Lab Problem

PS 1

```
import numpy as np
import matplotlib.pyplot as plt

X_positive = np.array([[3, 1], [3, -1], [6, 1], [6, -1]])
X_negative = np.array([[1, 0], [0, 1], [0, -1], [-1, 0]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])


X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[ 3  1]
 [ 3 -1]
 [ 6  1]
 [ 6 -1]
 [ 1  0]
 [ 0  1]
 [ 0 -1]
 [-1  0]]
[ 1.  1.  1.  1. -1. -1. -1. -1.]
```

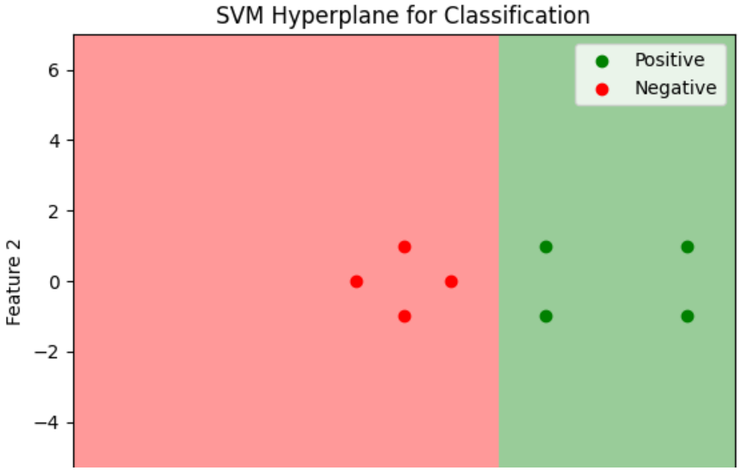
```
from sklearn import svm

clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

```
SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
1.0004816608996538 * x + -1.6653345369377348e-16 * y + -2.001123875432526 = 0
```

```
x_min, x_max = -7, 7
y_min, y_max = -7, 7
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)

plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



PS 2

```
X_positive = np.array([[2, 2], [2, -2], [-2, -2], [-2, 2]])
X_negative = np.array([[1, 1], [1, -1], [-1, -1], [-1, 1]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])


X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[ 2  2]
 [ 2 -2]
 [-2 -2]
 [-2  2]
 [ 1  1]
 [ 1 -1]
 [-1 -1]
 [-1  1]]
[ 1.  1.  1.  1. -1. -1. -1. -1.]
```

```
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

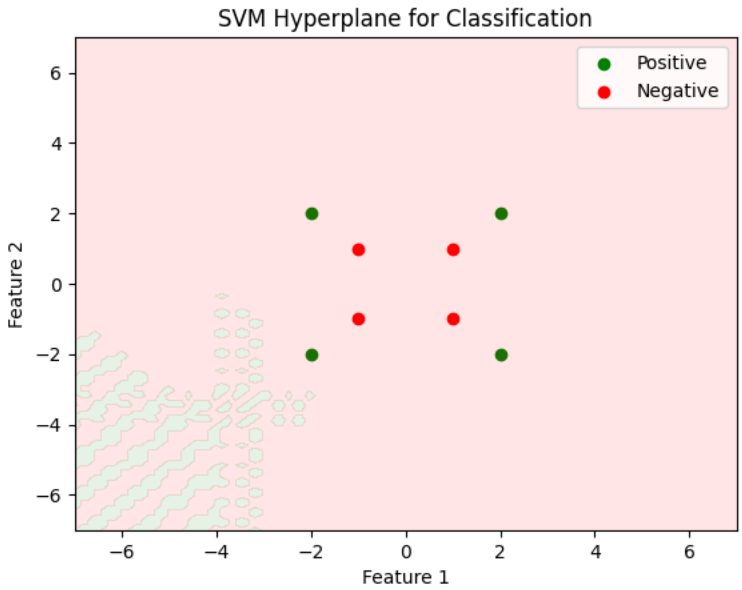
```
▾ SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
0.0 * x + 0.0 * y + -0.0 = 0
```

