M4-L2 Problem 1

Now you will try support vector classification on data with nonlinear decision boundaries. You will use the sklearn SVC tool on four datasets. Your job is to find an appropriate choice of kernel and regularization strength that does a qualitatively good job separating the data.

Run this cell first:

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from matplotlib.colors import ListedColormap

# Plotting functions:
def plot_data(X,c,s=30):
    lims = [0,1]

    markers = [dict(marker="o", color="royalblue"), dict(marker="s", color="red"),

    x,y = X[:,0], X[:,1]
    iter = 0
    for i in np.unique(c):
        marker = markers[iter]
        iter += 1
        plt.scatter(x[c==i], y[c==i], s=s, **(marker), edgecolor="black", linewidth

def plot_SVs(svm, s=120):
    sv = svm.support_vectors_
    x, y = sv[:,0], sv[:,1]
    plt.scatter(x, y, s=s, edgecolor="black", facecolor="none", linewidths=1.5)

def plot_SV_decision_boundary(svm, margin=True,extend=True, shade_margins=False, sh
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xrange = xlim[1] - xlim[0]
    yrange = ylim[1] - ylim[0]

    x = np.linspace(xlim[0] - extend*xrange, xlim[1] + extend*xrange, 200)
    y = np.linspace(ylim[0] - extend*yrange, ylim[1] + extend*yrange, 200)

    X,Y = np.meshgrid(x,y)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = svm.decision_function(xy)

    P = P.reshape(X.shape)
    ax.contour(X, Y, P, colors='k',levels=[0],linestyles=['-'])
    if margin:
        ax.contour(X, Y, P, colors='k',levels=[-1, 1], alpha=0.6,linestyles=['--'])
    if shade_margins:
        cmap = ListedColormap(["white","lightgreen"])
        plt.pcolormesh(X,Y,np.abs(P)<1,shading="nearest",cmap=cmap,zorder=-999999)

    if shade_decision:
        cmap = ListedColormap(["lightblue","lightcoral"])
        pred = (svm.predict(xy).reshape(X.shape) == 1).astype(int)
        plt.pcolormesh(X,Y,pred,shading="nearest",cmap=cmap,zorder=-1000)

    plt.xlim(xlim)
    plt.ylim(ylim)

