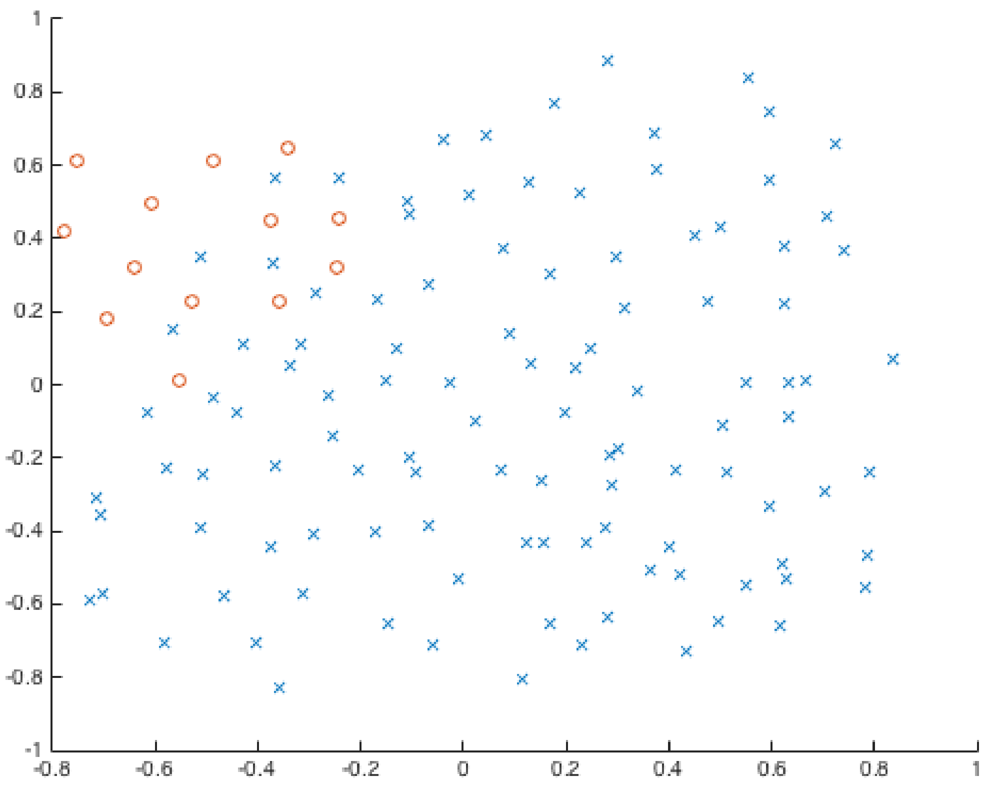
Block 5

```
clear
load('data/example_dataset_3.mat');
sum(labels==-1)
sum(labels==1)
```

```
ans =

    13


ans =

   110
```