```
x_min, x_max = -7, 7
y_min, y_max = -7, 7
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)

plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
```

```
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



PS 3

```
X_positive = np.array([[1, 1], [2, 2], [2, 0]])
X_negative = np.array([[0, 0], [1, 0], [0, 1]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])


X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[1 1]
 [2 2]
 [2 0]
 [0 0]
 [1 0]
 [0 1]]
[ 1.  1.  1. -1. -1. -1.]
```

```
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

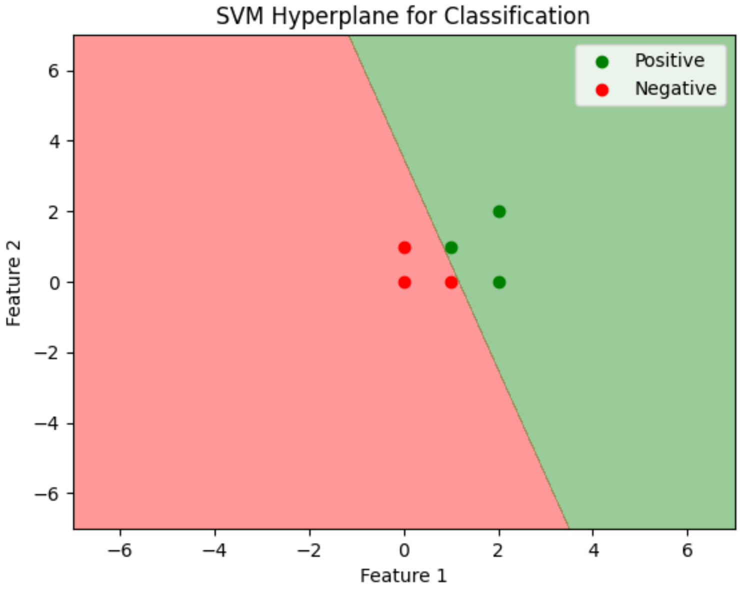
```
SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
1.2000000000000002 * x + 0.4 * y + -1.4 = 0
```

```
x_min, x_max = -7, 7
y_min, y_max = -7, 7
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)
```

```
plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



Assignments

1

```
X_negative = np.array([[2, 2], [3, 3], [4, 4], [5, 5],[4, 6],[3, 7],[4, 8],[5, 9],[6, 10]])
X_positive = np.array([[6, 2], [7, 3], [8, 4], [9, 5],[8,8],[7,7],[7,8],[7,9],[8,10]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])


X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[ 6  2]
 [ 7  3]
 [ 8  4]
 [ 9  5]
 [ 8  8]
 [ 7  7]
 [ 7  8]
 [ 7  9]
 [ 8 10]
 [ 2  2]
 [ 3  3]
 [ 4  4]
 [ 5  5]
 [ 4  6]
 [ 3  7]
 [ 4  8]
 [ 5  9]
 [ 6 10]]
[ 1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
```

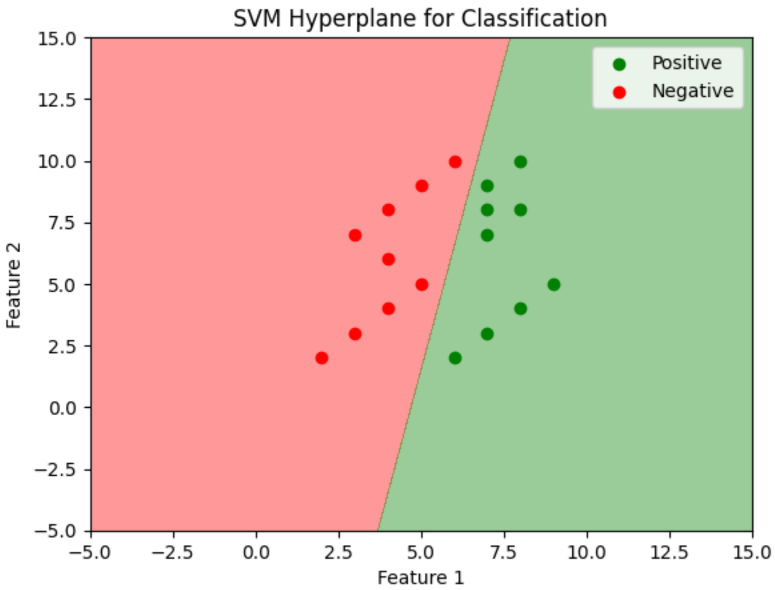
```
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

