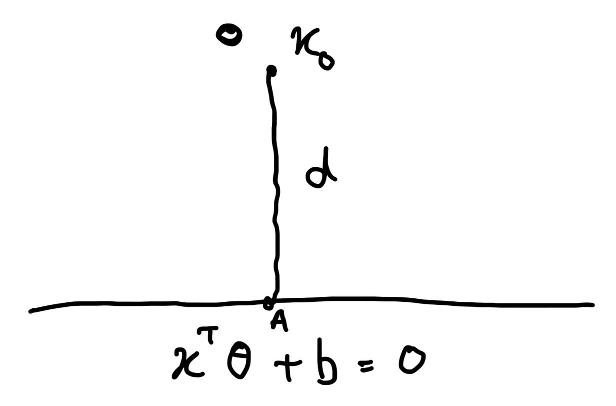
Sheet 10

Monday, 24. January 2022 20:06

Question 1



Let A be any point on the hyperplane $x^T\theta + b = 0$

$\Rightarrow x_A^T\theta + b = 0$

Distance between hyperplane and x_0 is $AO = d$

$$\Rightarrow \quad x_A + d = x_0$$
$$\Rightarrow \quad x_A = x_0 - d$$

$$\therefore \quad (x_0 - d)^T\theta + b = 0$$
$$x_0^T\theta + b = d^T\theta$$

Here, d is parallel to the normal vector of hyperplane:-

$$\Rightarrow \vec{d} = \lambda\vec{\theta}$$

$$\therefore \quad x_0^T\theta + b = \lambda\theta^T\theta$$

$$\therefore \quad \lambda = \frac{x_0^T\theta + b}{\theta^T\theta}$$

$$\therefore \quad \vec{d} = \left[\frac{x_0^T\theta + b}{\theta^T\theta}\right]\vec{\theta}$$

$$\therefore ||d|| = \sqrt{d^Td} = \sqrt{\lambda^2\theta^T\theta} = \lambda\sqrt{\theta^T\theta}$$

$$\therefore \quad d = \frac{x_0^T\theta + b}{\theta^T\theta} \times \sqrt{\theta^T\theta}$$

$$d = \frac{x_0^T\theta + b}{\sqrt{\theta^T\theta}}$$

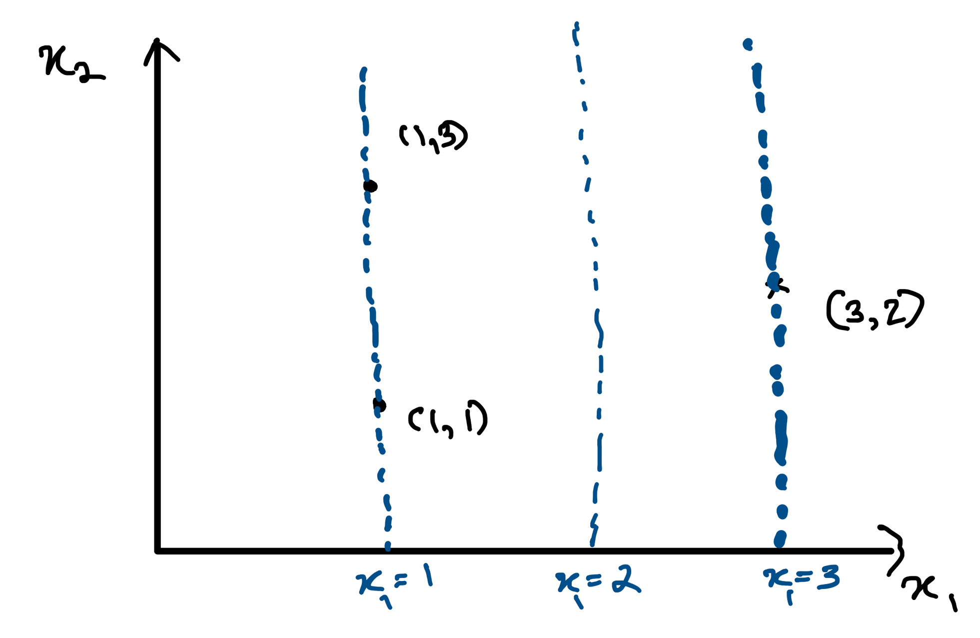
as $\theta^T\theta = ||\theta||^2$

$$\Rightarrow \quad d = \frac{x_0^T\theta + b}{||\theta||}$$

Question 2

$x_1 = [1 \quad 1]^T$, $x_2 = [1 \quad 3]^T$, $x_3 = [3 \quad 2]^T$

$y_1 = 1$, $y_2 = 1$, $y_3 = -1$



The maximum margin decision boundary is shown in the figure.

