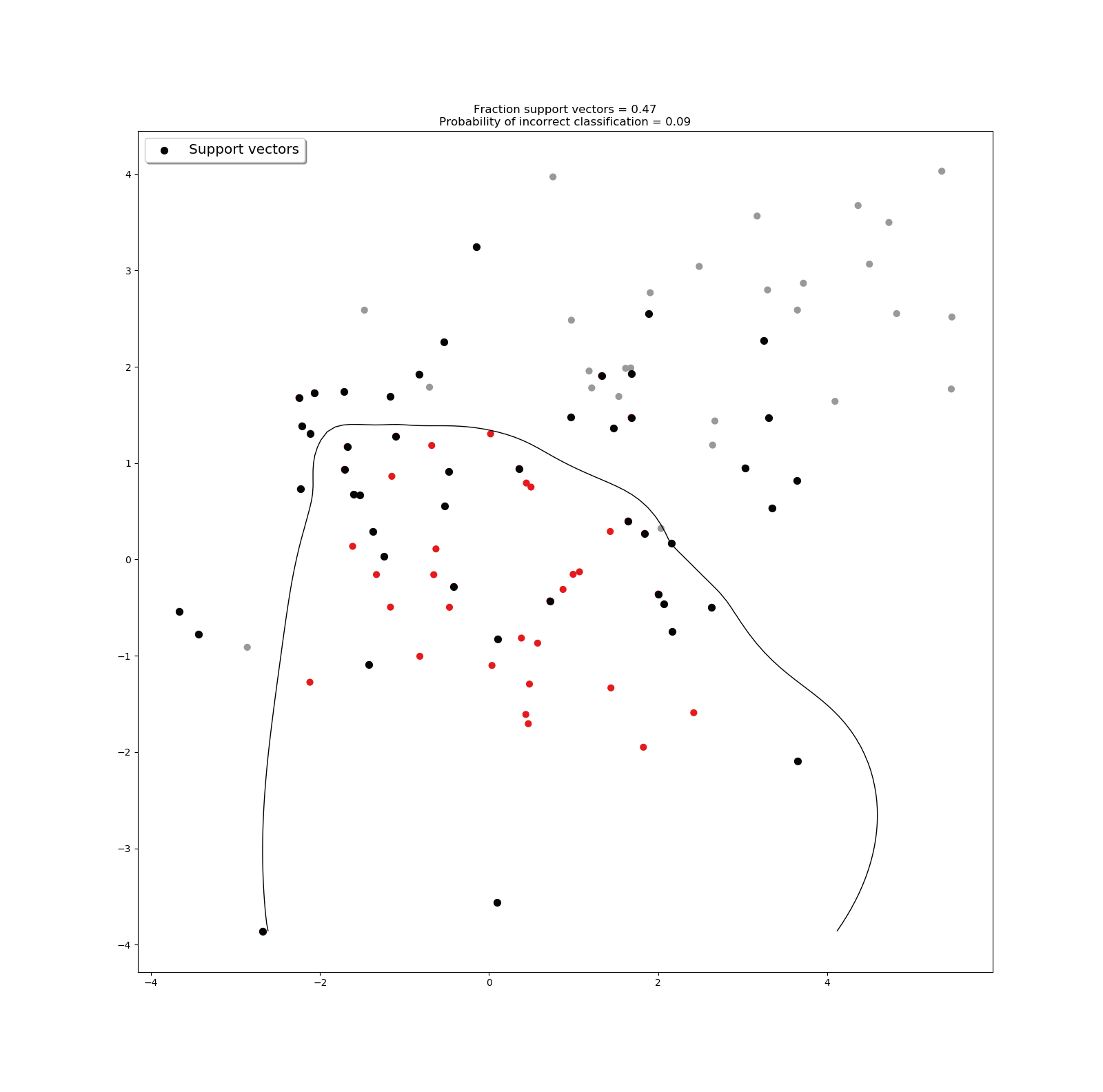
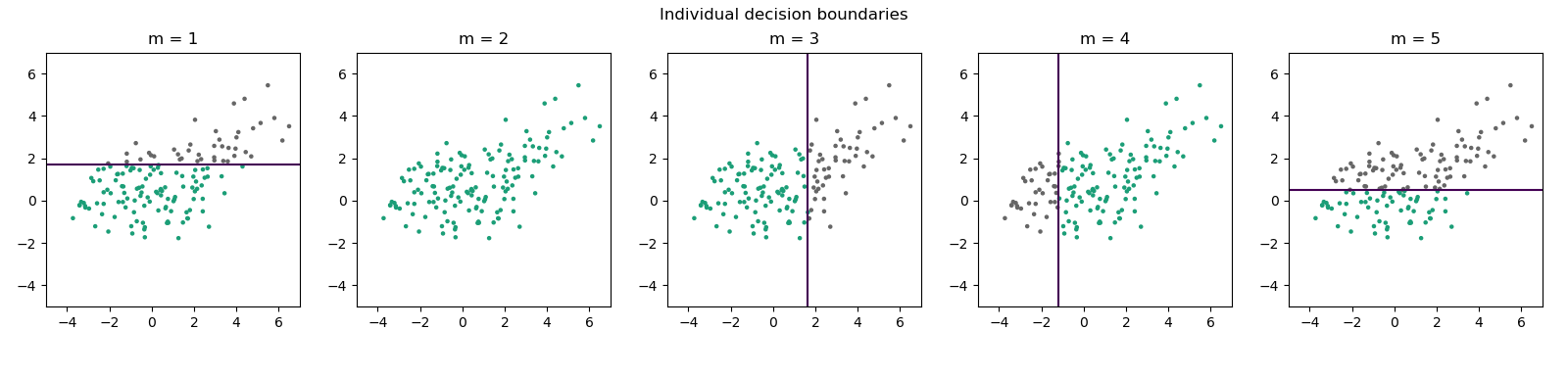
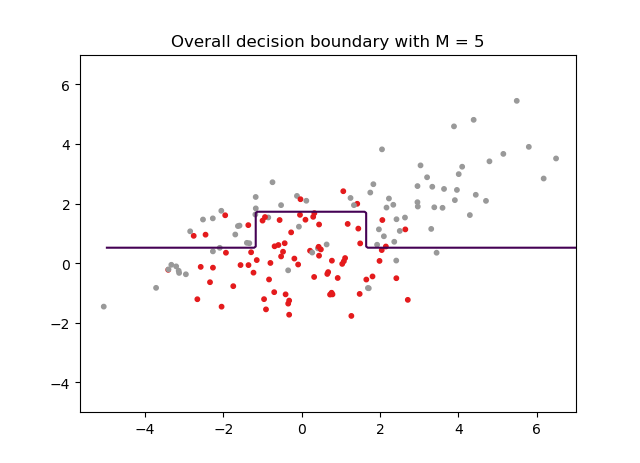
# Homework 4 ECE 283 Morten Lie