def plot(Xdata, ydata, svm_model=None, title=""):
    plt.figure(figsize=(5,5))
    plot_data(Xdata,ydata)
    if svm_model is not None:
        plot_SVs(svm_model)
        plot_SV_decision_boundary(svm_model,margin=True,shade_decision=True)
    plt.legend()
    plt.xlabel("$x_1$")
    plt.ylabel("$x_2$")
    plt.title(title)
    plt.show()
```

Loading the data

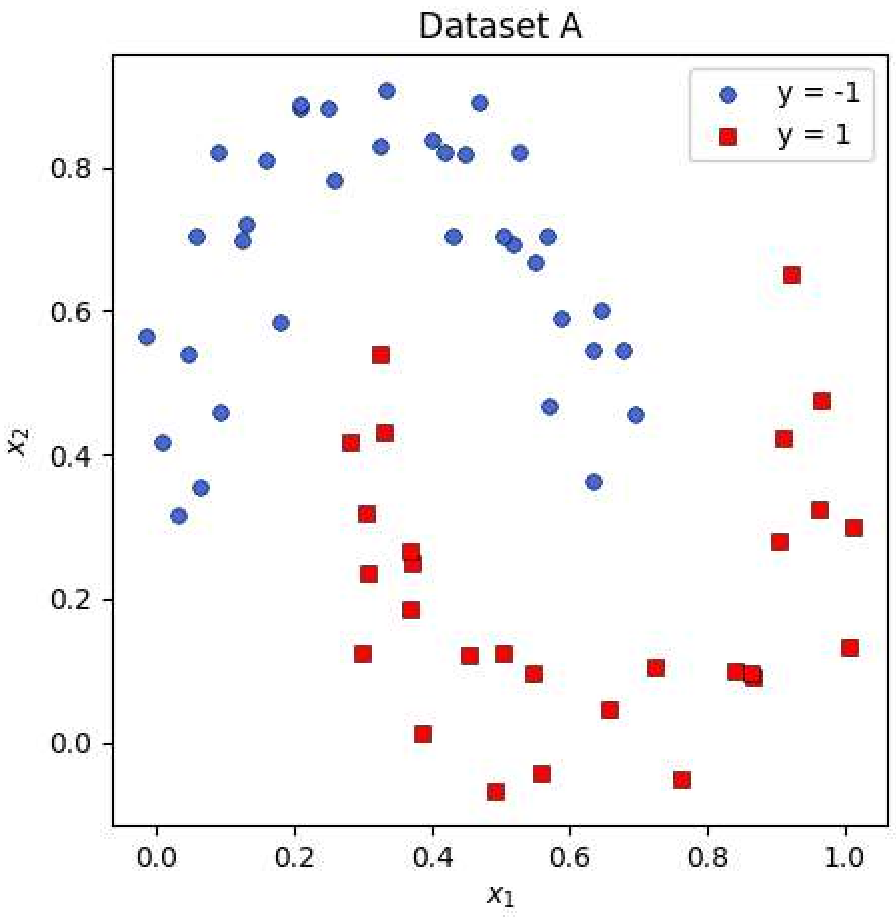
There are four datasets, all 2D and with X and y names as follows:

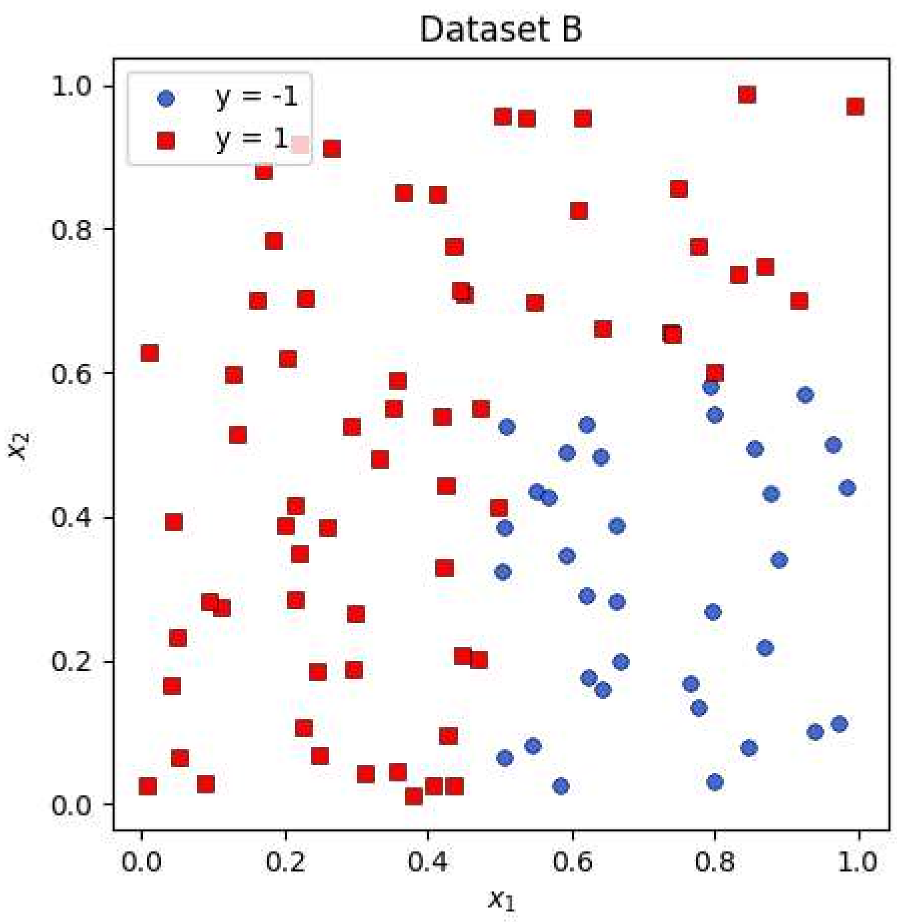
- Xa, ya
- Xb, yb
- Xc, yc
- Xd, yd

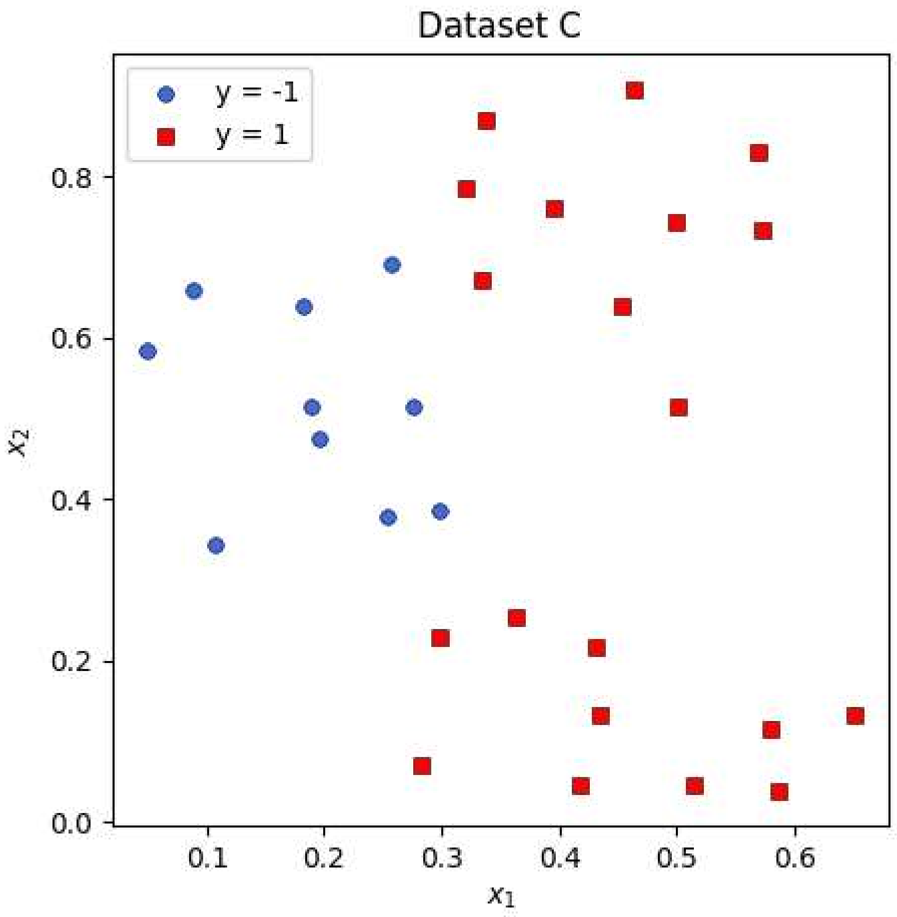
Run this cell to load and plot the data:

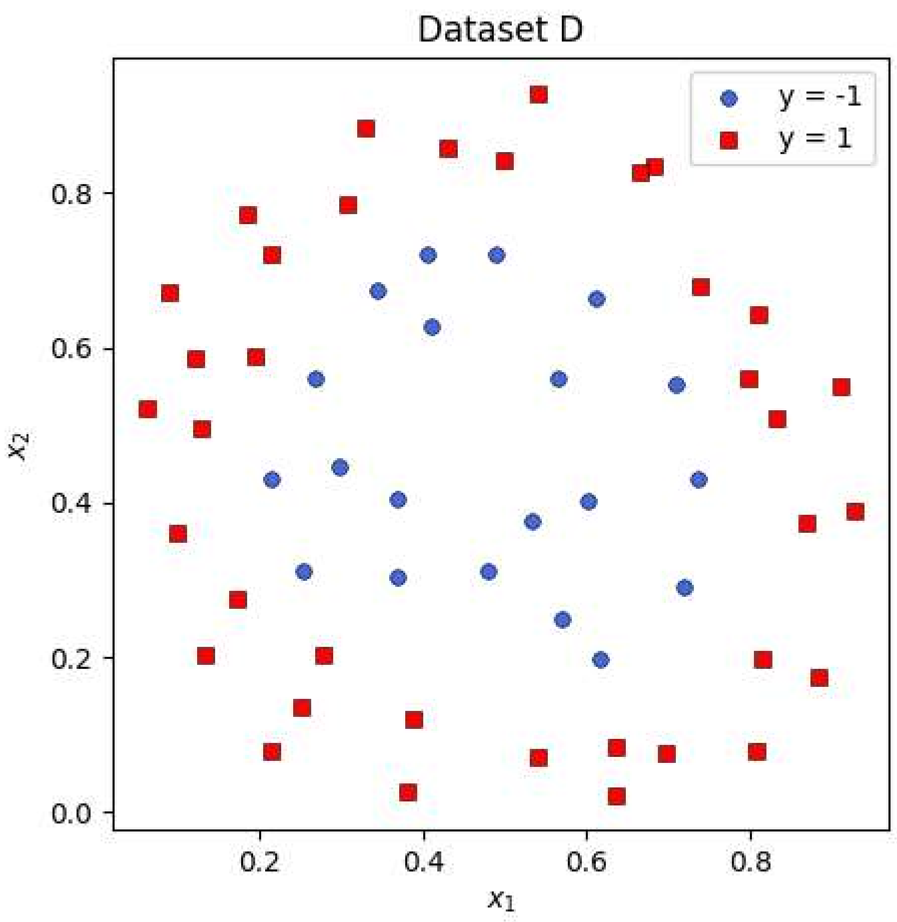
In [7]:
```
xa0 = np.array([0.00806452, 0.0467742, -0.0145161, 0.0564516, 0.130645, 0.0887097,
ya0 = np.array([0.418367, 0.540816, 0.565306, 0.704082, 0.720408, 0.822449, 0.88367
xa1 = np.array([0.324194, 0.304839, 0.372581, 0.369355, 0.453226, 0.546774, 0.72419
ya1 = np.array([0.540816, 0.320408, 0.25102, 0.185714, 0.120408, 0.0959184, 0.10408
Xa = np.concatenate([np.vstack([xa0,ya0]).T,np.vstack([xa1,ya1]).T],0)
ya = np.array([-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
plot(Xa,ya,title="Dataset A")

xb1 = np.array([0.43599,0.54966,0.42037,0.20465,0.29965,0.62113,0.13458,0.18444,0.8
xb2 = np.array([0.025926,0.43532,0.33033,0.61927,0.26683,0.52914,0.51358,0.78534,0.
yb = np.array([1,-1,1,1,1,-1,1,1,-1,-1,-1,1,1,1,1,1,-1,-1,-1,1,-1,-1,-1,1,1,-1,1,1,
Xb = np.vstack([xb1,xb2]).T
plot(Xb,yb,title="Dataset B")

xc1 = np.array([0.05,0.08871,0.18226,0.18871,0.27581,0.25323,0.10806,0.19516,0.2564
xc2 = np.array([0.58571,0.65918,0.63878,0.51633,0.51633,0.37755,0.3449,0.47551,0.69
Xc = np.vstack([xc1,xc2]).T
```

```
yc = np.array([-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
plot(Xc,yc,title="Dataset C")

xd1 = np.array([0.062903,0.08871,0.18548,0.33065,0.54032,0.68226,0.81129,0.91129,0.
xd2 = np.array([0.52041,0.67143,0.77347,0.88367,0.92857,0.83469,0.64286,0.54898,0.3
Xd = np.vstack([xd1,xd2]).T
yd = np.array([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
plot(Xd,yd,title="Dataset D")
```




Dataset B
y = -1
y = 1
x_1
x_2


Dataset C
y = -1
y = 1
x_1
x_2



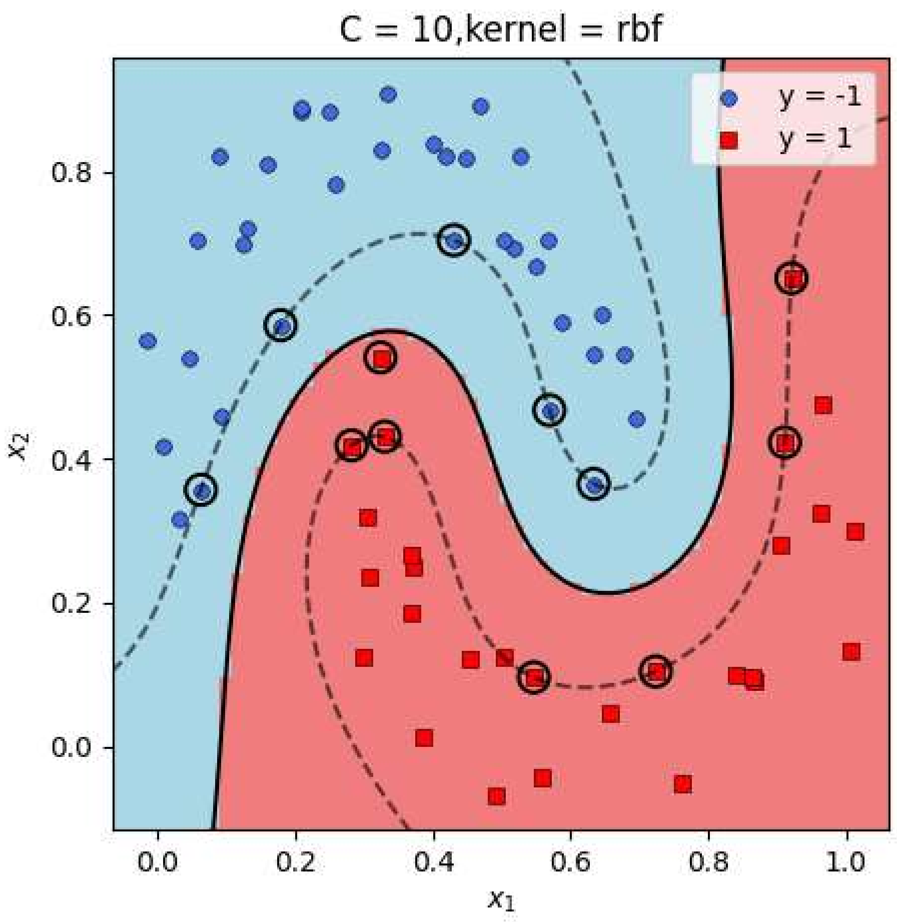
Using the Kernel Trick

Now, train four SVC models, one for each dataset. Try out different combinations of 'kernel' and 'C', until you find a satisfactory classifier in each case.

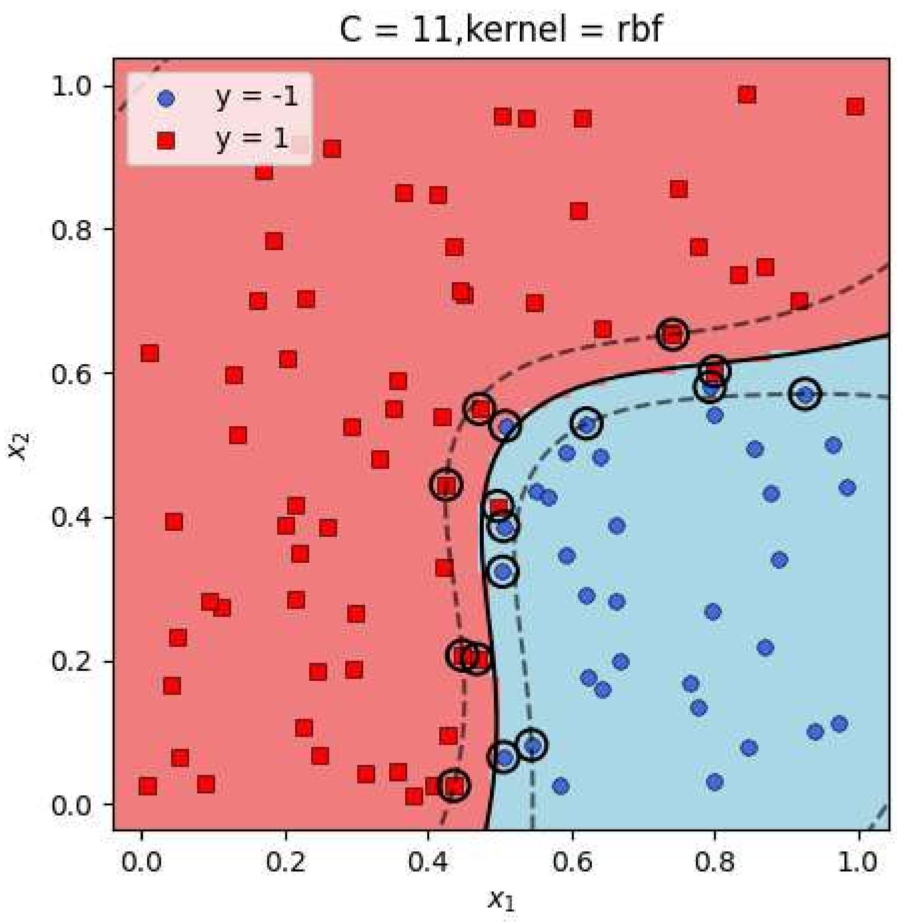
Please generate a plot for each dataset showing the results of a trained support vector classifier, using the provided function:

`plot(Xdata, ydata, svm_model, title)`

In [8]:
```
# YOUR CODE GOES HERE
# (Dataset A)