```
SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
1.428847397087818 * x + -0.2858982647252102 * y + -6.714531019633633 = 0
```

```
x_min, x_max = -5, 15
y_min, y_max = -5, 15
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)

plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



2

```
X_positive = np.array([[3, 1], [3, -1], [6, 1], [6,-1]])
X_negative = np.array([[1, 0], [0, 1], [0, -1], [-1, 0]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])


X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[ 3  1]
 [ 3 -1]
 [ 6  1]
 [ 6 -1]
 [ 1  0]
```

```
 [ 0  1]
 [ 0 -1]
 [-1  0]]
[ 1.  1.  1.  1. -1. -1. -1. -1.]
```

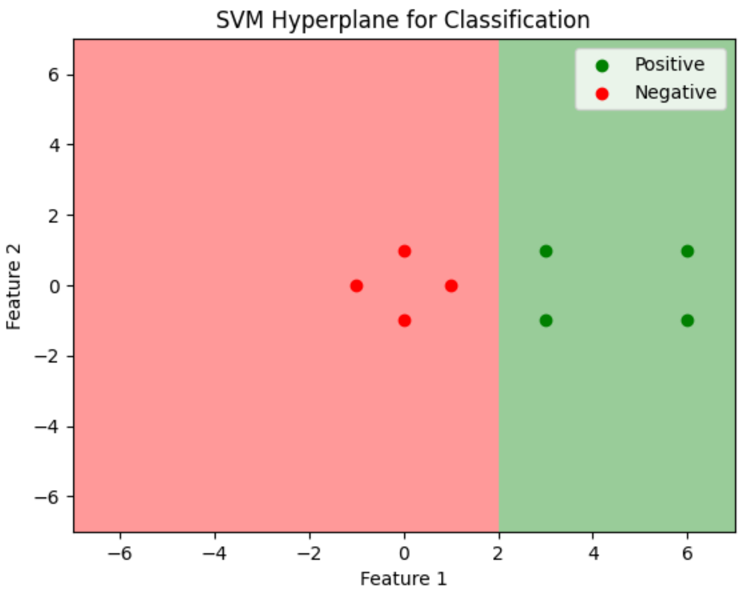
```
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

```
▾        SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
1.0004816608996538 * x + -1.6653345369377348e-16 * y + -2.001123875432526 = 0
```

```
x_min, x_max = -7, 7
y_min, y_max = -7, 7
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)

plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



3

```
X_positive = np.array([[2, 2], [2, -2], [-2, 2], [-2, 2]])
X_negative = np.array([[1, 1], [1, -1],[-1, -1],[-1, 1]])


y_positive = np.ones(X_positive.shape[0])
y_negative = -np.ones(X_negative.shape[0])
```

```
X = np.vstack((X_positive, X_negative))
y = np.hstack((y_positive, y_negative))

print(X)
print(y)
```

```
[[ 2  2]
 [ 2 -2]
 [-2  2]
 [-2  2]
 [ 1  1]
 [ 1 -1]
 [-1 -1]
 [-1  1]]
[ 1.  1.  1.  1. -1. -1. -1. -1.]
```

```
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

```
▾ SVC
SVC(kernel='linear')
```

```
w = clf.coef_[0]
b = clf.intercept_[0]

print("Equation of the hyperplane:")
print(f"{w[0]} * x + {w[1]} * y + {b} = 0")
```

```
Equation of the hyperplane:
-4.440892098500626e-16 * x + 0.6666666666666667 * y + -0.33333333333333337 = 0
```

```
x_min, x_max = -7, 7
y_min, y_max = -7, 7
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Calculate the distances from each point to the hyperplane
distances = clf.decision_function(X)

plt.figure()
plt.scatter(X_positive[:, 0], X_positive[:, 1], color='green', label='Positive')
plt.scatter(X_negative[:, 0], X_negative[:, 1], color='red', label='Negative')
plt.contourf(xx, yy, Z, levels=[-100, 0, 100], colors=['red', 'green'], alpha=0.4)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.title('SVM Hyperplane for Classification')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```