Support Vector Machine

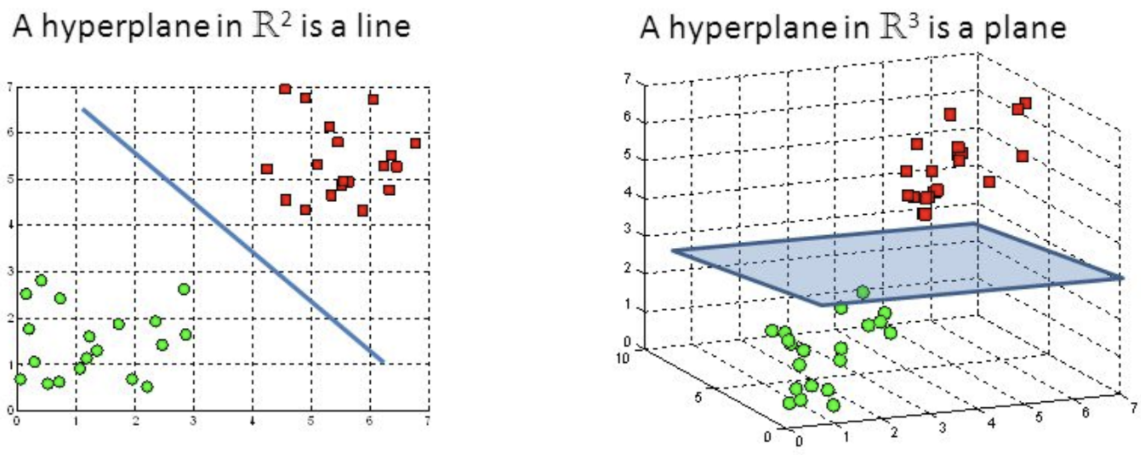
Support Vector Machines(SVMs) are particularly powerful and flexible class of supervised algorithms for both classification and regression. With this assignment we will be going to look into what is Support vector machine under hood, and see what it does. Given a set of trained examples, each examples are marked as belonging to a particlular category using this algorithm or predicting the values of a given data from trained examples.

Support vector machine is another simple algorithm that every machine learning expert should have in his/her arsenal. Support vector machine is highly preferred by many as it produces significant accuracy with less computation power.

What is support vector machine?

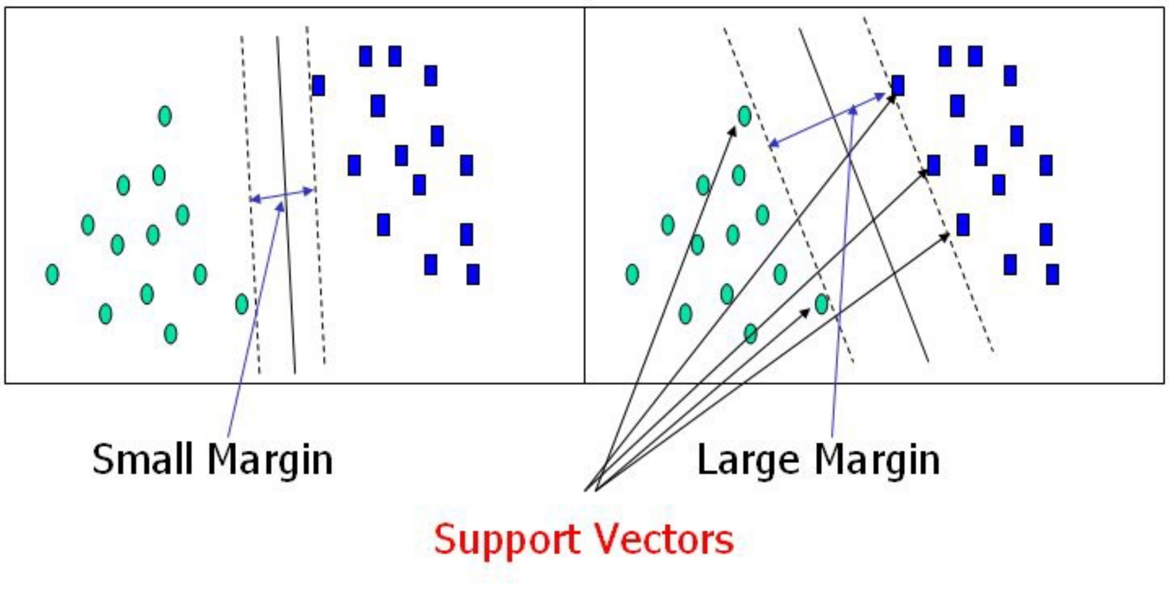
The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



Hyperplanes and Support Vectors!

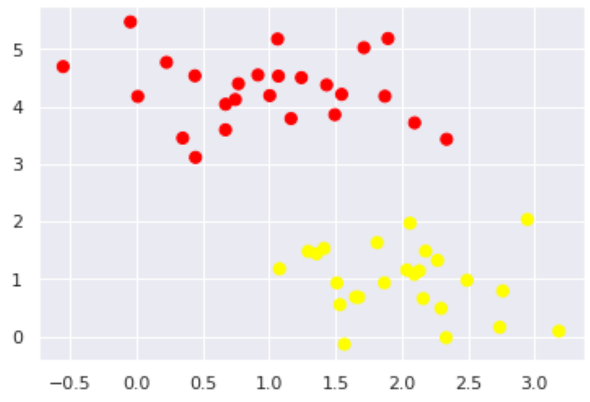
Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

As an example of this, consider the simple case of a classification task, in which the two classes of points are well separated. Our aim is to simply find a line or curve (in two dimensions) or manifold (in multiple dimensions) that divides the classes from each other.

In [10]:
```
# creating a dataset

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# use seaborn plotting defaults
import seaborn as sns; sns.set()

from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=50, centers=2,
                  random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
```

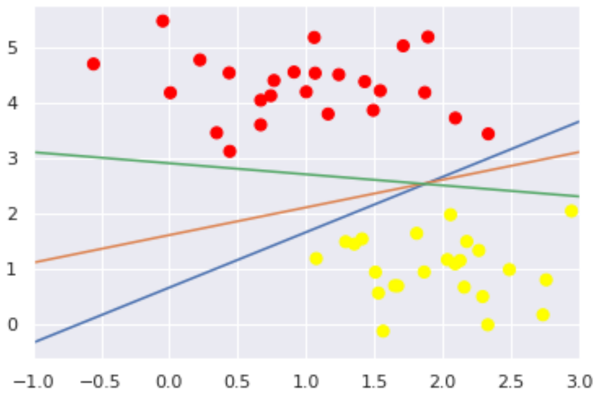


A linear discriminative classifier would attempt to draw a straight line separating the two sets of data, and thereby create a model for classification. For two dimensional data like that shown here, this is a task we could do by hand. But immediately we see a problem: there is more than one possible dividing line that can perfectly discriminate between the two classes!

We can draw them as follows:

In [22]:
```
xfit = np.linspace(-1,3.5)
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')

for m, b in [(1,0.65), (0.5,1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m*xfit + b)

plt.xlim(-1,3,5)
```

Out[22]:
```
(-1.0, 3.0)
```

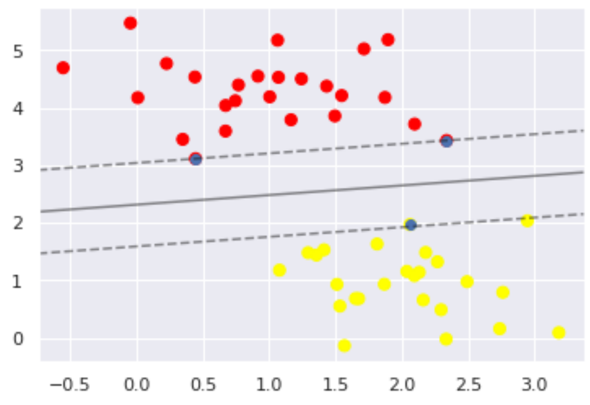


Support vector machines offer one way to improve on this. The intuition is this: rather than simply drawing a zero-width line between the classes, we can draw around each line a margin of some width, up to the nearest point.

Let's see the result of an actual fit to this data: we will use `Scikit-Learn's support vector classifier to train an SVM model on this data`. For the time being, we will use a linear kernel and set the C parameter to a very large number (we'll discuss the meaning of these in more depth momentarily).

We can visualize what's happening here, let's create a quick convenience function that will plot SVM decision boundaries for us:

In [48]:
```
from sklearn.svm import SVC

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

# plotting the SVM decision function
def plot_svc_decision_function(model, ax=None, plot_support=True):
    if ax is None:
        ax = plt.gca()

    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    ax.contour(X, Y, P, colors='k', levels=[-1,0,1],
              alpha=0.5, linestyles=['--','-','--'])

    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   (model.support_vectors_[:, 1]),
                   )

    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
```



This is the dividing line that maximizes the margin between the two sets of points. Notice that a few of the training points just touch the margin: they are indicated by the black circles in this figure. These points are the pivotal elements of this fit, and are known as the support vectors, and give the algorithm its name. In Scikit-Learn, the identity of these points are stored in the `support_vectors_` attribute of the classifier:

In [ ]:
```
mo
```

explain linear svm

In [ ]:
```
```