# for k in ["linear",'poly','rbf','sigmoid']:
#     for C in [1e-5,1e-3,0.1,1,10,1e3,1e5]:
#         svm = SVC(C=C,kernel=k)
#         svm.fit(Xa,ya)
#         plot(Xa,ya,svm,f"C = {C},kernel = {k}")

k = 'rbf'
C = 10
svm = SVC(C=C,kernel=k)
svm.fit(Xa,ya)
plot(Xa,ya,svm,f"C = {C},kernel = {k}")
```

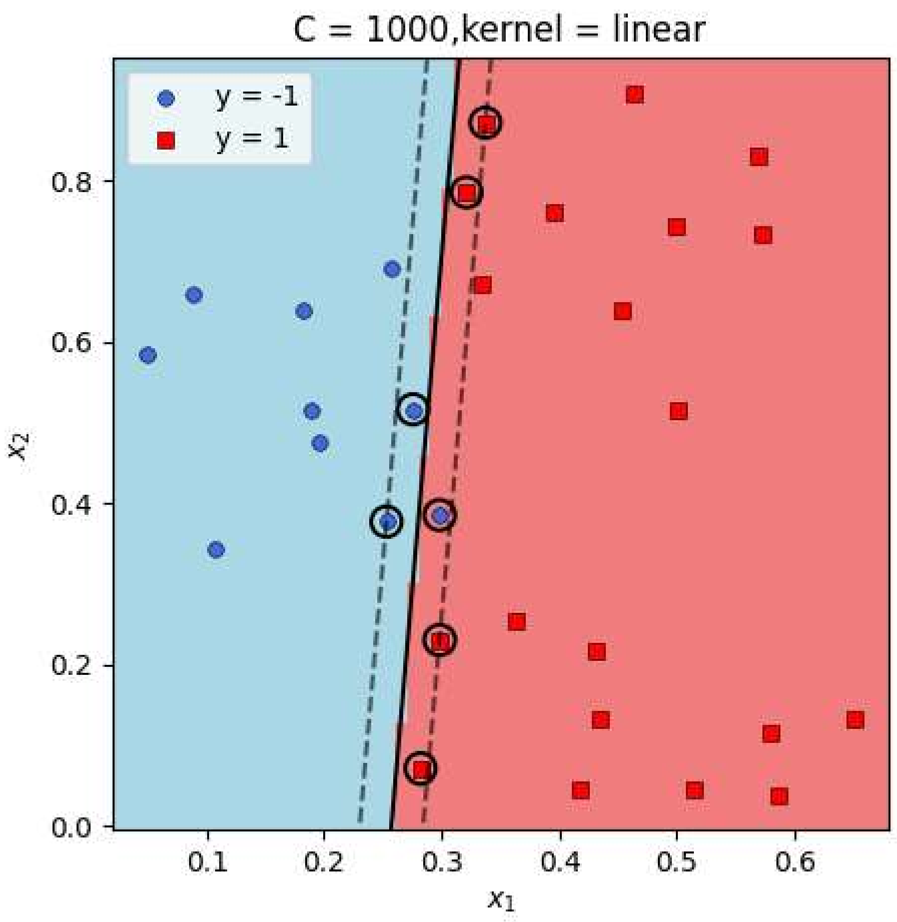


In [13]:
```
# YOUR CODE GOES HERE
# (Dataset B)
# for k in ["linear",'poly','rbf','sigmoid']:
#     for C in [1e-5,1e-3,0.1,1,10,1e3,1e5]:
#         svm = SVC(C=C,kernel=k)
#         svm.fit(Xb,yb)
#         plot(Xb,yb,svm,f"C = {C},kernel = {k}")

