svm example 2

July 12, 2022

[ ]:

```
# Written by Miyeon Lee assignment #7 Intro MLDL Fall 2019 Prof. Lim #
import numpy as np
import seaborn as sns
from sklearn.svm import SVC
import matplotlib.pyplot as plt

classifier = SVC(kernel = 'linear', C = 10)
training_points = np.array([[-1, 4], [-2, 3], [-3, 4], [5, 6], [4, 5], [5, 5]])
labels = [-1, -1, -1, 1, 1, 1]
classifier.fit(training_points,labels)
```

[ ]:

```
SVC(C=10, kernel='linear')
```

[ ]:

```
x, y = training_points.T

print("x:", x)
print("y:", y)
```

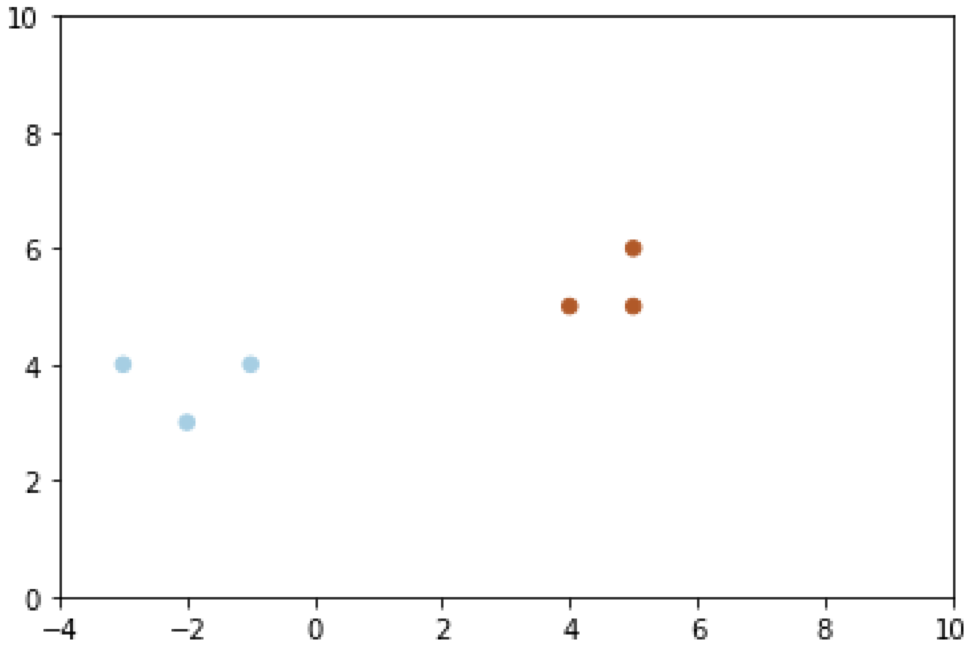
```
x: [-1 -2 -3  5  4  5]
y: [4 3 4 6 5 5]
```

[ ]:

```
plt.xlim(-4, 10)
plt.ylim(0, 10)
plt.scatter(x, y , c = labels, s = 30, cmap = plt.cm.Paired)
```

[ ]:

```
<matplotlib.collections.PathCollection at 0x7fbd1b36f400>
```



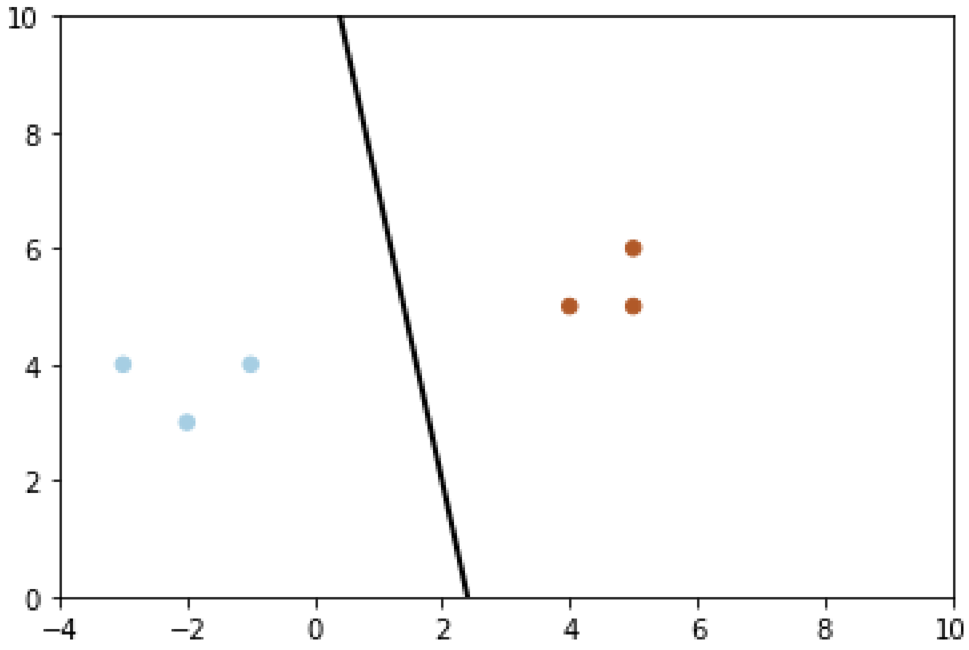
```
[ ]: # find a, b, c for the decision boundary ax + by + c1 = 0
     # a = w[0]; b = w[1]; c = classifier.intercept_
     w = classifier.coef_[0]
     c1 = classifier.intercept_[0]

     print("classifier.coef_:", classifier.coef_)
     print("classifier.intercept_:", classifier.intercept_)
     print("w:", w)
```

```
classifier.coef_: [[0.38461538 0.07692308]]
classifier.intercept_: [-0.92307692]
w: [0.38461538 0.07692308]
```

```
[ ]: # Get the decision boundary equation (the separating hyperplane)
     m = -w[0] / w[1] # slope
     xx = np.linspace(-4, 8)
     yy = m * xx - c1/w[1]
```

```
[ ]: # Draw the decision boundary
     plt.scatter(x, y, c = labels, s = 30, cmap = plt.cm.Paired)
     plt.xlim(-4, 10)
     plt.ylim(0, 10)
     plt.plot(xx, yy, linewidth = 2, color = 'black')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7fbd1b481fd0>]
```



