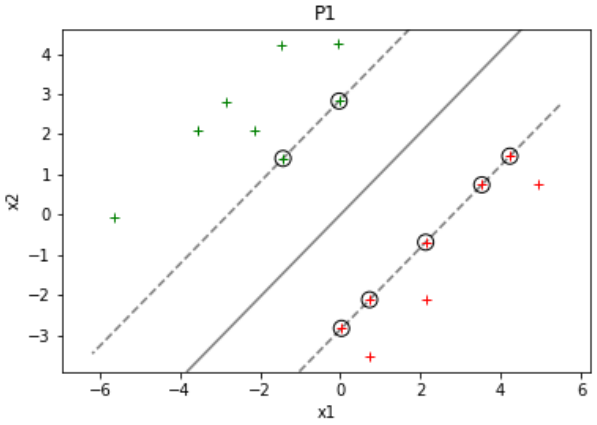
11752 Machine Learning
Master in Intelligent Systems
Universitat de les Illes Balears

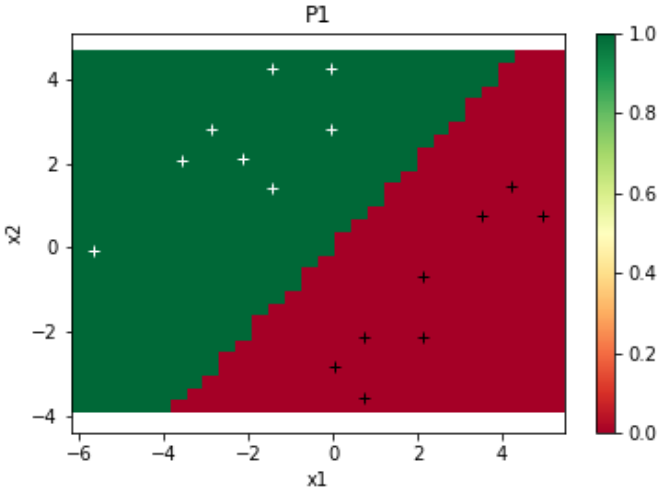
These are the kind of graphical results which are expected for T1.b and T1.c (linear SVM), T2.b and T2.c (non-linear SVM), etc.

LINEAR CASE

example of plot with training samples and the 2D decision curve, highlighting the support vectors

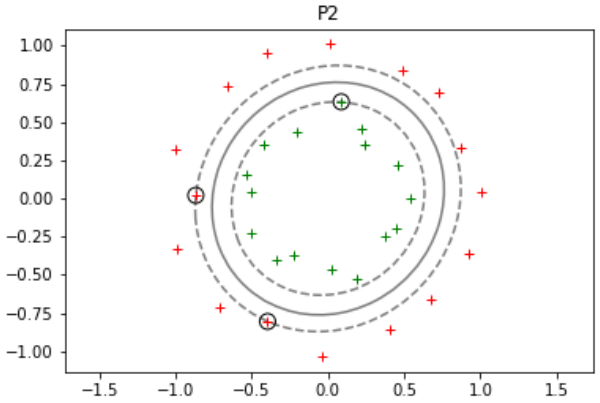


example of classification map (in this case, label 0/*black crosses* corresponds to the class at the "negative" side of the hyperplane)

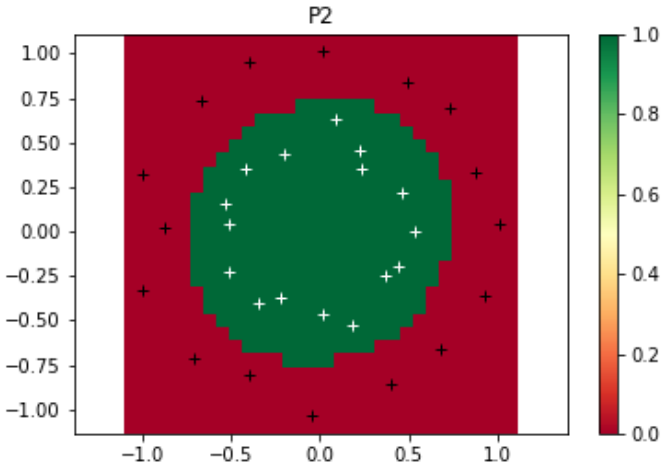


NON-LINEAR CASE

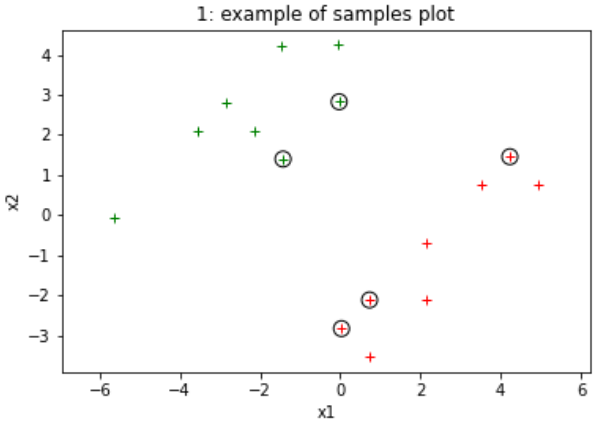
example of plot with training samples and the 2D decision curve, highlighting the support vectors

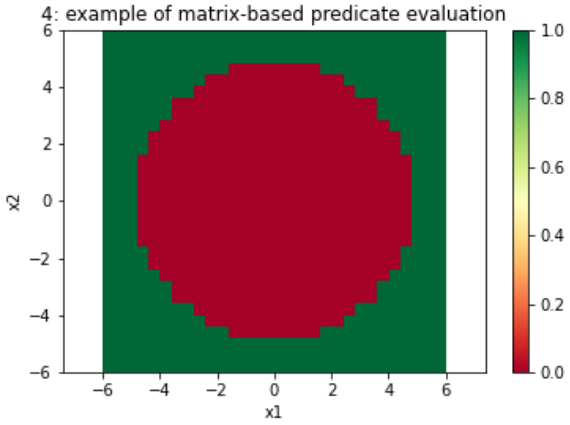


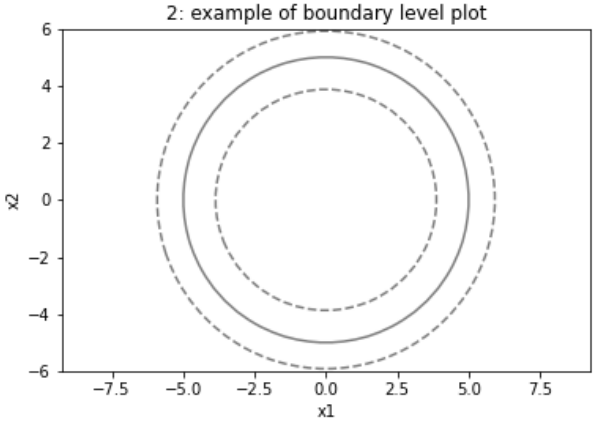
example of classification map (in this case, label 0/*black crosses* corresponds to the class at the "negative" side of the hyperplane)

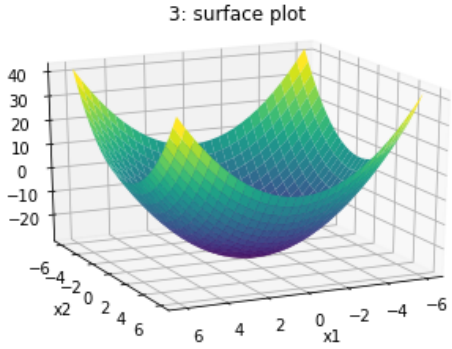


By way of example, the next plots:









have been generated by means of the following source code (assuming the dataset X has been properly loaded):

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# plot samples, highlighting some of them
plt.figure(1)
plt.plot(X[0:8,0],X[0:8,1],'+r')   # class w1
plt.plot(X[8:16,0],X[8:16,1],'+g') # class w2
hil = [1,  2,  5, 10, 11] # samples to highlight
ax = plt.gca()
ax.scatter(X[hil,0], X[hil,1], s=100, linewidth=1, facecolors='none', edgecolors='k')
plt.xlabel('x1')
plt.ylabel('x2')
plt.axis('equal')
plt.title('example of samples plot')
plt.show(block=False) # to force visualization

# create grid to evaluate function
xx = np.linspace(-6, 6, 30)
yy = np.linspace(-6, 6, 30)
YY, XX = np.meshgrid(yy, xx)
```

```
Z = np.zeros((30 * 30,1))
k = 0
for x1 in xx:
    for x2 in yy:
        Z[k] = x1 ** 2 + x2 ** 2 - 30
        k += 1

# plot boundary of levels -15, -5 and +5
plt.figure(2)
ax = plt.gca()
ax.contour(XX,YY,Z.reshape(XX.shape),colors='k',levels=[-15, -5, 5],alpha=0.5,linestyles=['--', '-', '--'])
plt.xlabel('x1')
plt.ylabel('x2')
plt.axis('equal')
plt.title('example of boundary level plot')
plt.show(block=False) # to force visualization

plt.figure(3)
ax = plt.axes(projection='3d')
ax.plot_surface(XX, YY, Z.reshape(XX.shape), rstride=1, cstride=1, cmap='viridis', edgecolor='none')
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('surface plot')
plt.show(block=False) # to force visualization

# matrix-based predicate evaluation
C = np.where(Z >= -5, 1, 0)
plt.figure(4)
plt.imshow(C.reshape(XX.shape), origin='lower', extent=(-6, 6, -6, 6), cmap='RdYlGn')
plt.colorbar()
plt.xlabel('x1')
plt.ylabel('x2')
plt.axis('equal')
plt.title('example of matrix-based predicate evaluation')
plt.show(block=False) # to force visualization
```