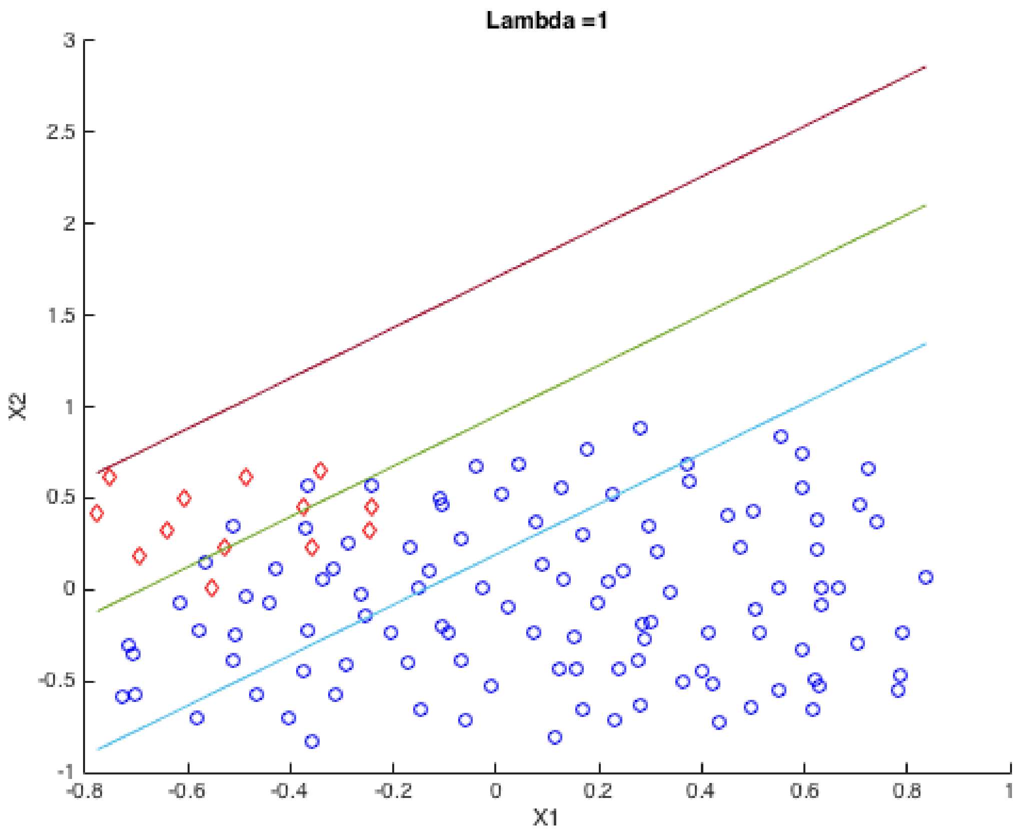
The number of items in the class -1 is 13, the number of items in the class 1 is 110.

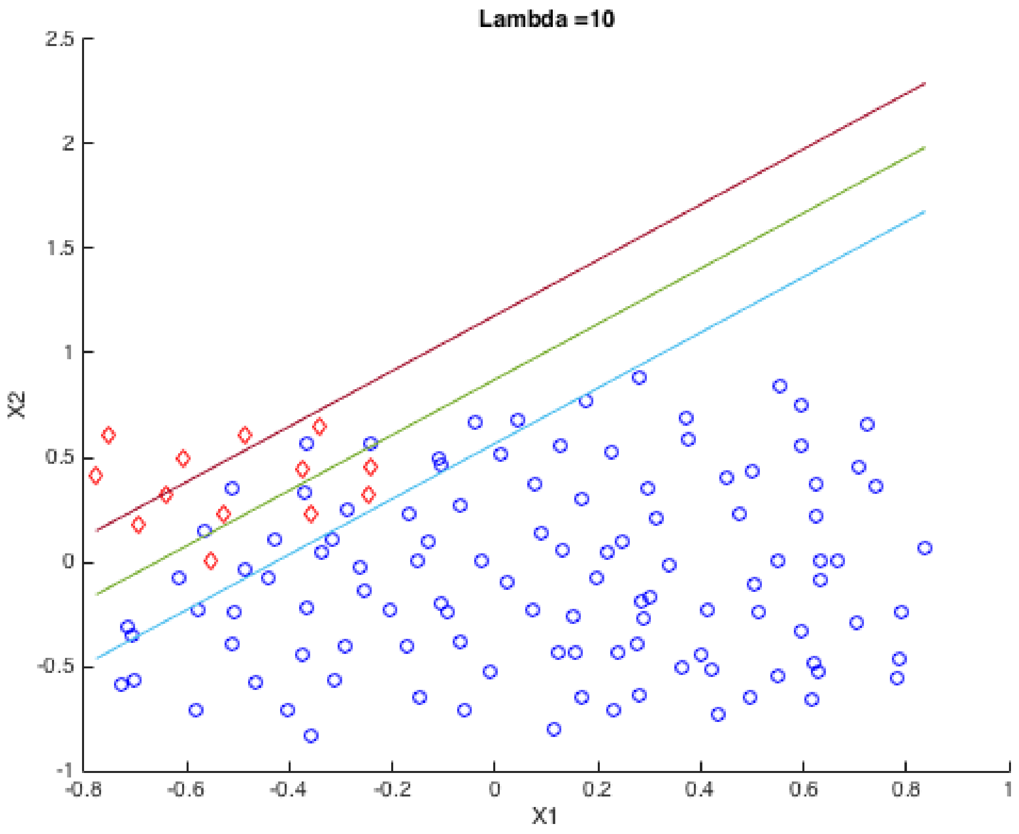
```
figure;
hold on
scatter(data(1, labels==1), data(2, labels==1), 'x');
scatter(data(1, labels==-1), data(2, labels==-1), 'o');
hold off
```