C = 10
k = 'rbf'
svm = SVC(C=C,kernel=k)
svm.fit(Xb,yb)
plot(Xb,yb,svm,f"C = {C},kernel = {k}")
```

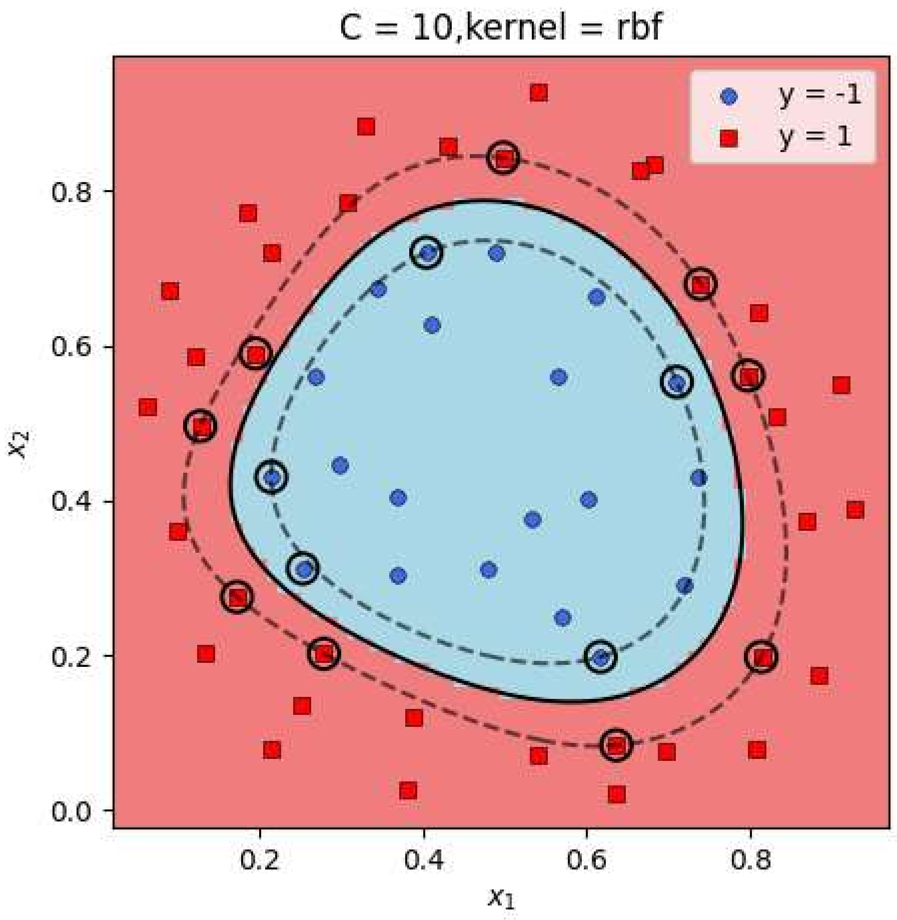


In [15]:

```
# YOUR CODE GOES HERE
# (Dataset C)
# for k in ["linear",'poly','rbf','sigmoid']:
#     for C in [1e-5,1e-3,0.1,1,10,1e3,1e5]:
#         svm = SVC(C=C,kernel=k)
#         svm.fit(Xc,yc)
#         plot(Xc,yc,svm,f"C = {C},kernel = {k}")

C = 1000
k = 'linear'
svm = SVC(C=C,kernel=k)
svm.fit(Xc,yc)
plot(Xc,yc,svm,f"C = {C},kernel = {k}")
```



```
In [16]: # YOUR CODE GOES HERE
         # (Dataset D)
         # for k in ["linear",'poly','rbf','sigmoid']:
         #     for C in [1e-5,1e-3,0.1,1,10,1e3,1e5]:
         #         svm = SVC(C=C,kernel=k)
         #         svm.fit(Xd,yd)
         #         plot(Xd,yd,svm,f"C = {C},kernel = {k}")

         C = 10
         k = 'rbf'
         svm = SVC(C=C,kernel=k)
         svm.fit(Xd,yd)
         plot(Xd,yd,svm,f"C = {C},kernel = {k}")
```


C = 10,kernel = rbf
y = -1
y = 1
0.0
0.2
0.4
0.6
0.8
0.2
0.4
0.6
0.8
x_1
x_2