In [1]:
```
import numpy as np
import numpy as np
import pandas as pd

import cvxcannot import name 'plot_contour' from 'utils' (/home/manishakarim

opt

from scipy.io import loadmat
import matplotlib.pyplot as plt
```

1.

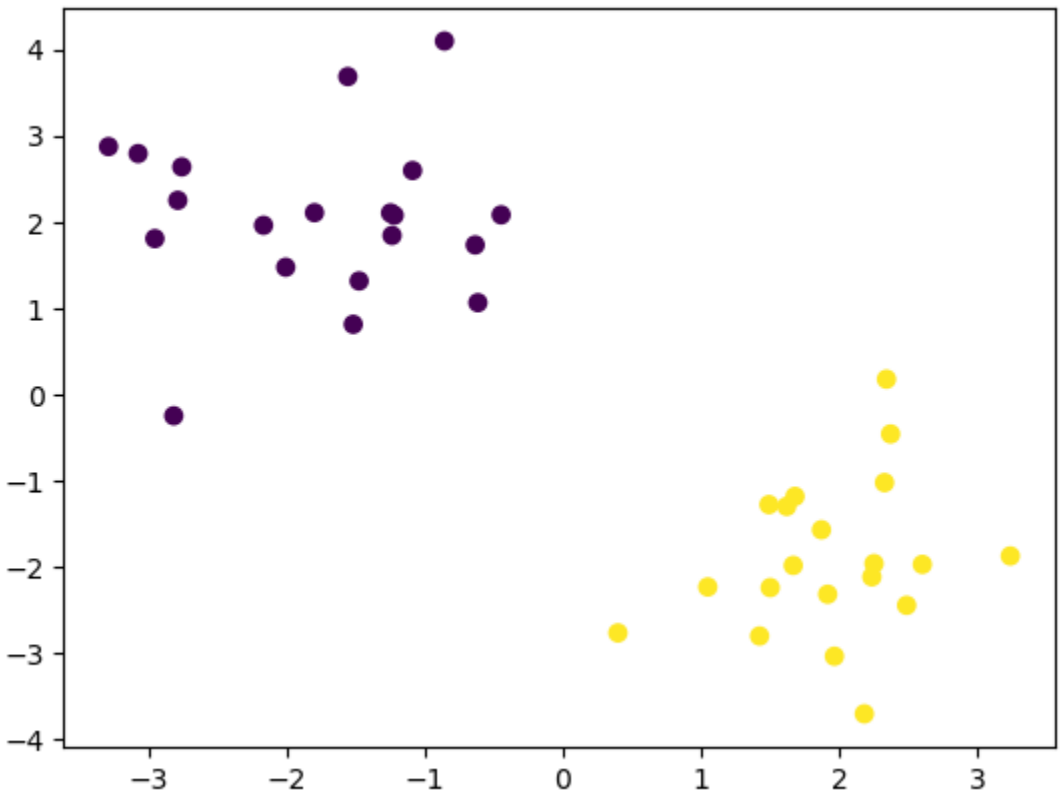
In [2]:
```
data = loadmat(r"SVM_data.mat")
data.keys()
```

Out[2]:
```
dict_keys(['__header__', '__version__', '__globals__', 'x', 'y'])
```

In [3]:
```
x = data['x']
y = data['y']
```

In [4]:
```
plt.scatter(x[:,0], x[:,1], marker='o',c=y)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x77abf9247970>
```