Its equation is $x_1 = 2$

$$\Rightarrow \quad [-1 \quad 0]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 2 = 0$$

$$\Rightarrow \theta^Tx + b = 0$$

$\Rightarrow \theta = [-1 \quad 0]^T$, $b = 2$.

here, $||\theta|| = 1$

The primal hard margin SVM problem is:-

minimize $\frac{1}{2}||\theta||^2$

s.t $y_n(x_n^T\theta + b) \geq 1$ for $n \in \{1, \ldots, N\}$

$x_1 = [1, 1]^T$, $y_1 = 1 \Rightarrow y_1(x_1^T\theta + b) = [1 \quad 1]\begin{bmatrix} -1 \\ 0 \end{bmatrix} + 2 = 1 \geq 1$

$x_2 = [1, 3]^T$, $y_2 = 1 \Rightarrow y_2(x_2^T\theta + b) = [1 \quad 3]\begin{bmatrix} -1 \\ 0 \end{bmatrix} + 2 = 1 \geq 1$

$x_3 = [3, 2]^T$, $y_3 = -1 \Rightarrow y_3(x_3^T\theta + b) = -\left\{[3 \quad 2]\begin{bmatrix} -1 \\ 0 \end{bmatrix} + 2\right\} = 1 \geq 1$

$\Rightarrow$ linear inequality constraints are satisfied.

Currently, we have, $||\theta|| = 1$

Now, $y(x^T\theta + b)$ for $[1, 1]^T$

$$\theta_1 + \theta_2 + 2 \geq 1 \quad \Rightarrow \quad \theta_1 + \theta_2 \geq -1$$

$[1, 3]^T \quad \Rightarrow \quad \theta_1 + 3\theta_2 \geq -1$

$[3, 2]^T \quad \Rightarrow \quad -3\theta_1 - 2\theta_2 \geq -1 \Rightarrow 3\theta_1 + 2\theta_2 \leq 1$

The three inequalities show that if we select any other θ, its norm would be higher.

c) Now, for the dual problem, we know:-

$$\theta = \sum_{n=1}^{3} \alpha_n x_n y_n$$

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix} = \alpha_1\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \alpha_2\begin{bmatrix} 1 \\ 3 \end{bmatrix} - \alpha_3\begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\Rightarrow \quad \alpha_1 + \alpha_2 - 3\alpha_3 = -1$$
$$\alpha_1 + 3\alpha_2 - 2\alpha_3 = 0$$

also, $\sum_{n=1}^{3} \alpha_n y_n = 0 \Rightarrow \alpha_1 + \alpha_2 - \alpha_3 = 0$

$$\Rightarrow \begin{bmatrix} 1 & 1 & -3 & -1 \\ 1 & 3 & -2 & 0 \\ 1 & 1 & -1 & 0 \end{bmatrix} \xrightarrow[R_3 \to R_3 - R_1]{R_2 \to R_2 - R_1} \begin{bmatrix} 1 & 1 & -3 & -1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

$$\Rightarrow \quad 2\alpha_3 = 1 \quad \Rightarrow \quad \alpha_3 = \frac{1}{2}$$
$$\Rightarrow \quad 2\alpha_2 + \alpha_3 = 1 \quad \Rightarrow \quad \alpha_2 = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$$

$$\Rightarrow \quad \alpha_1 + \alpha_2 - 3\alpha_3 = -1$$
$$\Rightarrow \quad \alpha_1 + \frac{1}{4} - \frac{3}{2} = -1 \Rightarrow \alpha_1 = \frac{1}{4}$$

Hence, $\alpha \in R^3 = \frac{1}{4}\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$, which is the solution to the dual SVM problem $\alpha^* = \underset{\alpha}{\operatorname{argmin}}\ \tilde{L}(\alpha)$ s.t $\sum \alpha_n y_n = 0$ and $\alpha_n \geq 0$ is

$$\alpha^* = \frac{1}{4}\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

Question 3

$$K(x, z) = -\frac{\langle x, z \rangle}{||x||_2 \, ||z||_2}$$

For a valid kernel function, the Gram matrix must be positive semidefinite

$$\Rightarrow \quad K \succeq 0$$

Now, inner products are symmetric and are positive definite.

$\therefore \quad \frac{\langle x, z \rangle}{||x||_2 \, ||z||_2}$ is a valid kernel as the Gram matrix is symmetric and positive semi-definite.