[ ]:

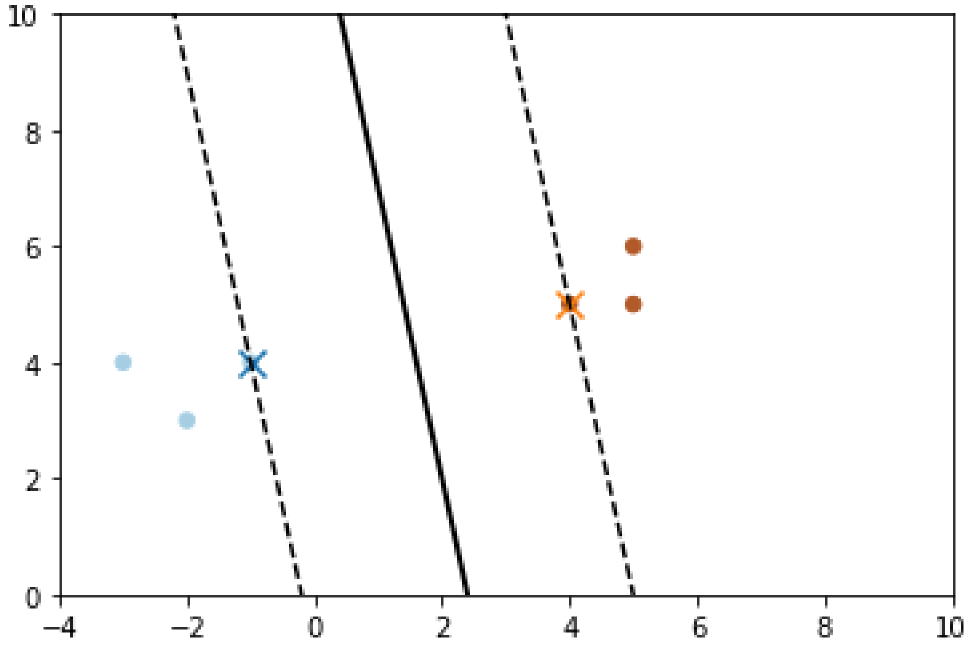
```
# find support vectors
sv1 = classifier.support_vectors_[0]
sv2 = classifier.support_vectors_[1]

print("classifier.support_vectors_:", classifier.support_vectors_)
print("Support Vector 1:", sv1)
print("Support Vector 2:", sv2)
```

```
classifier.support_vectors_: [[-1.  4.]
 [ 4.  5.]]
Support Vector 1: [-1.  4.]
Support Vector 2: [4. 5.]
```

[ ]:

```
# Draw the parallel hyperplanes that pass through the support vectors
yy_down = m * (xx - sv1[0]) + sv1[1]
yy_up = m * (xx - sv2[0]) + sv2[1]

plt.scatter(x, y, c = labels, s = 30, cmap = plt.cm.Paired)
plt.xlim(-4, 10)
plt.ylim(0, 10)
plt.plot(xx, yy, linewidth = 2, color = 'black')
plt.plot(xx, yy_down, 'k--')
plt.plot(xx, yy_up, 'k--')
plt.scatter(sv1[0], sv1[1], marker = "x", s = 100)
plt.scatter(sv2[0], sv2[1], marker = "x", s = 100)
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7fbd1b5ee460>
```



```
[ ]: print("Predicting what class (5, 4) is a part of:", classifier.predict([[5,4]]))
```

```
Predicting what class (5, 4) is a part of: [1]
```

```
[ ]: print("Predicting what class (-2, 1) is a part of:", classifier.
      ↪predict([[-2,1]]))
```

```
Predicting what class (-2, 1) is a part of: [-1]
```