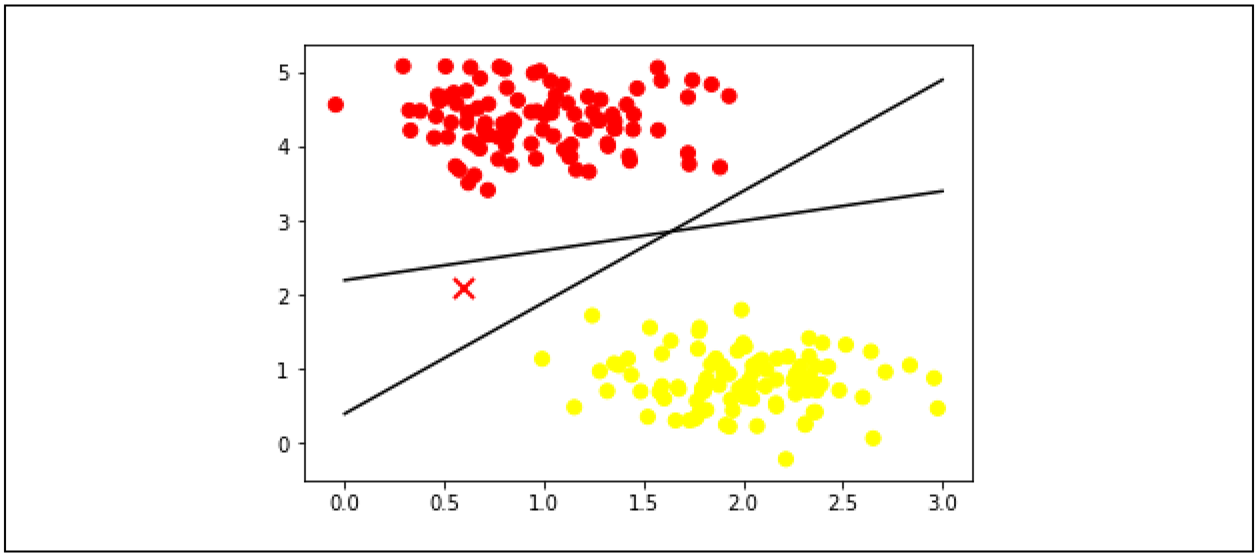
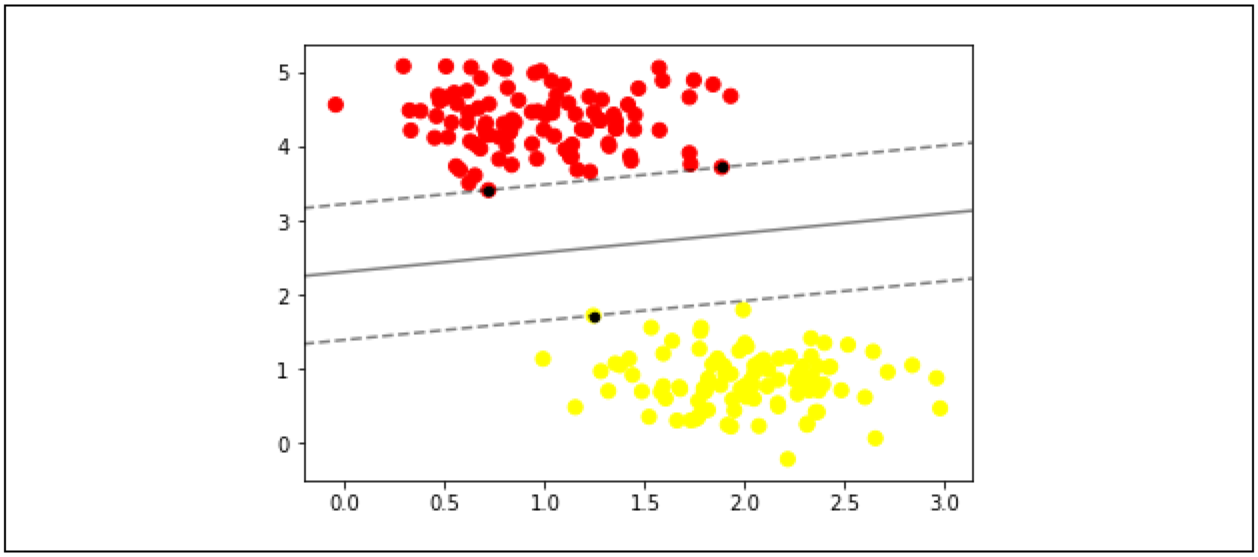
2) Linear SVM

- The first line: [0.4, 2.2], The second line: [1.5, 0.4]
-



The first line is better since it has a larger (better) margin. The distance of the line with the closest samples is larger in the first-line case; the will lead to a more robust classification.

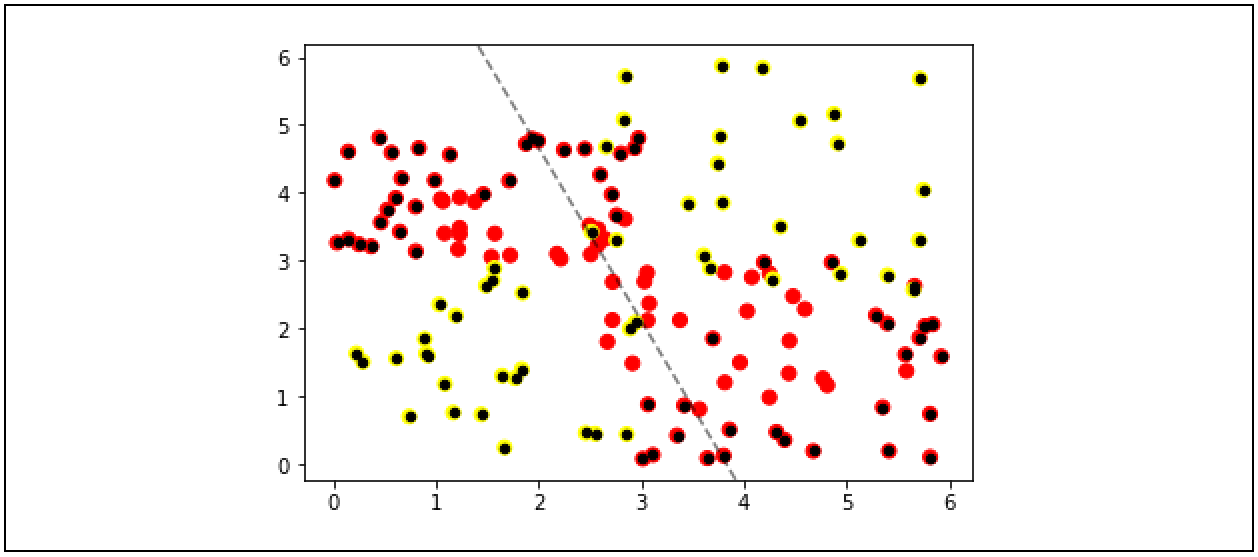
-



Three. They are the samples (vectors) closest to the discriminator line and on the boundaries.

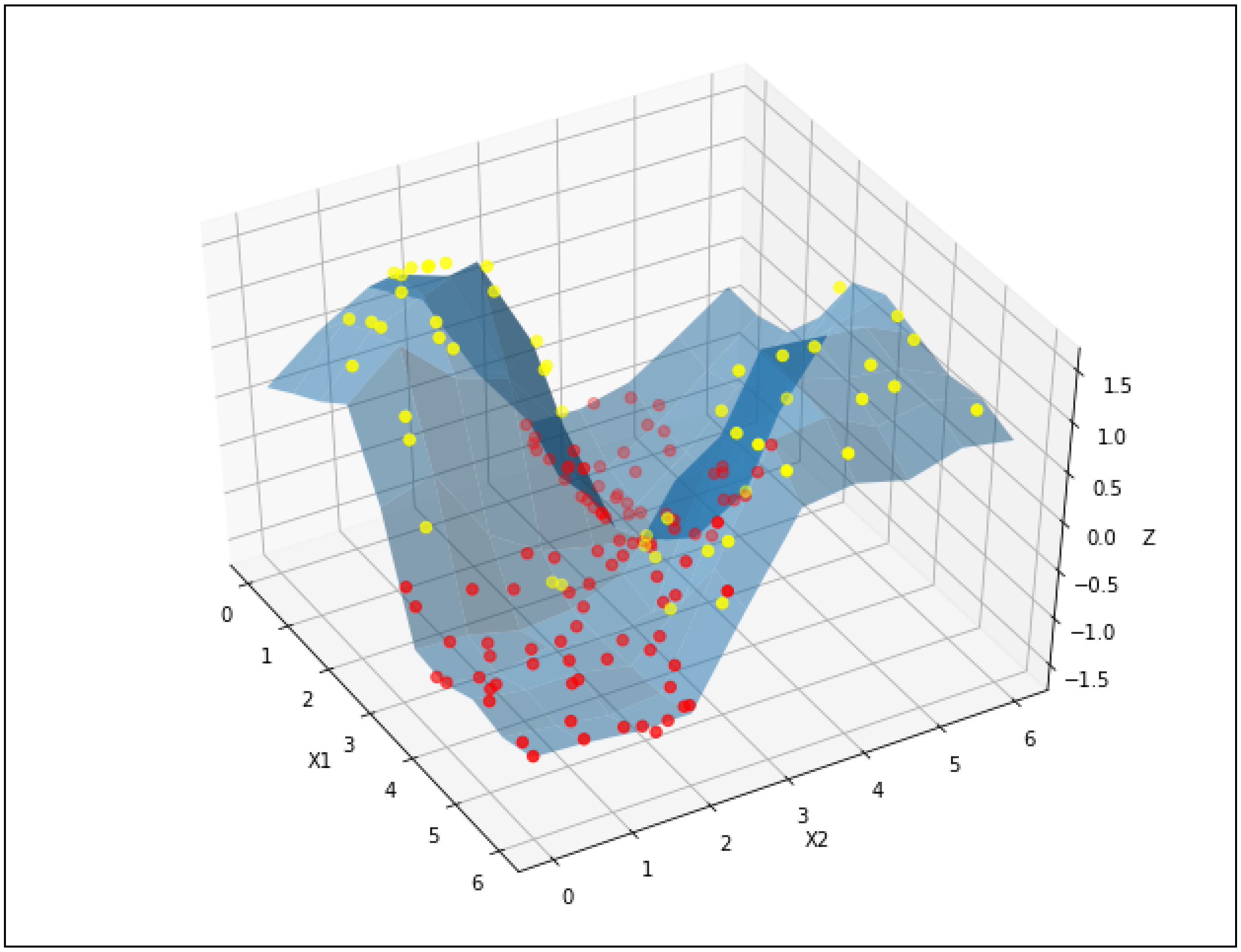
3) Kernel SVM

- The result:



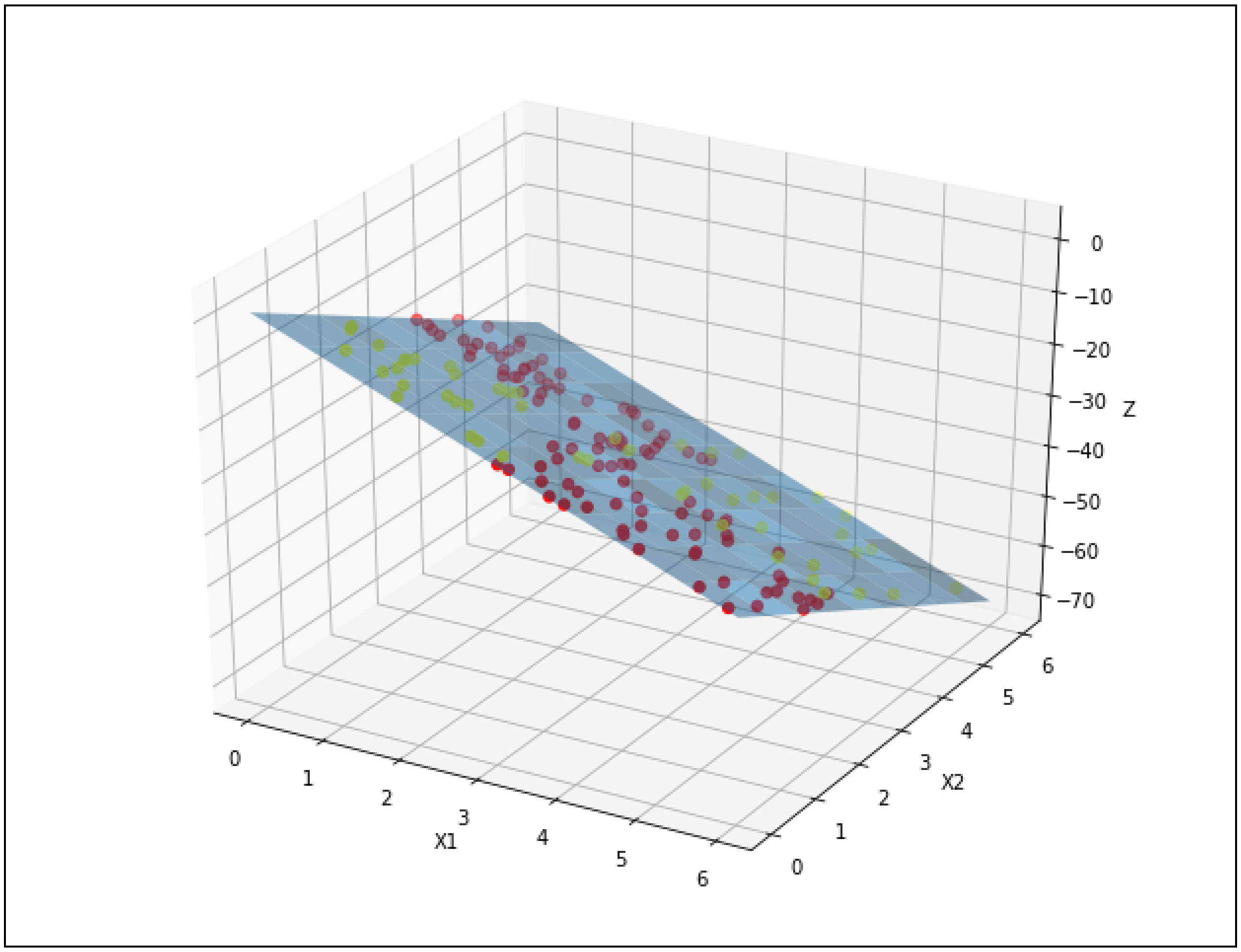
The result is not good since the data is not linearly separable, so linear SVM does not work well on such data.

- The result:



They are very close to the surface. Like the surface, the red points are usually below plane Z=0 and yellow points are usually above the plane Z=0.

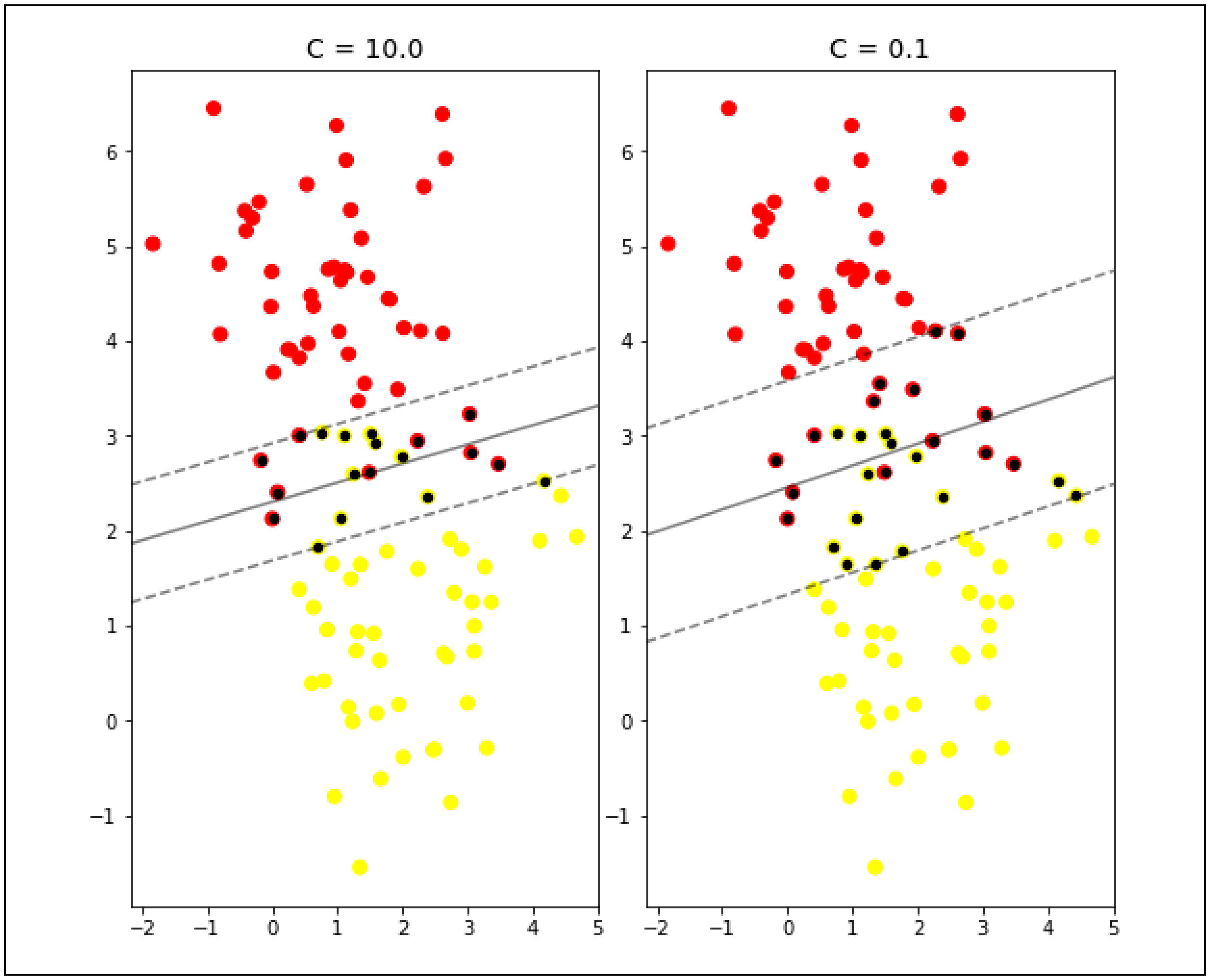
-



The surface is strait, and still, two planes are needed for classification. It seems that red points, unlike yellow points, usually are between plane Z=20 and Z=40.

4) Softening the margin

- The result:



The results, the learnt models, are slightly different. The margins (the distance of the borders to the discriminator line), as well as the number of support vectors, is larger in smaller C (C=0.1).

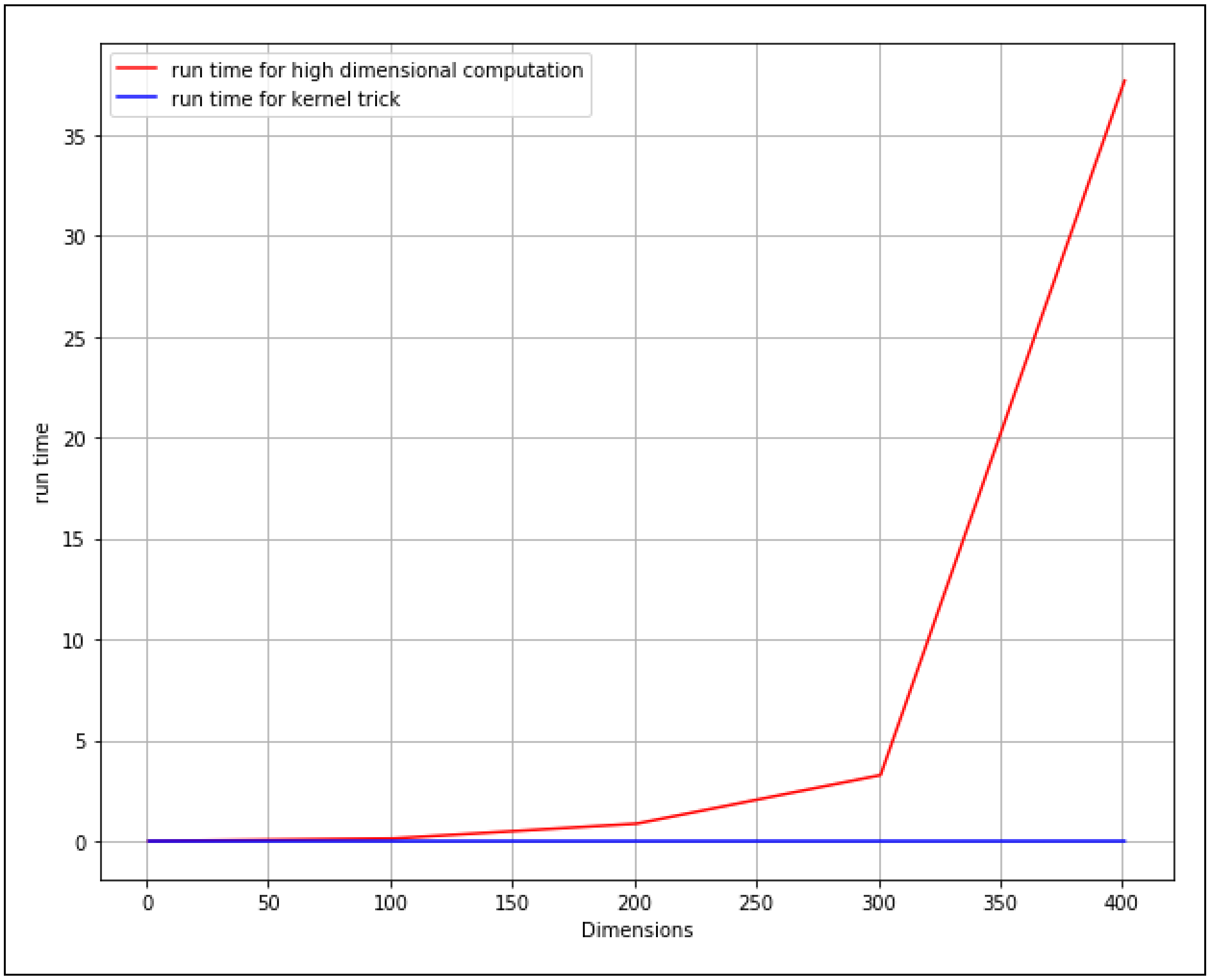
- Please see the notebook in the last section

```
X_train, X_test, y_train, y_test =  train_test_split( iris.data,  iris.target, test_size = 0.25, random_state = 0)
```

- The best value for C is :1.200
  The accuracy: 0.9736842105263158

5) Kernel Trick

- Please see the notebook in the last section
- Please see the notebook in the last section



- For larger dimensions, the computational load and consequently the runtime for the first procedure (high dimensional computation) is higher, and when the number of attributes in data grows the computational load grows dramatically.