worksheet_15

March 31, 2024

1 Worksheet 15

Name: **Bowen Li**
UID: **U79057147**

1.0.1 Topics

- Support Vector Machines

1.1 Support Vector Machines

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

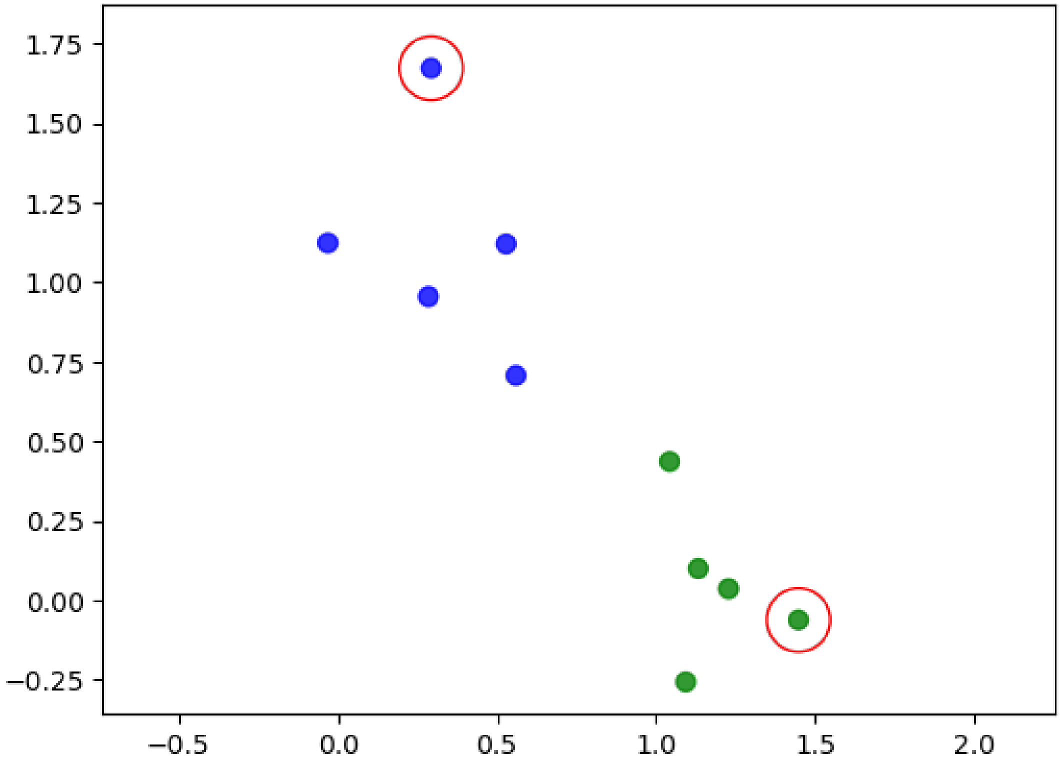
[22]:
```
import numpy as np
from PIL import Image as im
import matplotlib.pyplot as plt
import sklearn.datasets as datasets

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 == 1 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)))


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 = w[0] * x[0] + w[1] * x[1] + b
    if (y < 0 and y_predict < 0) or (y >= 0 and y_predict >= 0):
        # classified correctly
        # are you in the street?
        if y_predict < 1 and y_predict > -1:
            # in the street
            w = w + x * y * alpha * retracting_rate
            b = b + y * alpha * retracting_rate
        else:
```

```
            w = w * expanding_rate
            b = b * expanding_rate
    else:
        # misclassified
        w = w + x * y * alpha * expanding_rate
        b = b + y * alpha * expanding_rate

        error = True

    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:

[4]:
```
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()
```



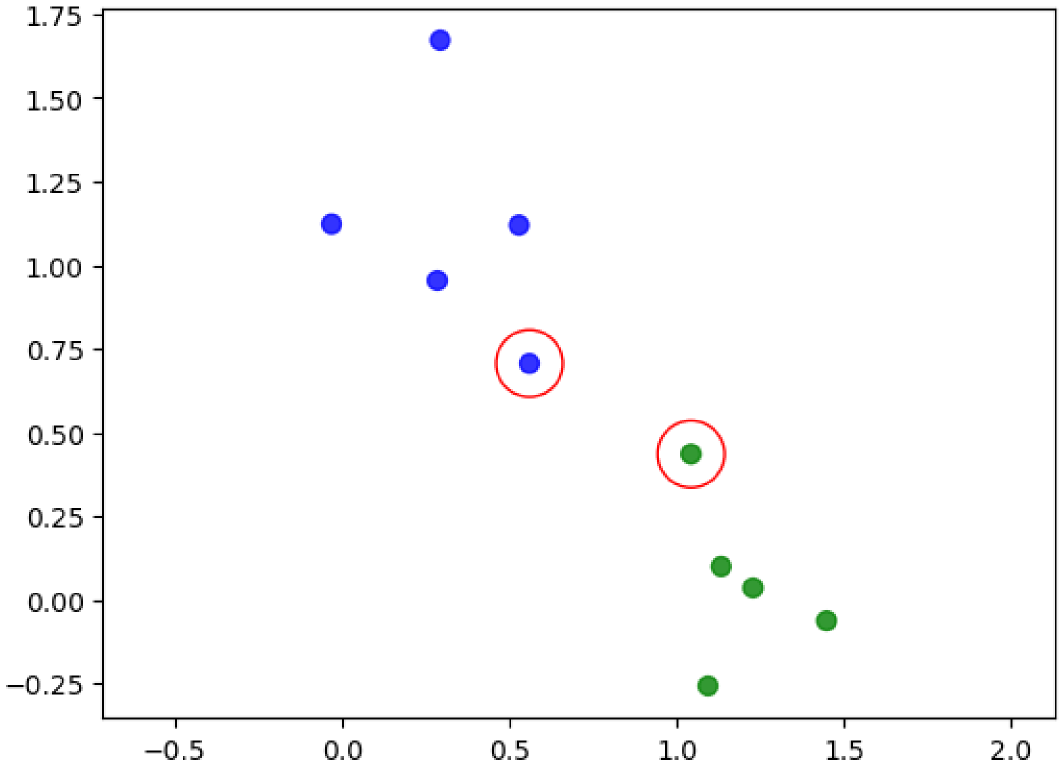
if we fit an SVM to the above dataset, moved the points circled in red, and re-fit the SVM, describe how the fit would change depending on how the points are moved.

The fit of the SVM wouldn't change much if the circled points were moved just a little since they would be far from the decision boundary and margins. The fit would only change if the points were moved very close to the original fit's decision boundary.

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 points closest to the decision boundary would affect it the most.

[5]:
```
_, 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[2][0], X[2][1]), .1, color='r',fill=False))
ax.add_patch(plt.Circle((X[6][0], X[6][1]), .1, color='r',fill=False))
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