We have $K = -\frac{\langle x, z \rangle}{||x||_2 \, ||z||_2}$ where we multiply a valid kernel with negative constant.

Hence, $K = -\frac{\langle x, z \rangle}{||x|| \, ||z||}$ is NOT a valid kernel function

In [1]:

```
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn import svm
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

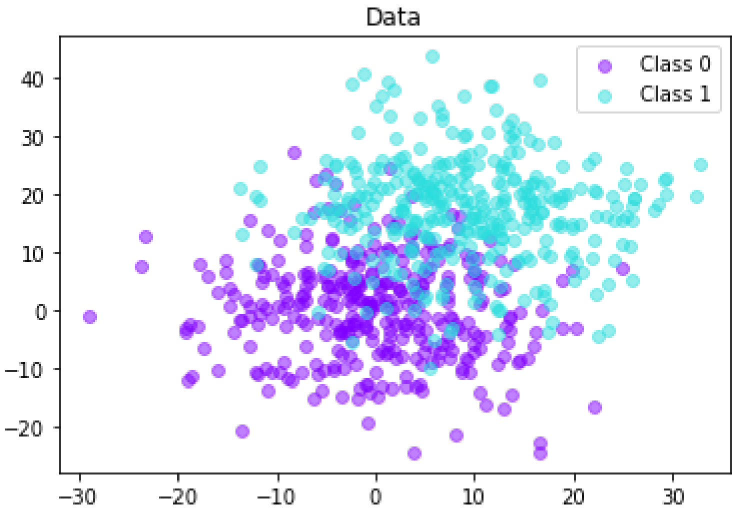
```
# The following lines generate a random set of points in the 2D space. Please refer to make
X,Y = make_blobs(n_samples=1000, n_features=2, centers=np.array([[0,0],[10,18]]), cluster_s
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, shuffle=True)
```

In [3]:

```
def plot_dataset(x,y):
    # This function would plot the generated points
    plt.figure()
    unique_classes = np.unique(y)
    colors = cm.magma(np.linspace(0.0,1.0), unique_classes.size)
    rainbow = cm.get_cmap('rainbow',4)
    for this_class in unique_classes:
        color = rainbow(this_class)
        indices = np.where(y == this_class)
        points = x[indices]
        plt.scatter(
            points[:,0],
            points[:,1],
            color=color,
            label="Class {}".format(this_class),
            alpha=0.5
        )
        plt.title('Data')
    plt.legend()
    plt.show()
```

In [4]:

```
plot_dataset(X_train,Y_train)
```



In [5]:

```
# The following lines learn a SVM over the generated data.
# Please refer to the svm.SVC() class in scikit-learn for further details.
clf = svm.SVC(kernel='linear', degree=7, C=20, max_iter=1000,  verbose=True)
```

In [6]:

```
def fit_data(clf, train_features, train_labels, normalize=False):
    if normalize:
        normalizer = StandardScaler().fit(train_features)
        data = normalizer.transform(train_features)
    else:
        data = train_features
        normalizer=None

    clf.fit(data, train_labels)
    return clf, normalizer
```

In [7]:

```
clf, normalizer = fit_data(clf, X_train, Y_train, normalize=True)
```

```
[LibSVM]
```

In [8]:

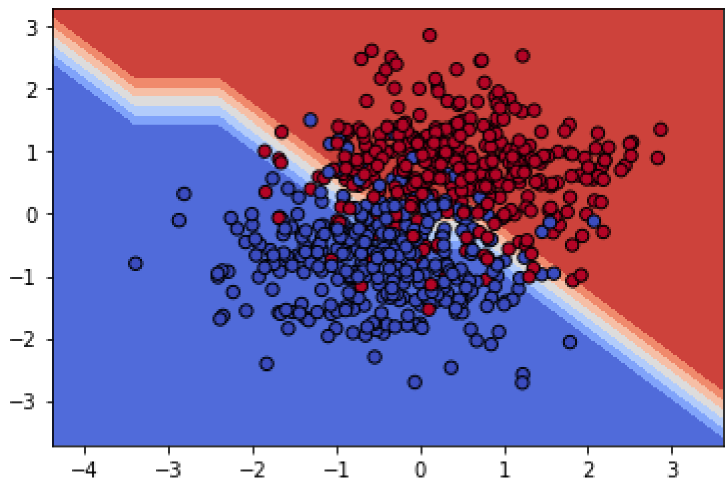
```
# These are helper functions. Please do not modify them for this tutorial

def make_meshgrid(x, y, h=1):
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out
```

In [9]:

```
# This function plots the learnt decision boundary.
# You will need to modify this function to plot the support vectors
def plot_decision_boundary(clf, x,y, normalizer=None):
    if normalizer is not None:
        x = normalizer.transform(x)
    xx,yy = make_meshgrid(x[:,0], x[:,1])
    fig, ax = plt.subplots()
    plot_contours(ax, clf, xx, yy, cmap=cm.coolwarm, alpha=1.0, normalizer=normalizer)
    ax.scatter(x[:,0], x[:,1], c=y, cmap=plt.cm.coolwarm, s=40, edgecolors='k')
    plt.show()
```

In [10]:

```
plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```



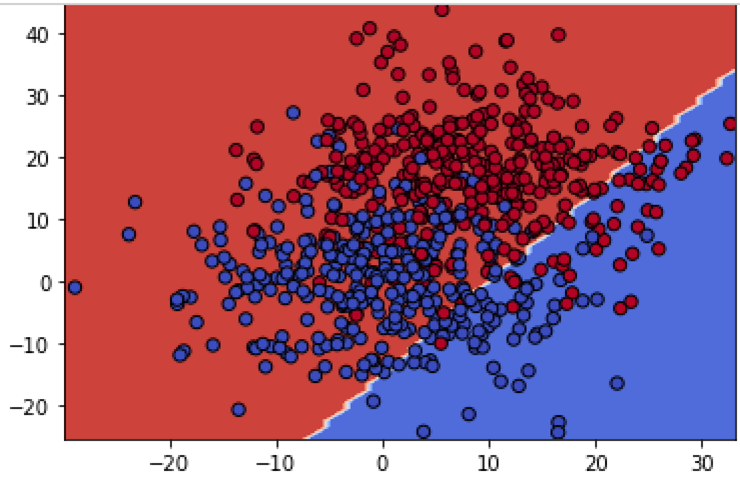
In [24]:

```
# Write a function to predict the test instances using the learnt SVM. Please refer to svm.
def predict_test(clf,x_test, y_test, normalizer=None):
    # If normalizer is None, then the data will be directly predicted and the accuracy comp
    # Otherwise, the x_test should be normalized using the provided normalizer and then pre
    # Please refer to the documentation of StandardScaler in sklearn to see how to do this.
    if normalizer:
        normalizer = StandardScaler().fit(x_test)
        data = normalizer.transform(x_test)
    else:
        data = x_test
        normalizer=None
    pred=clf.predict(x_test)
    accuracy=(pred==y_test).mean()
    return accuracy
```

Fitting with normalize = False

In [29]:

```
C=np.arange(1,51,5)
for c in C:
    clf = svm.SVC(kernel='linear', degree=7, C=c, max_iter=1000,  verbose=True)
    clf, normalizer= fit_data(clf, X_train, Y_train, normalize=False)
    print('\nFor C=',c,' the number of support vectors for each class {0,1} is',clf.n_suppo
    print('\nFor C=',c,' the accuracy is: %0.2f'%predict_test(clf,X_test,Y_test, normalizer
    plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```



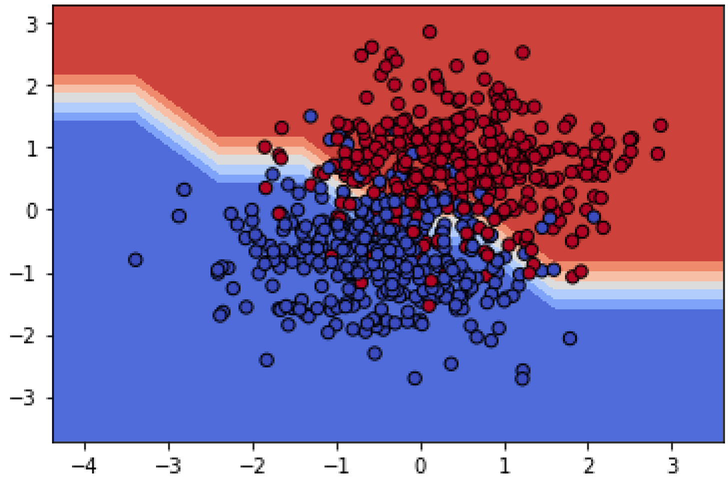

```
[LibSVM]
For C= 11  the number of support vectors for each class {0,1} is [21 31]