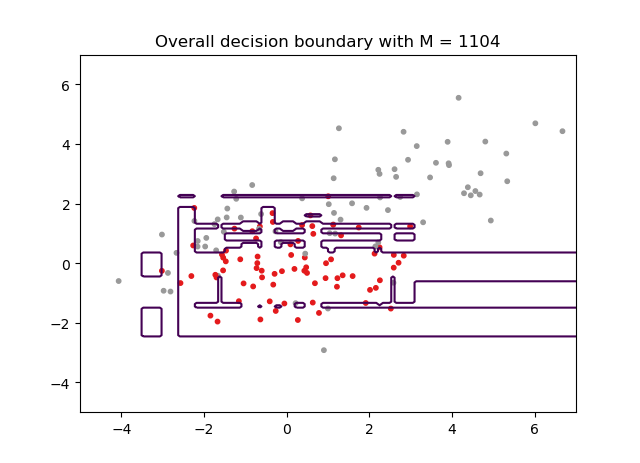
**Problem 1**1.1)  
Decision boundary is displayed in the figure below. 47% of the dataset were support vectors when the algorithm was used with a dataset of N=100 datapoints in the training set.   


1.2)  
The misclassification that was accomplished from the SMO algorithm SVM was lower than what was experienced with the kernelized logistic regression in HW1. The implementation is displayed in the appendix.

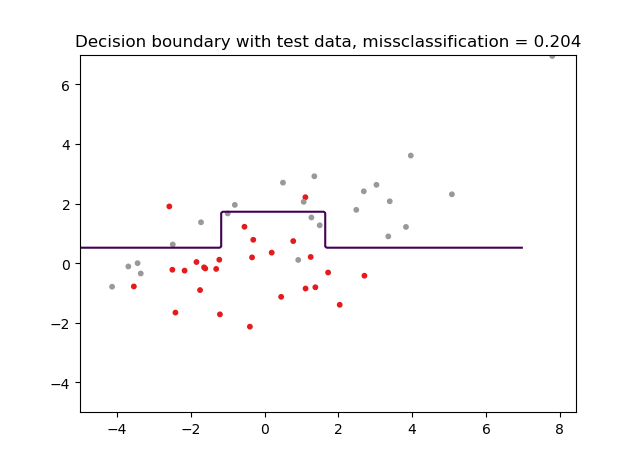
**Problem 2**1.1)  
The first figure displays the decision boundaries created for m={1,5}. The second figure displays the resulting decision boundary created when the weights are applied to the decision boundaries displayed in the first figure. The dataset what was used had 200 data points, where 75% were training data and the remaining 25% were used for testing data.

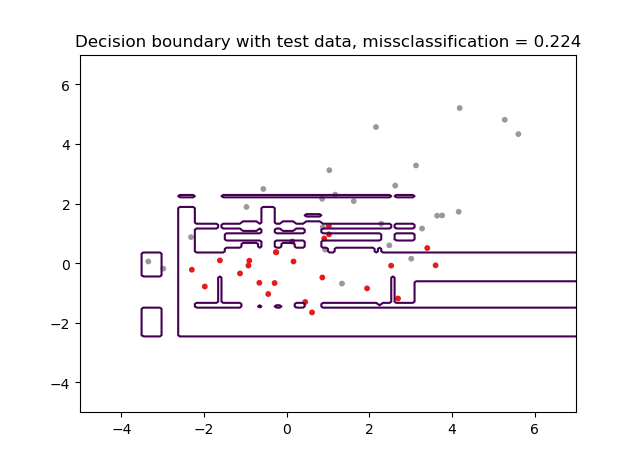




1.2)  
When running the algorithm until the error vanishes yielded M = 1104. In this case there is a lot of overfitting, which is also reflected in the misclassification. We can notice that the algorithm frames single data points, which happens to have the right label for the training data, but could very well have the wrong label for the test data.   


1.3)  
Using the testing data set on both implementations; where M is set to 5 and where M goes until there is no more error, is displayed in the next two figures.





1.3)   
Notice that the misclassification is 0.204 and 0.224 for the two implementations, respectively. Both of these performances are worse than what we got from the SVM algorithm. But then again, there is room for improvement on the implementation of Adaboost. Using a set M could be hard to optimize, but then again, having an arbitrary M could lead to overfitting. If we could develop an algorithm that would be able to monetarize the overfitting as the algorithm was running, and breaking the optimization in real time, one would expect better results. When we compare all the implementations we have done so far in this quarter, Adaboost has been performing good as far as computational time goes. To be far, the previous implementations that I have done have been in Matlab, so it can be hard to compare. But upon discussion with my homework group, we agreed that this was the case. Further, neural network performed well too, with a lot of room for improvement as we created a shallow network that could also be improved. In my implementations of logistic regression, this was the algorithm that turned out to perform worse.