In [5]:
```
class SVM:
    def __init__(self, lr, iterations, lambda_par):
```

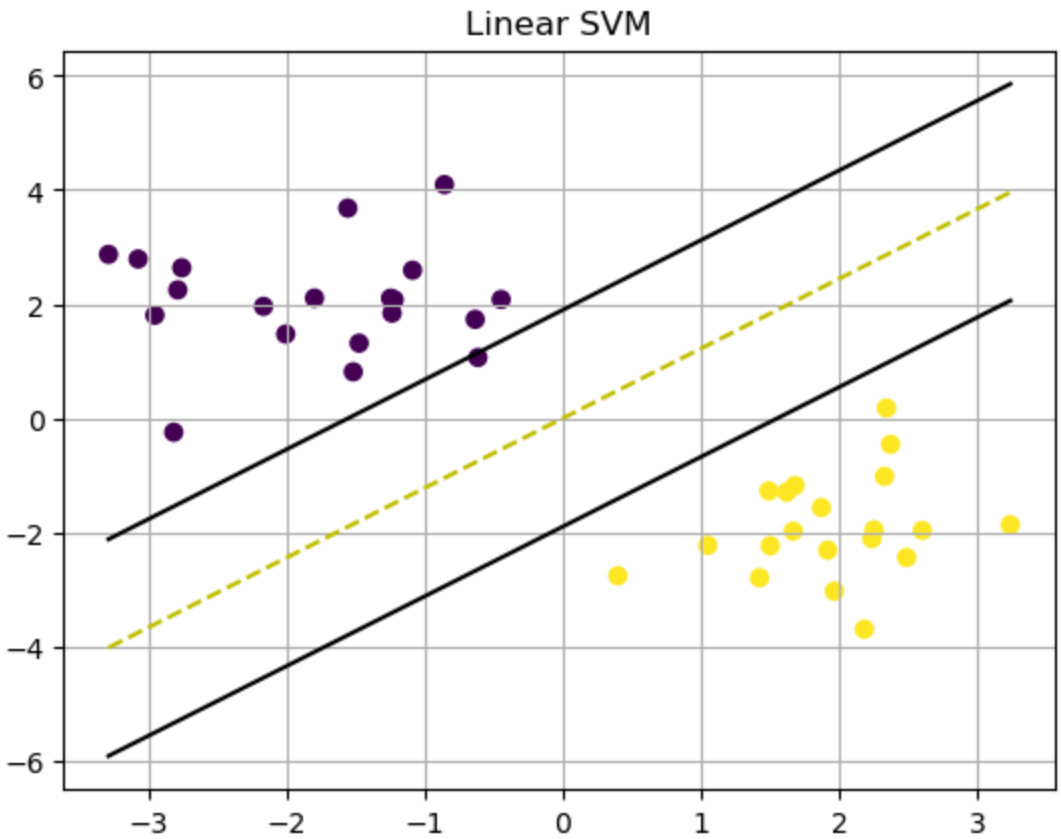
```
        self.lr = lr
        self.iterations = iterations
        self.lambda_par = lambda_par

    def fit(self, x, y):


        self.x = x
        self.y = y
        self.m, self.n = x.shape

        #initialize weights and biases
        self.w = np.zeros(self.n)
        self.b = 0

        #update weights and biases
        for iteration in range(self.iterations):
            for i, Xi in enumerate(x):


                if ((y[i] * (np.dot(Xi, self.w) - self.b)) >= 1).all() :

                        Xi= Xi.reshape(2,1)

                        dw = self.lr * (2 * self.lambda_par * self.w)
                        db = 0
                else:

                        Xi= Xi.reshape(2,1)

                        dw = self.lr * (2 * self.lambda_par * self.w - np.do
                        db = self.lr * y[i]

                self.w = self.w - self.lr * dw
                self.b = self.b - self.lr * db

        return self.w, self.b
```

In [7]:
```
classifier = SVM(lr=0.5, iterations=100, lambda_par = 0.1)
w, b = classifier.fit(x,y)
```

In [33]:
```
def get_hyperplane(x, w, b, offset):
    return (-w[0] * x + b + offset) / w[1]

def plot_svm(X, y, w, b, title):

    fig = plt.figure()
    plt.scatter(X[:,0], X[:,1], marker='o',c=y)

    x0_1 = np.amin(X[:,0])
    x0_2 = np.amax(X[:,0])

    x1_1 = get_hyperplane(x0_1, w, b, 0)
    x1_2 = get_hyperplane(x0_2, w, b, 0)
```

```
    x1_1_m = get_hyperplane(x0_1, w, b, -1)
    x1_2_m = get_hyperplane(x0_2, w, b, -1)


    x1_1_p = get_hyperplane(x0_1, w, b, 1)
    x1_2_p = get_hyperplane(x0_2, w, b, 1)

    plt.plot([x0_1, x0_2],[x1_1, x1_2], 'y--')
    plt.plot([x0_1, x0_2],[x1_1_m, x1_2_m], 'k')
    plt.plot([x0_1, x0_2],[x1_1_p, x1_2_p], 'k')


    plt.title('Linear SVM')
    plt.grid()
```

In [9]:
```
plot_svm(x, y, w, b)
```



2.

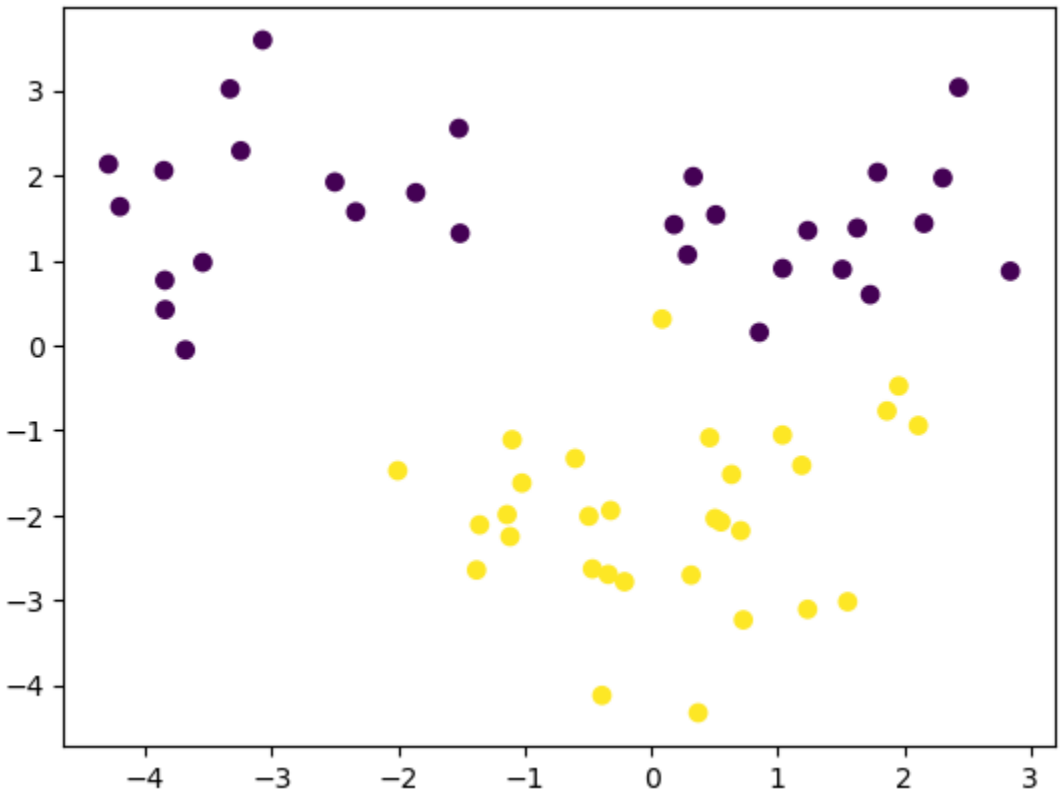
In [10]:
```
data = loadmat(r"SVM_data_nonlinear.mat")
data.keys()
```

Out[10]:
```
dict_keys(['__header__', '__version__', '__globals__', 'x', 'y'])
```

In [11]:
```
X = data['x']
y = data['y']
```

In [14]:
```
plt.scatter(X[:,0], X[:,1], marker='o',c=y)
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x77abf70f8370>
```



In [22]:
```
def gaussian(x, z, sigma=0.1):
    return np.exp(-np.linalg.norm(x - z, axis=1) ** 2 / (2 * (sigma ** 2)))
```

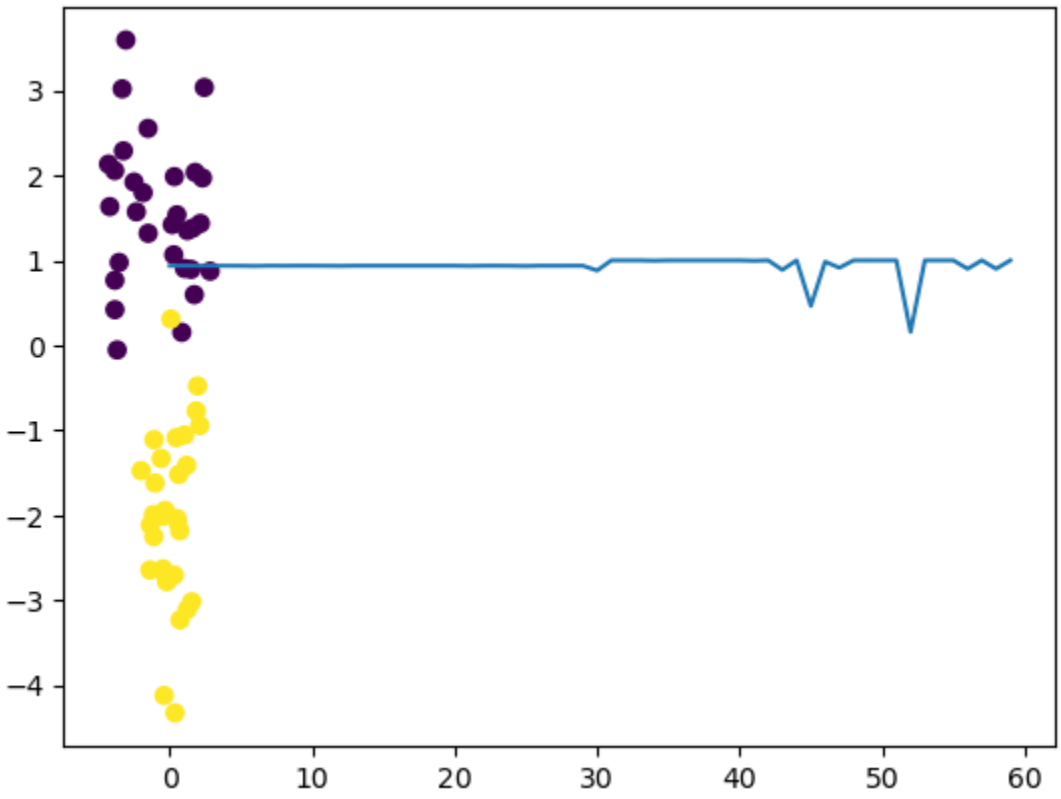
In [31]:
```
class SVM:
    def __init__(self, kernel=gaussian, C=1):
        self.kernel = kernel
        self.C = C

    def fit(self, X, y):
        self.y = y
        self.X = X
        m, n = X.shape

        # Calculate Kernel
        self.K = np.zeros((m, m))
        for i in range(m):
            self.K[i, :] = self.kernel(X[i, np.newaxis], self.X)


        # Used cvxopt instead of quadprog[matlab]

        P = cvxopt.matrix(np.outer(y, y) * self.K)
        q = cvxopt.matrix(-np.ones((m, 1)))
        G = cvxopt.matrix(np.vstack((np.eye(m) * -1, np.eye(m))))
        h = cvxopt.matrix(np.hstack((np.zeros(m), np.ones(m) * self.C)))
        y = y.astype(np.double) # To make buffer s
        A = cvxopt.matrix(y, (1, m), "d")
```

```
        b = cvxopt.matrix(np.zeros(1))
        cvxopt.solvers.options["show_progress"] = False
        sol = cvxopt.solvers.qp(P, q, G, h, A, b)
        self.alphas = np.array(sol["x"])
        return self.alphas
```

In [32]:
```
svm = SVM()
alpha = svm.fit(X, y)
```

In [30]:
```
plt.scatter(X[:,0], X[:,1], marker='o',c=y)
plt.plot(alpha)
```

Out[30]: [<matplotlib.lines.Line2D at 0x77abc80cc190>]



In [ ]: