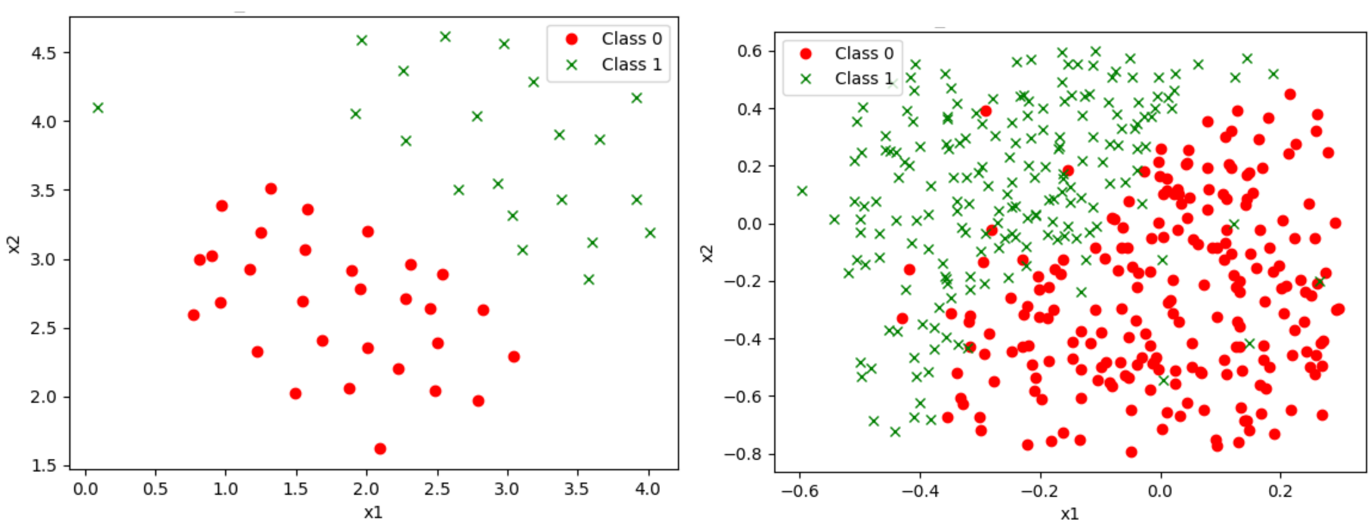
Understanding Linear Support Vector Machines (SVM)s

Sami Alperen Akgun
sami.alperen.akgun@gmail.com

I. Understanding Linear Support Machines

A. *Visualization of Data*

The visualization of datasets dataset1 and dataset2 can be seen in Figure 1.



(a) Dataset1 (b) Dataset2

Fig. 1. Data used to understand Linear SVMs

B. *Train a linear SVM using Python Sklearn*

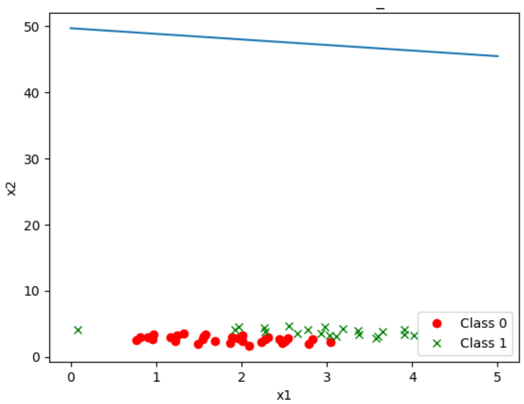
To train a linear support vector machine (SVM) with different penalization factors, i.e. different c values, Python machine learning library scikit-learn (sklearn) was employed[1]. One can see an example usage below:

```
from sklearn import svm

X, Y = read_data(data_dir,"dataset1.csv",2)

clf = svm.SVC(kernel='linear', C = 0.01)
# Train SVM classifier
clf.fit(X,Y)
# Visualize decision boundary

weights = clf_[0]
m = -weights[0] / weights[1]
x_axis = np.linspace(0,5,100)
y_axis = m * x_axis - clf.intercept_[0] / weights[1]

fig1 = plt.figure()
plt.plot(x_axis, y_axis)
plt.show()
plt.close()
```

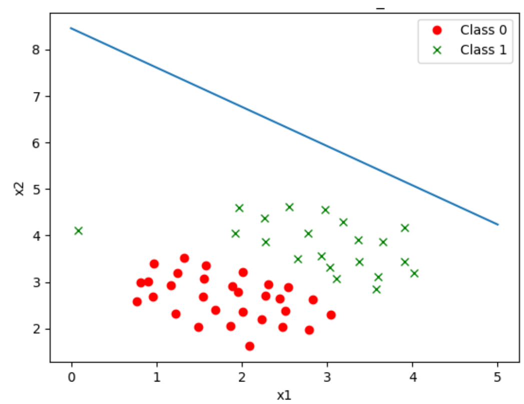
C. *Use Different c Values and Plot Decision Boundaries to Compare*

Different decision boundaries corresponding to different penalization factors, c values (0.001, 0.01, 0.1 , 1), were visualized to offer an intuitive comparison. Resulted plots for first and second dataset can be seen in Figure 2 and 4 respectively. Decision boundaries with margins (resulted support vectors) for first and second dataset can be seen in Figure 3 and 5 respectively.

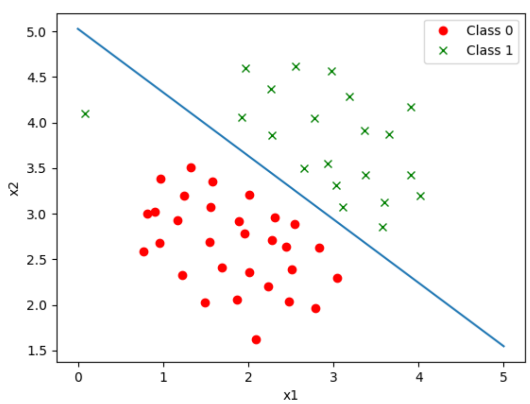
[1]https://scikit-learn.org/stable



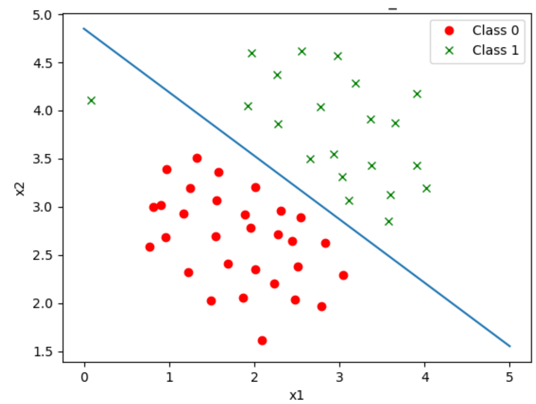
(a) c=0.001



(b) c=0.01



(c) c=0.1



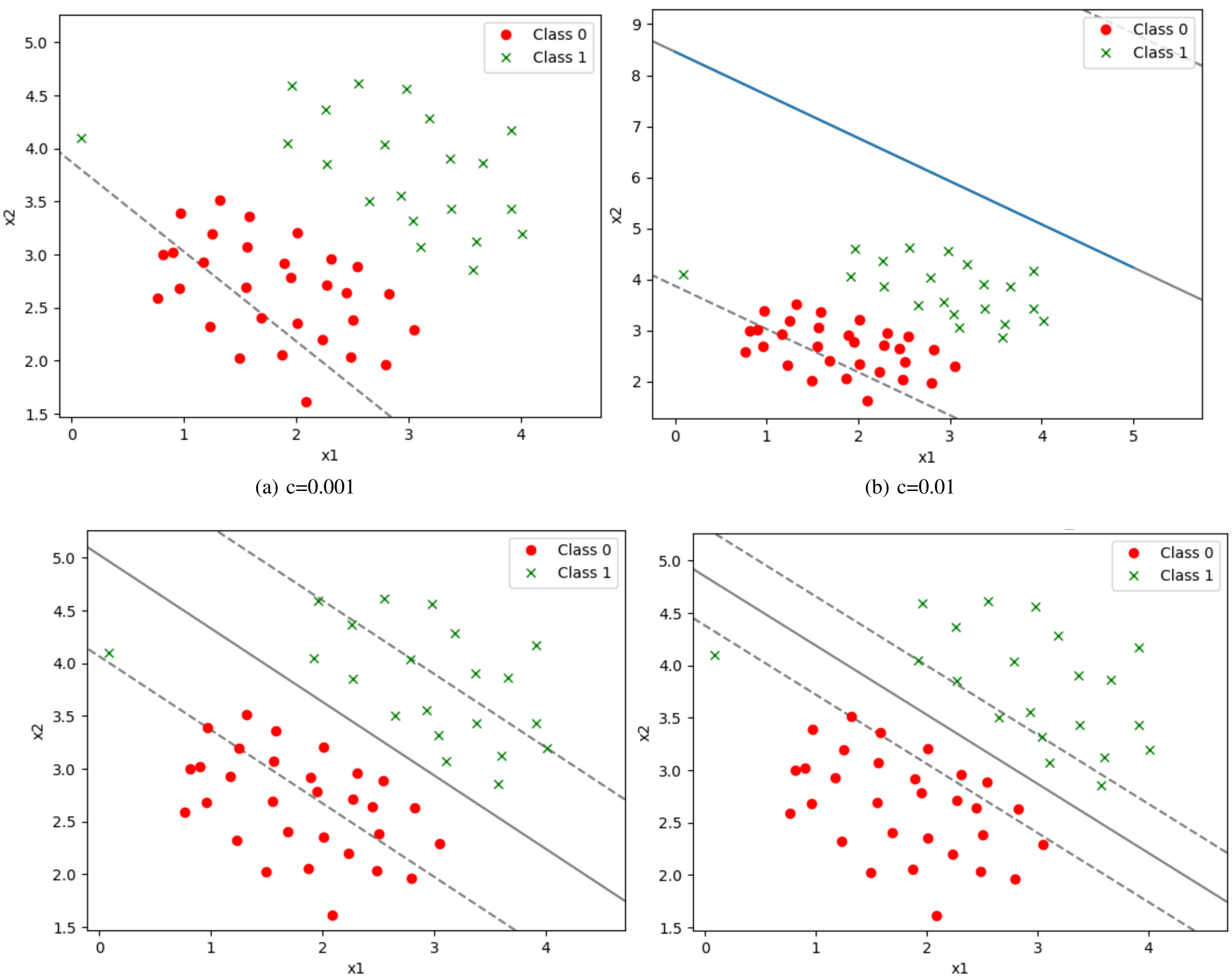
(d) c=1

Fig. 2. Different decision boundaries for Dataset1

D. *Select Best c Value (Penalization Factor) among trained SVMs*

The effect of penalization factor c is to control the trade-off between complexity of SVM classifier and miss-classification error. When c value is selected high, the classifier tries to minimize the miss-classification error as much as possible. Nonetheless, this might lead to over-fitting. On the other hand, smaller values of c mean to select classifiers with lower complexity, i.e. lower model order, which is a good way to eliminate over-fitting. However, if c is selected so low, then underfitting might occur. Hence, the value of c usually found thanks to fine tuning process with cross-validation approach.

In this assignment, best c value would be 0.1 for dataset1 among given c values since it is the lowest c value that provides the best performance regarding misclassification error. As it can be seen in Figure 2, selecting a larger c (c=1) will not effect the current classification error. It will only increase the complexity unnecessarily which will lead to higher convergence time and possible over-fitting. This also means that the trained classifier won't be generalizable for possible new data. On the other hand, best c value would be 1 for dataset2 instead of 0.1 because as it can be seen in Figure 4, miss-classification error is minimized for c=1 case. This can be actually expected, since dataset2 has higher number of training samples, so it logically requires a classifier with higher complexity (model order) to correctly classify given training data.



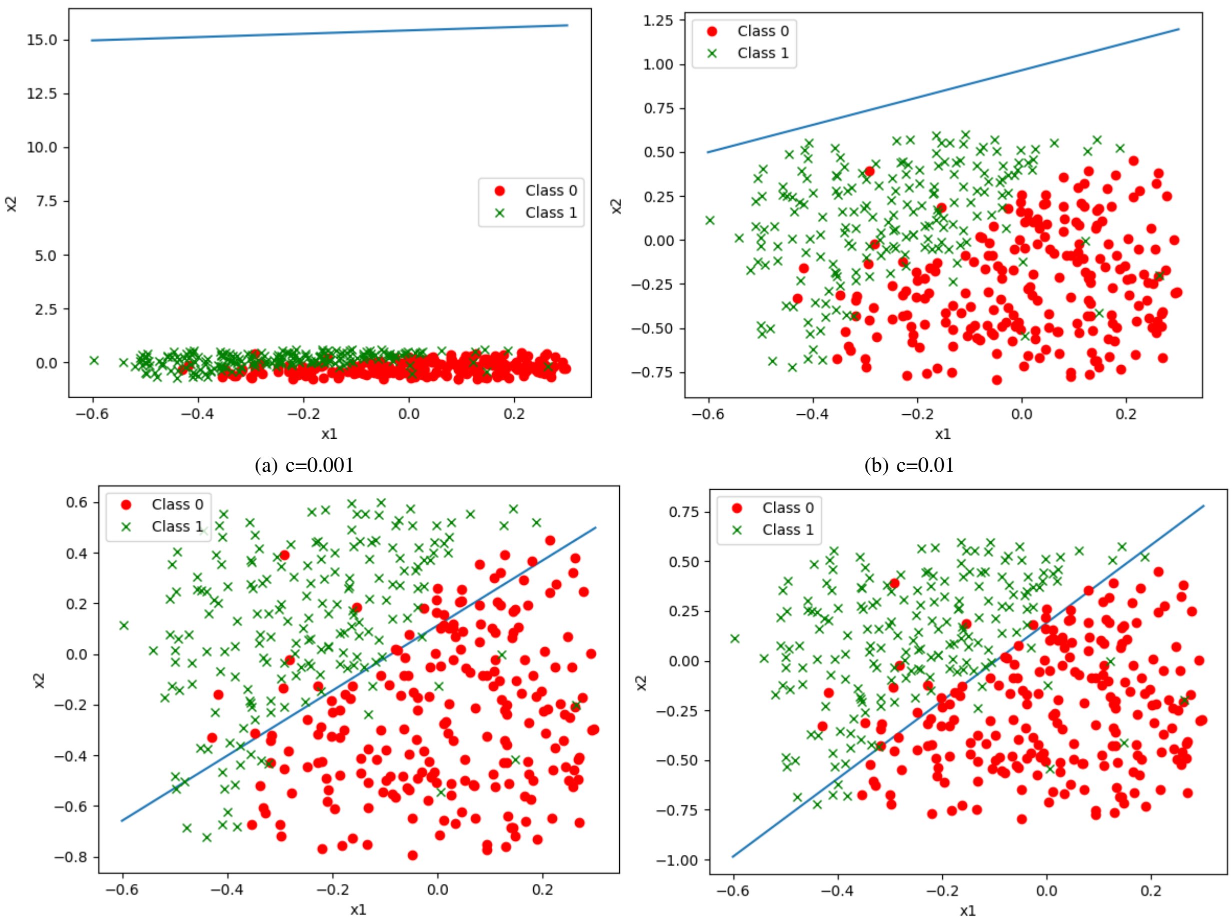
(a) c=0.001

(b) c=0.01

(c) c=0.1

(d) c=1

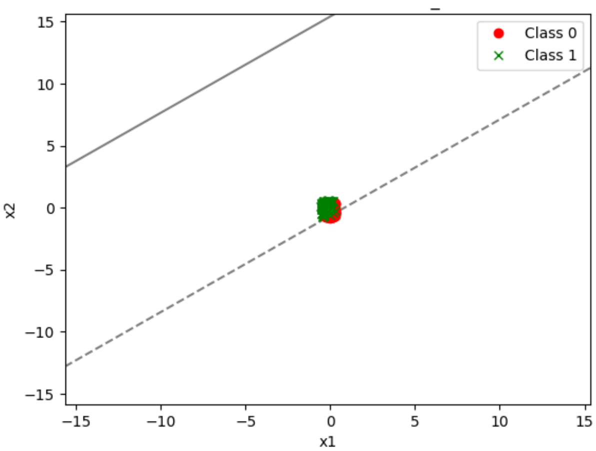
Fig. 3. Different decision boundaries with margins for Dataset1



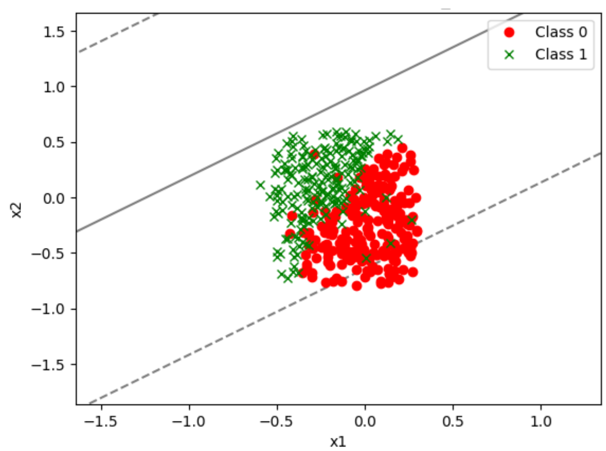
(a) c=0.001 (b) c=0.01

(c) c=0.1 (d) c=1

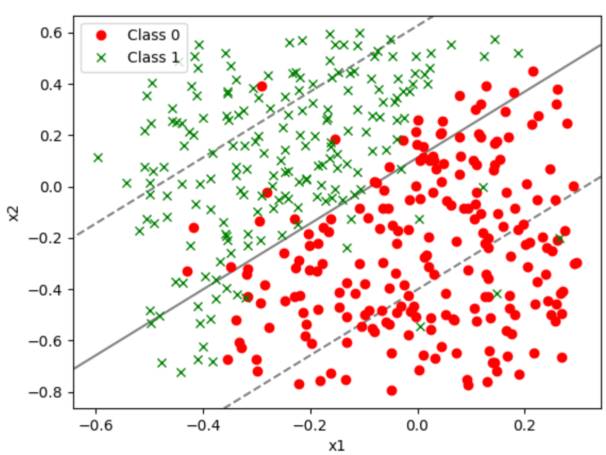
Fig. 4. Different decision boundaries for dataset2



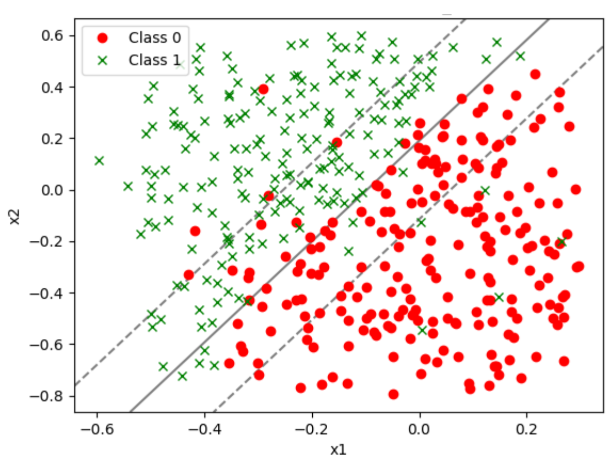
(a) c=0.001



(b) c=0.01



(c) c=0.1



(d) c=1

Fig. 5. Different decision boundaries with margins for dataset2