Worksheet 15

Name: Jeong Yong Yang, Junho Son UID: U95912941, U64222022

Topics

- Support Vector Machines

Support Vector Machines

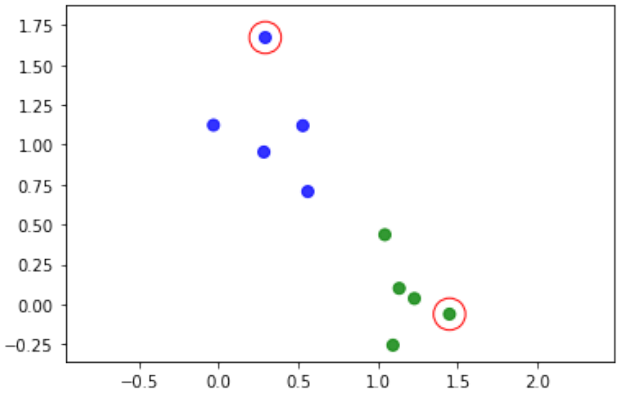
a) Follow along in class to implement the perceptron algorithm and create an animation of the algorithm.

```
import numpy as np
from PIL import Image as im
import matplotlib.pyplot as plt
import sklearn.datasets as datasets

TEMPFILE = "temp.png"
CENTERS = [[0, 1], [1, 0]]

# Dataset
X, labels = datasets.make_blobs(n_samples=10, centers=CENTERS, cluster_std=0.2, random_state=0)
Y = np.array(list(map(lambda x : -1 if x == 0 else 1, labels.tolist())))

# Initializing w and b
w = np.array([1, 1])
b = 0.1

# Perceptron Parameters
epochs = 100
alpha = .05
expanding_rate = .99
retracting_rate = 1.1

def snap(x, w, b, error):
    """
        Plot the street induced by w and b.
        Circle the point x in red if it was
        misclassified or in yellow if it was
        classified correctly.
    """

    xplot = np.linspace(-3, 3)
    cs = np.array([x for x in 'gb'])

    svm = - (w[1]/w[0]) * xplot - b/w[0]
    left_svm = - (1/w[0]) - (w[1]/w[0]) * xplot - b/w[0]
    right_svm = (1/w[0]) - (w[1]/w[0]) * xplot - b/w[0]

    fig, ax = plt.subplots()
    ax.scatter(X[:,0],X[:,1],color=cs[labels].tolist(), s=50, alpha=0.8)
    if error:
        ax.add_patch(plt.Circle((x[0], x[1]), .2, color='r',fill=False))
    else:
        ax.add_patch(plt.Circle((x[0], x[1]), .2, color='y',fill=False))
    ax.plot(xplot, left_svm, 'g--', lw=2)
    ax.plot(xplot, svm, 'r-', lw=2)
    ax.plot(xplot, right_svm, 'b--', lw=2)
    ax.set_xlim(min(X[:, 0]) - 1, max(X[:,0]) + 1)
    ax.set_ylim(min(X[:, 1]) - 1, max(X[:,1]) + 1)
    fig.savefig(TEMPFILE)
    plt.close()

    return im.fromarray(np.asarray(im.open(TEMPFILE)))
```

```
    return im.fromarray(np.asarray(im.open(TEMPFILE)))


images = []
for _ in range(epochs):
    # pick a point from X at random
    i = np.random.randint(0, len(X))
    x, y = X[i], Y[i]
    error = False

    y_predict = np.dot(w, x) + b
    if not((y < 0 and y_predict < 0) or (y > 0 and y_predict > 0)):
      # misclassified
      error = True

      # move in the direction of the point
      w = w + y * x * alpha
      b = b + y * alpha

      #expand
      w = w * expanding_rate
      b = b * expanding_rate
    else:
        # are you in the street
        if (y_predict < 1) and (y_predict > -1):
          # yes
          w = w * retracting_rate
          b = b * retracting_rate



    images.append(snap(x, w, b, error))

images[0].save(
    'svm.gif',
    optimize=False,
    save_all=True,
    append_images=images[1:],
    loop=0,
    duration=100
)
```

b) Consider the following dataset:

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

centers = [[0, 1], [1, 0]]
X, _ = datasets.make_blobs(n_samples=10, centers=centers, cluster_std=0.3, random_state=0)
Y = np.array([1 if x[0] - x[1] >= 0 else 0 for x in X])

cs = np.array([x for x in 'bg'])
_, ax = plt.subplots()
ax.scatter(X[:,0],X[:,1],color=cs[Y].tolist(), s=50, alpha=0.8)
ax.set_aspect('equal', adjustable='datalim')
ax.add_patch(plt.Circle((X[0][0], X[0][1]), .1, color='r',fill=False))
ax.add_patch(plt.Circle((X[1][0], X[1][1]), .1, color='r',fill=False))
plt.show()
```



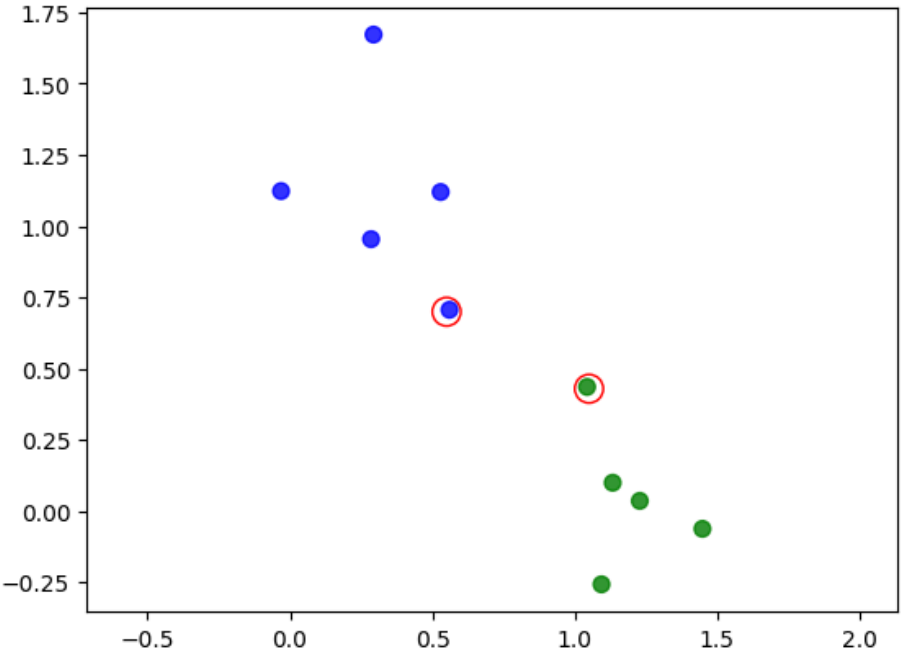
if we fit an SVM to the above dataset, moved the points circled in red, and re-fit the SVM, describe how the fit would change depending on how the points are moved.

Since the two points are not support vectors, the fit would not cause much of a change unless they are moved to become the new support vectors of the graph.

c) If we were to fit an SVM to the above dataset, which points do you think would affect the decision boundary the most? Circle them in red.

The two points that are in the center (the support vectors - the blue and green points that are closest to each other), would affect the decision boundary the most. Below is a photo of the points.

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.datasets as datasets
import matplotlib.patches as patches

centers = [[0, 1], [1, 0]]
X, _ = datasets.make_blobs(n_samples=10, centers=centers, cluster_std=0.3, random_state=0)
Y = np.array([1 if x[0] - x[1] >= 0 else 0 for x in X])

cs = np.array([x for x in 'bg'])
_, ax = plt.subplots()
ax.scatter(X[:,0],X[:,1],color=cs[Y].tolist(), s=50, alpha=0.8)
ax.set_aspect('equal', adjustable='datalim')
circleOne = patches.Circle((1.05, 0.43), radius=0.05, edgecolor='r', facecolor='none')
ax.add_patch(circleOne)
circleTwo = patches.Circle((0.55, 0.7), radius=0.05, edgecolor='r', facecolor='none')
ax.add_patch(circleTwo)
plt.show()
```


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.5
0.0
0.5
1.0
1.5
2.0