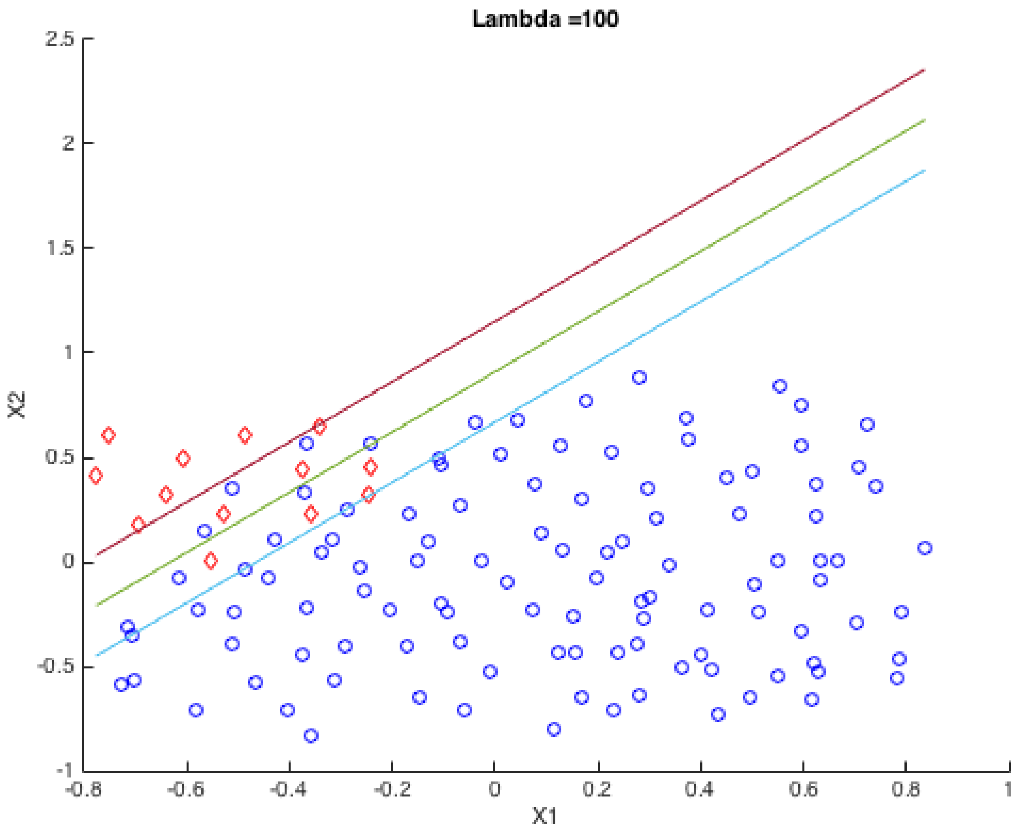


We can consider this dataset as unbalanced.

```
a = [1 10 1e2]';
hr = zeros(size(a));
hi = zeros(size(a));

for i=1:size(a,1)
    [ w, aa, b, upp, low ] = svm_soft_dual(data, labels, a(i));
    svm_draw(data,labels,w,b, upp, low);
    title(strcat('Lambda = ',num2str(a(i))));
    [j, hit] = svm_test(w, b, data, labels);
    hr(i) = hit;
    hi = sum(aa);
end
```







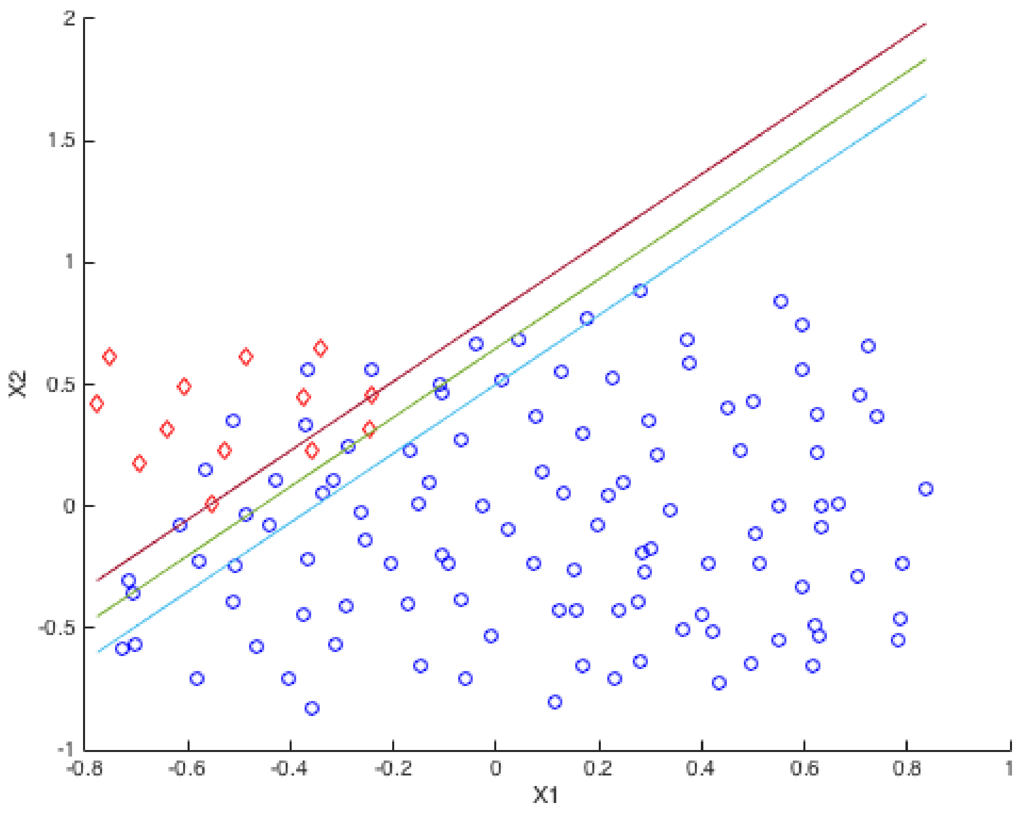
By considering the lambda as 1, 10, 100, 1000... we can see that, from the 10 value on, the errors and the missclassification errors are not incremented heavily, so in this case we can consider as a good value the lambda = 10. In a lambda of 1 we had 17 errors and 8 missclassification errors also, but from lambda 10 and greater lambda, the values keep around 14 errors and 7 missclassification errors.

It is not the best result, but can be considered as a good value, because the missclassification errors were around 6,6%. Given the dataset, and considering that is a non-linearly separable set, the results were quite reliable.

As it's said in the previous paragraph, the training error rate is 6.65%.

Block 6

```
clear
load('data/example_dataset_3.mat');
[ w, b, upp, low, ba, bb ] = svm_soft_ext(data, labels, 100);
svm_draw(data,labels,w,b, upp, low);
```



Considering the possible values of lambda as 1, 10, 100, 1000... we found that from the 100 and upper values, the SVM errors and the missclassificationerrors kept stuck at 22 and 13 repectively. So, we consider as the optimum lambda as 100.

Considering the result as a satisfying result depends on the objective of the SVM, it is satisfying when the objective is to classify correctly the negatives, and the positives are not so important as before (by comparing to the block 5). So, in this example, the results are quite good considering that it is a weighed SVM.

The error rate is higher, this is because there are much 1 class samples bad classified, instead all the -1 class samples are in the same class.

Now let's compare the weighted error on block 5 and block 6:

```
[ w, b, upp, low, ba, bb ] = svm_soft_ext(data, labels, 100);
[~, pct] = svm_test(w,b,data,labels, ba, bb)
[ w, aa, b, upp, low ] = svm_soft_dual(data, labels, 10);
[~, pct] = svm_test(w,b,data,labels, ba, bb)
```

```
pct =

    0.1042


pct =

    0.0267
```

As we can see, while the weighted error in the block 5 is above 10%, using this approach it's reduced to the 2%. This happens because with the balanced weight, a missclassified sample of the majoritary class doesn't penalize as much as one of the minoritary class.

Published with MATLAB® R2015a