SVM - Lab

September 5, 2021

Your work: - Load this dataset to numpy, with first two columns as features and last as target - Plot the data using a scatter plot - Perform the SVM classification using our scratch code

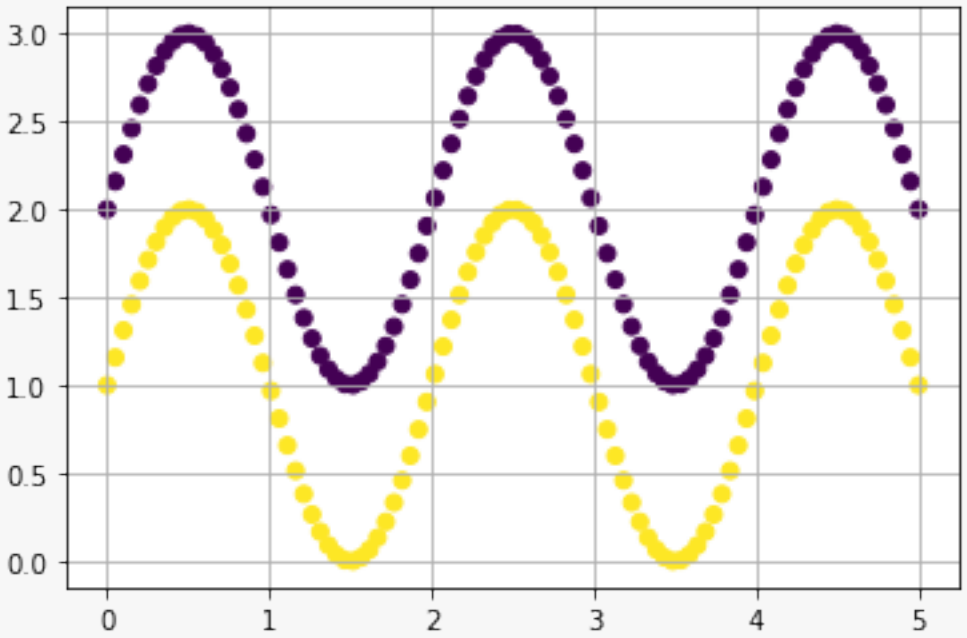
[63]:

[64]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[65]:
```
data = np.array(dataset)

print(data.shape)
plt.grid()
plt.scatter(data[:, 0], data[:, 1], c=data[:, 2])
```

```
(200, 3)
```

[65]:
```
<matplotlib.collections.PathCollection at 0x7f590044ffd0>
```



[66]:
```
import cvxopt
from sklearn import datasets

#here I use z instead of xprime since I don't know how to write prime in code...
 ↪.
def linear(x, z):
```

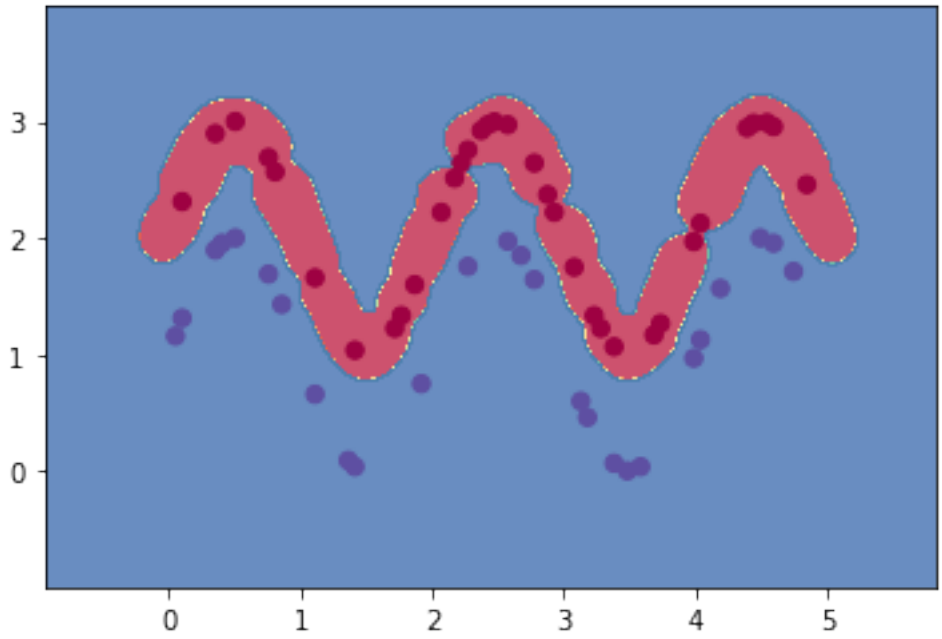
```
    return np.dot(x, z.T)

def polynomial(x, z, p=5):
    return (1 + np.dot(x, z.T)) ** p

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

def plot_contour(X, y, svm):
    # plot the resulting classifier
    h = 0.01
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

    points = np.c_[xx.ravel(), yy.ravel()]

    Z = svm.predict(points)
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral, alpha=0.8)

    # plt the points
    plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.Spectral)


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)

        # Solve with cvxopt final QP needs to be reformulated
        # to match the input form for cvxopt.solvers.qp
        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)))
```

```
        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"])

    def predict(self, X):  #<----this is X_test
        y_predict = np.zeros((X.shape[0]))
        sv = self.get_parameters(self.alphas)

        for i in range(X.shape[0]):
            y_predict[i] = np.sum(
                self.alphas[sv]
                * self.y[sv, np.newaxis]
                * self.kernel(X[i], self.X[sv])[:, np.newaxis]
            )

        return np.sign(y_predict + self.b)

    def get_parameters(self, alphas):
        threshold = 1e-5

        sv = ((alphas > threshold) * (alphas < self.C)).flatten()
        self.w = np.dot(self.X[sv].T, alphas[sv] * self.y[sv, np.newaxis])
        self.b = np.mean(
            self.y[sv, np.newaxis]
            - self.alphas[sv] * self.y[sv, np.newaxis] * self.K[sv, sv][:, np.
↪newaxis]
        )
        return sv
```

