CME 216, ME 343 - Winter 2020

Eric Darve, ICME

scikit-learn

To demonstrate how SVM works we are going to use scikit-learn.

The results in this section can be reproduced using this shared notebook.

The scikit-learn library can perform many important computations in machine learning including supervised and unsupervised learning.

See scikit supervised learning for more details about the functionalities that are supported.

We are going to demonstrate our concept through a simple example.

Let's generate 8 random points in the 2D plane.

Points in the top left are assigned the label -1 $(y > x)$ and points in the bottom right are assigned a label -1 $(y < x)$.

In Python, we set up two arrays X (coordinates) and y (labels) with the data:

```
print('Shape of X: ', X.shape)
print(X)
print(' Shape of y: ', y.shape)
print(y)
```

```
Shape of X:  (8, 2)
[[-0.1280102  -0.94814754]
 [ 0.09932496 -0.12935521]
 [-0.1592644  -0.33933036]
 [-0.59070273  0.23854193]
 [-0.40069065 -0.46634545]
 [ 0.24226767  0.05828419]
 [-0.73084011  0.02715624]
 [-0.63112027  0.5706703 ]]
Shape of y:  (8,)
[ 1.  1.  1. -1.  1.  1. -1. -1.]
```

Computing the separating hyperplane is done using

```
from sklearn import svm
clf = svm.SVC(kernel="linear",C=1e6)
clf.fit(X, y)
```

`clf` now contains all the information of the SVM. We will learn later on what the constant `C` is.

To visualize the result, we can plot the black solid line that separates the points.

Recall that the equation of the line is

$$w^T x + b = 0$$

w is given by `clf.coef_`:

```
print(clf.coef_)
[[ 2.1387469  -2.62113502]]
```

b is given by `clf.intercept_`:

```
print(clf.intercept_)
[0.63450173]
```

We can plot the points and line using Plotly syntax

```
x = np.linspace(-1, 1, 2)
a = -clf.coef_[0,0] / clf.coef_[0,1]
b = -clf.intercept_ / clf.coef_[0,1]
fig.add_trace(go.Scatter(x=x, y=a*x + b))
```

We can also visualize the support vectors.

There are 3 points in this case.

```
print(clf.support_vectors_)
[[-0.73084011  0.02715624]
 [-0.40069065 -0.46634545]
 [ 0.24226767  0.05828419]]
```

The lines going through these points satisfy the equations

Top line

$$w^T x + b = -1$$

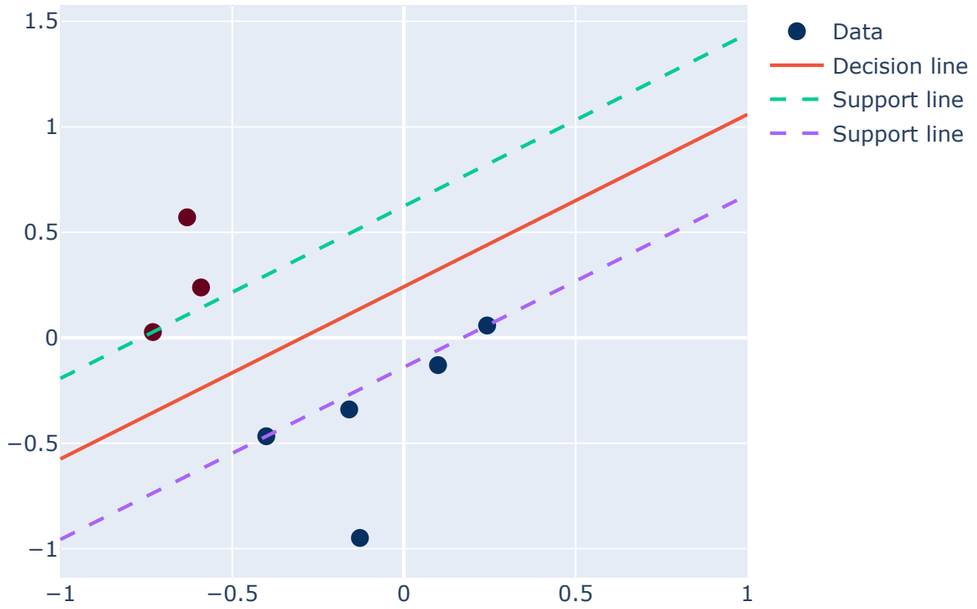
Bottom line

$$w^T x + b = 1$$

These lines can be plotted using

```
# green line
b1 = -(1 + clf.intercept_) / clf.coef_[0,1]
fig.add_trace(go.Scatter(x=x, y=a*x + b1))
# purple line
b2 = -(-1 + clf.intercept_) / clf.coef_[0,1]
fig.add_trace(go.Scatter(x=x, y=a*x + b2))
```

SVM


1.5
1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1
Data
Decision line
Support line
Support line

The orange line is the "farthest" away from the red and blue dots.

All the support vectors are at the same distance from the orange line.

The decision function, equal to $w^T x + b$ in our notations, can be computed using

```
clf.decision_function(X)
```

where `X` contains the coordinates of the points where the function is to be evaluated.

Decision function