For C= 11  the accuracy is: 0.56
```

Fitting with normalize= True

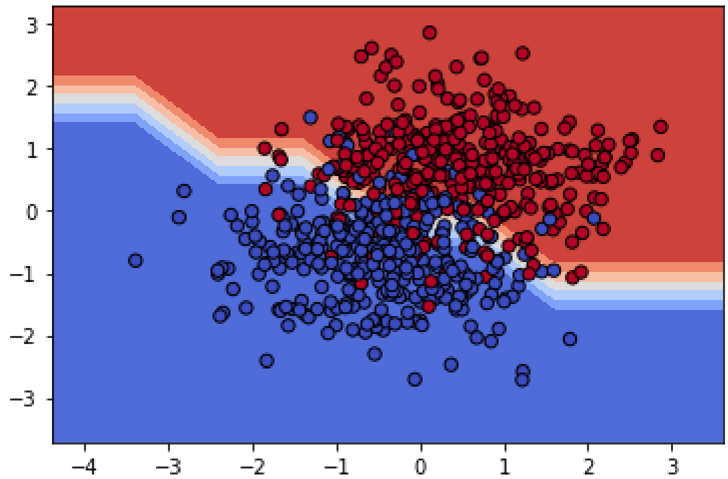
In [28]:

```
C=np.arange(1,51,5)
for c in C:
    clf = svm.SVC(kernel='linear', degree=7, C=c, max_iter=1000,  verbose=True)
    clf, normalizer= fit_data(clf, X_train, Y_train, normalize=True)
    print('\nFor C=',c,' the number of support vectors for each class {0,1} is',clf.n_suppo
    print('\nFor C=',c,' the accuracy is: %0.2f'%predict_test(clf,X_test,Y_test, normalizer
    plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```

```
[LibSVM]
For C= 1  the number of support vectors for each class {0,1} is [115 115]

For C= 1  the accuracy is: 0.76
```

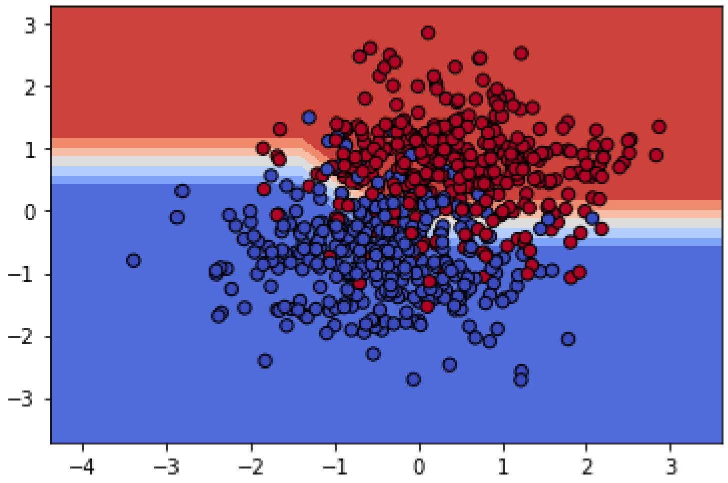



```
[LibSVM]
For C= 6  the number of support vectors for each class {0,1} is [115 115]

For C= 6  the accuracy is: 0.77
```

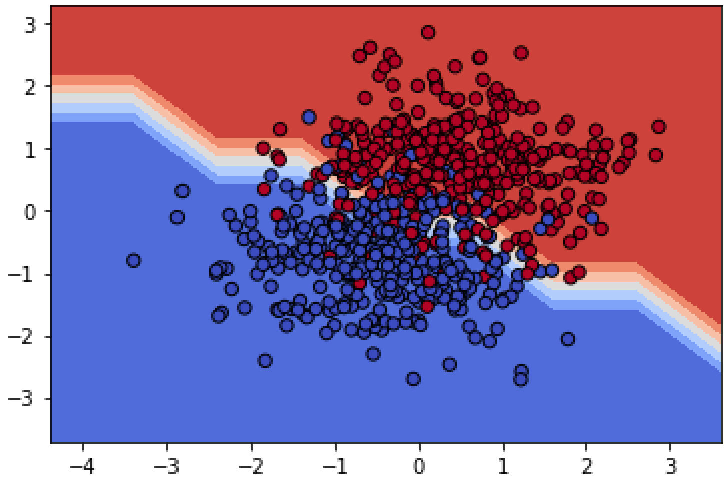



```
[LibSVM]
```

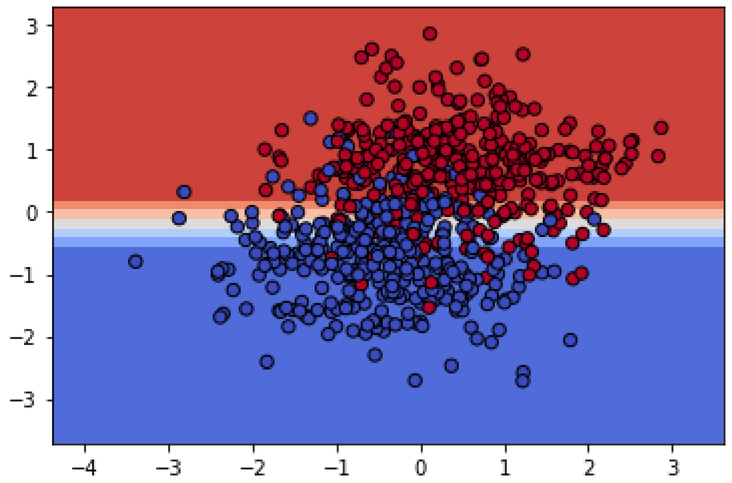
```
For C= 11  the number of support vectors for each class {0,1} is [113 114]

For C= 11  the accuracy is: 0.77
```



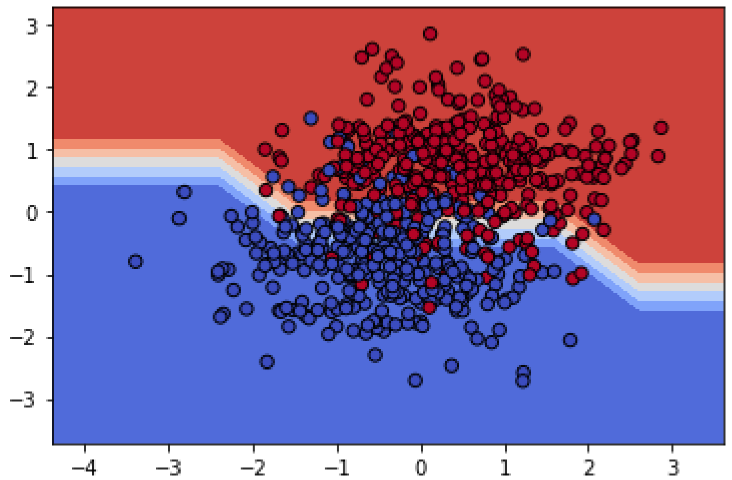
```
[LibSVM]
For C= 16  the number of support vectors for each class {0,1} is [114 113]

For C= 16  the accuracy is: 0.76
```



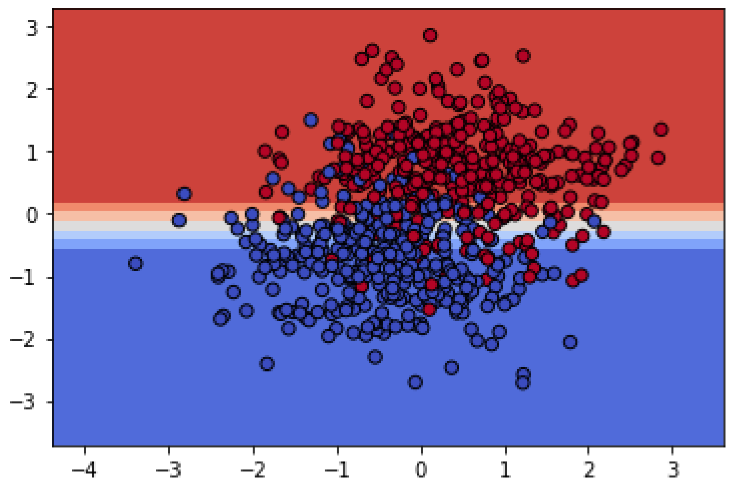
```
[LibSVM]
For C= 21  the number of support vectors for each class {0,1} is [112 112]

For C= 21  the accuracy is: 0.77
```



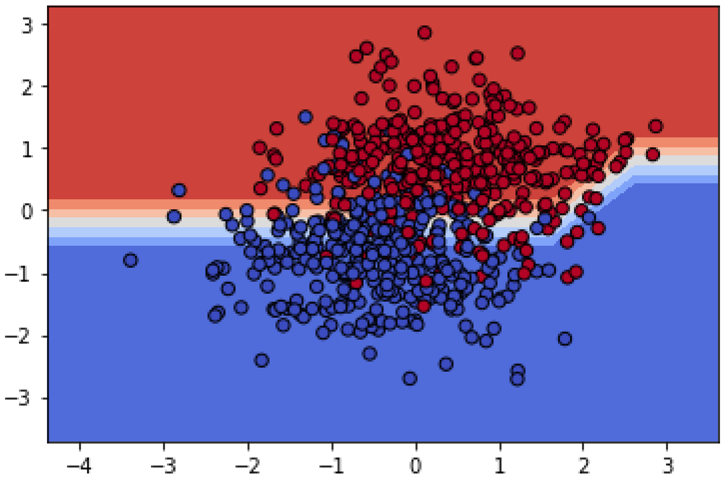

```
[LibSVM]
For C= 26  the number of support vectors for each class {0,1} is [111 110]

For C= 26  the accuracy is: 0.76
```



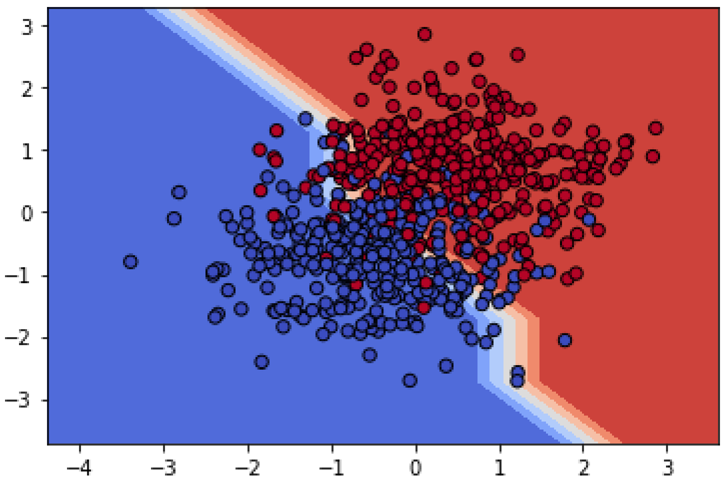

```
[LibSVM]
For C= 31  the number of support vectors for each class {0,1} is [108 109]

For C= 31  the accuracy is: 0.76
```



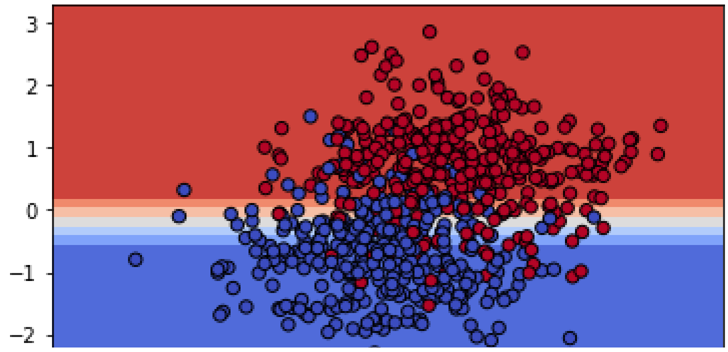
```
[LibSVM]
For C= 36  the number of support vectors for each class {0,1} is [107 107]

For C= 36  the accuracy is: 0.75
```




```
[LibSVM]
For C= 41  the number of support vectors for each class {0,1} is [107 106]

For C= 41  the accuracy is: 0.75
```




```
[LibSVM]
For C= 46  the number of support vectors for each class {0,1} is [105 105]

For C= 46  the accuracy is: 0.77
```



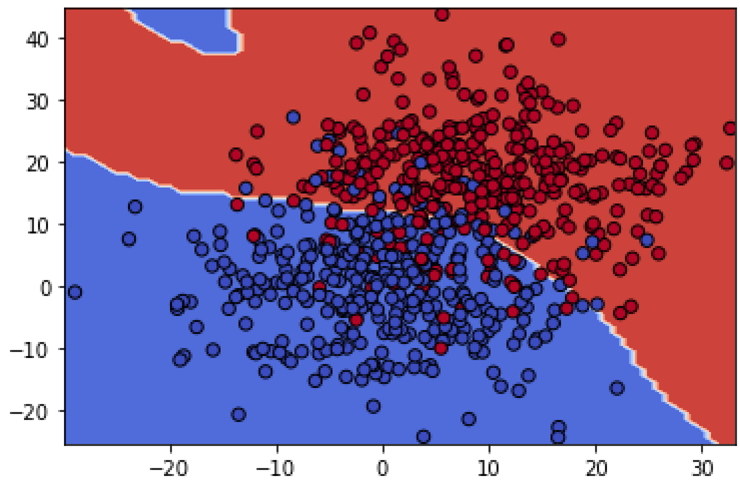
As we normalize, better predictions are seen on the test data. This is because it ensures the scale factors of the data is removed from the picture. This ensures all data is in the same range.

We vary C from [1,51] and observe that for higher values of C we allow more margin violations and do a tradeoff between bias and variance.

In [30]:

```
clf = svm.SVC(kernel='rbf', degree=7, C=c, max_iter=1000,  verbose=True)
clf, normalizer= fit_data(clf, X_train, Y_train, normalize=False)
print('\nFor C=',c,' the number of support vectors for each class {0,1} is',clf.n_support_)
print('\nFor C=',c,' with RBF kernel the accuracy is: %0.2f'%predict_test(clf,X_test,Y_test
plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```

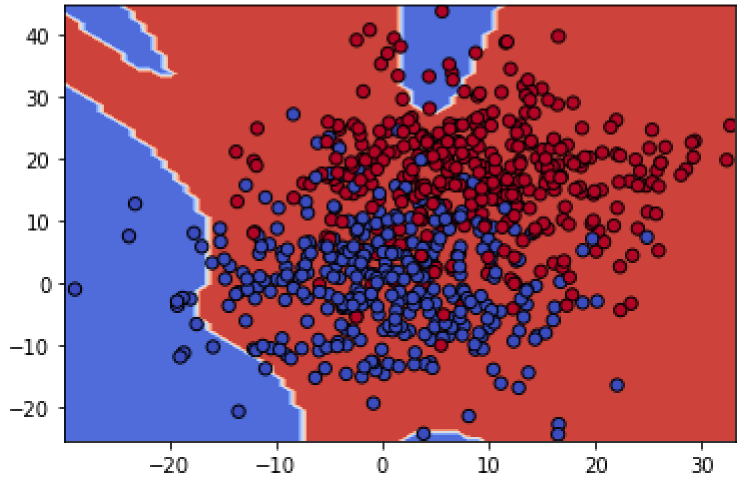
```
[LibSVM]
For C= 46  the number of support vectors for each class {0,1} is [148 119]

For C= 46  with RBF kernel the accuracy is: 0.90
```



In [32]:

```
clf = svm.SVC(kernel='poly', degree=7, C=c, max_iter=1000,  verbose=True)
clf, normalizer= fit_data(clf, X_train, Y_train, normalize=False)
print('\nFor C=',c,' the number of support vectors for each class {0,1} is',clf.n_support_)
print('\nFor C=',c,' with polynomial kernel the accuracy is: %0.2f'%predict_test(clf,X_test
plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```

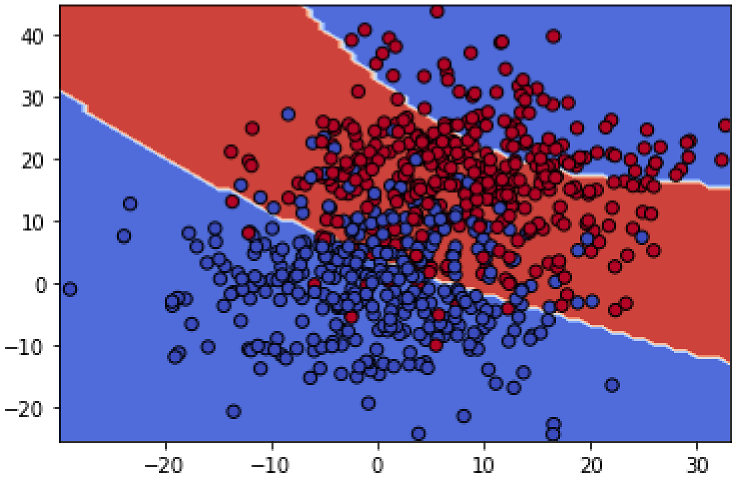
```
[LibSVM]
For C= 46  the number of support vectors for each class {0,1} is [12 61]

For C= 46  with polynomial kernel the accuracy is: 0.50
```



In [33]:

```
clf = svm.SVC(kernel='sigmoid', degree=7, C=c, max_iter=1000,  verbose=True)
clf, normalizer= fit_data(clf, X_train, Y_train, normalize=False)
print('\nFor C=',c,' the number of support vectors for each class {0,1} is',clf.n_support_)
print('\nFor C=',c,' with sigmoid kernel the accuracy is: %0.2f'%predict_test(clf,X_test,Y_
plot_decision_boundary(clf,X_train,Y_train, normalizer=normalizer)
```

```
[LibSVM]
For C= 46  the number of support vectors for each class {0,1} is [111 111]

For C= 46  with sigmoid kernel the accuracy is: 0.71
```



We see that the RBF kernel gives the highest accuract of 0.9.

Index of comments

10.1 higher C -> less regularization -> more complex model -> less points to be misclassified -> less number of support vectors