[68]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data[:,[0,1]], data[:,2],␣
↪test_size=0.3)

y_train[y_train==0] = -1
y_test[y_test==0] = -1
```

[69]:
```
svm = SVM(kernel=gaussian)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
```

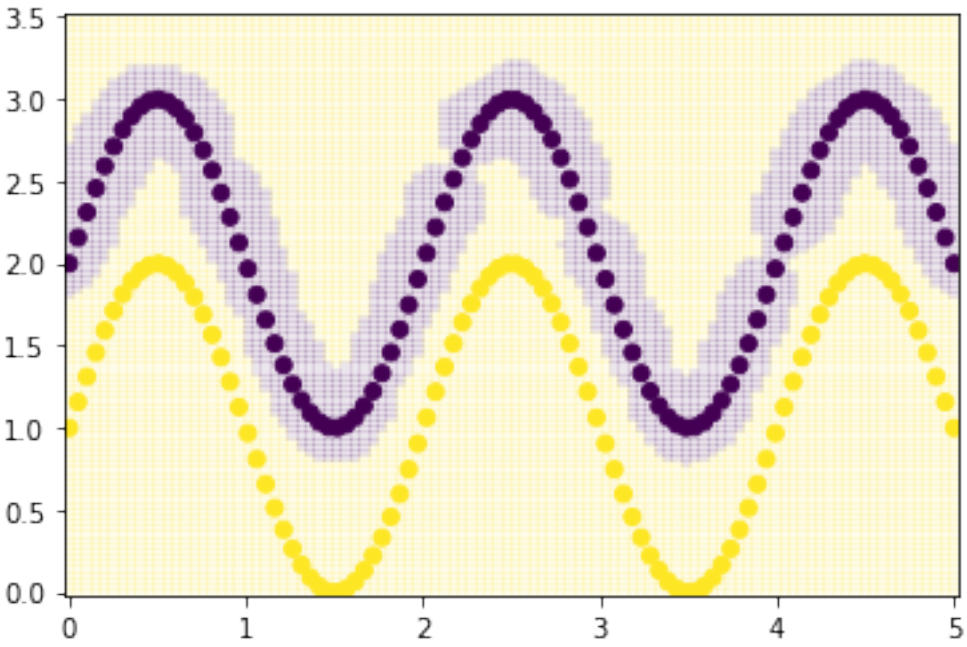
[73]:
```
plot_contour(X_test, y_pred, svm)
```



[71]:
```
xlim = np.linspace(0, 5, 100)
ylim = np.linspace(0, 3.5, 100)

x_mesh,y_mesh = np.meshgrid(xlim,ylim)

x_mesh = x_mesh.reshape(-1,1)
y_mesh = y_mesh.reshape(-1,1)

mesh = np.append(x_mesh,y_mesh,axis=1)
y_p = svm.predict(mesh)
x_mesh = x_mesh.reshape(100,100)
y_mesh = y_mesh.reshape(100,100)
y_p = y_p.reshape(100,100)
print(x_mesh.shape, y_mesh.shape, mesh.shape, y_p.shape)
plt.pcolormesh(x_mesh,y_mesh,y_p,cmap='viridis',shading='auto',alpha=0.1)
plt.scatter(data[:, 0], data[:, 1], c=data[:, 2])
```

```
(100, 100) (100, 100) (10000, 2) (100, 100)
```

[71]:
```
<matplotlib.collections.PathCollection at 0x7f59003687c0>
```


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5

[ ]: