svm_admissions

September 30, 2022

The most basic way goes direct to libsvm using the original libsvm authors' code, i.e. without using scikit-learn as a wrapper.

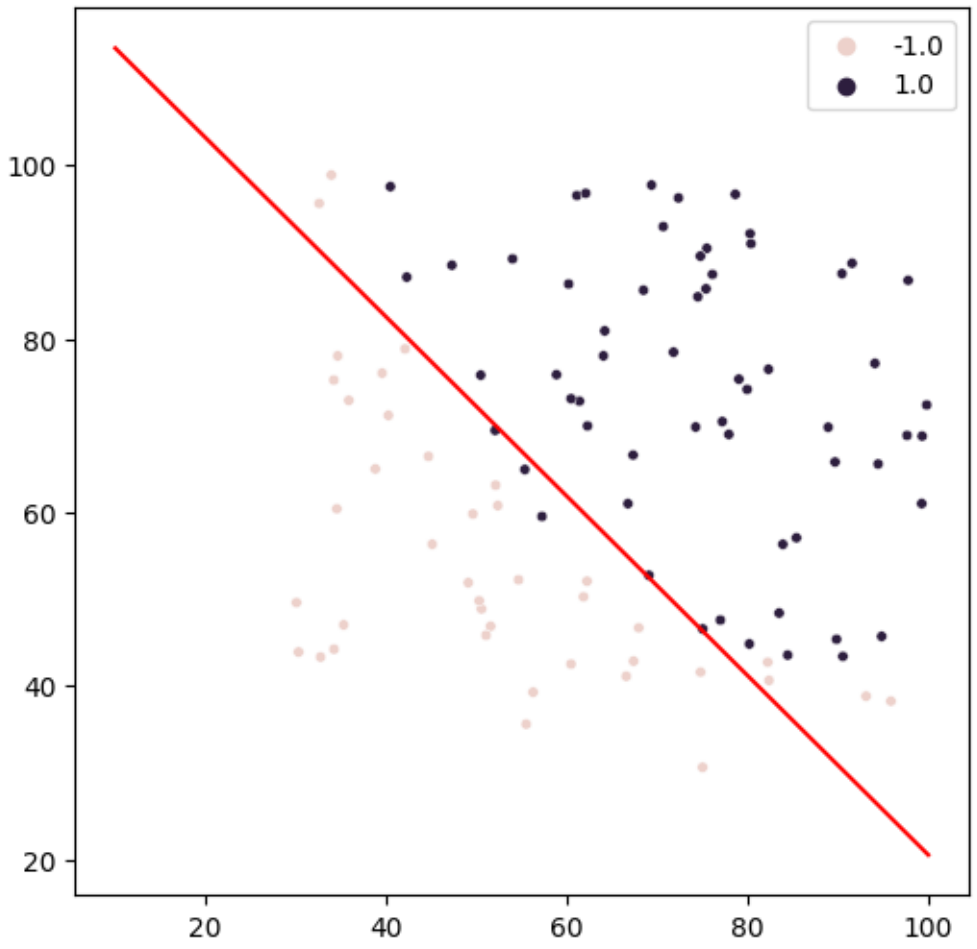
[1]:
```
from libsvm import svmutil
import os

# Read data in LIBSVM format
myfile = "".join([os.environ['HOME'], "/marks.libsvm"])
y, libsvm_x = svmutil.svm_read_problem(myfile) # y: ndarray, x: csr_matrix
m = svmutil.svm_train(y, libsvm_x, '-c 4')
```

```
.*.*
optimization finished, #iter = 217
nu = 0.219116
obj = -43.822177, rho = -0.184691
nSV = 100, nBSV = 0
Total nSV = 100
```

Alternately, using the sklearn wrapper, we can create a figure very much like the one in the lecture notes!

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC

## convert the libsvm format into regular numpy matrix
X = np.zeros((len(libsvm_x), 2))

for i in range(len(libsvm_x)):
    X[i,0] = libsvm_x[i][1]
    X[i,1] = libsvm_x[i][2]

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

plt.figure(figsize=(6, 6))# Plotting our two-features-space

sns.scatterplot(x=X[:, 0],
```

```
                y=X[:, 1],
                hue=y,
                s=16);

# Constructing a hyperplane using a formula.
w = clf.coef_[0]           # w consists of 2 elements
b = clf.intercept_[0]      # b consists of 1 element
x_points = np.linspace(10, 100)
y_points = -(w[0] / w[1]) * x_points - b / w[1]  # getting corresponding␣
 ↪y-points
# Plotting a red hyperplane

plt.plot(x_points, y_points, c='r');
```



[ ]: