Support Vector Machines

This lesson will focus on training Support Vector Machines.

WE'LL COVER THE FOLLOWING

Support Vector Machine

A **Support Vector Machine (SVM)** is a supervised machine learning algorithm that can be used for both classification and regression problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane/line that differentiates the two classes. A **hyper-plane** is the higher dimensional version of a line. As a line can separate classes on two-dimensional data, a hyper-plane can separate on higher dimensions.

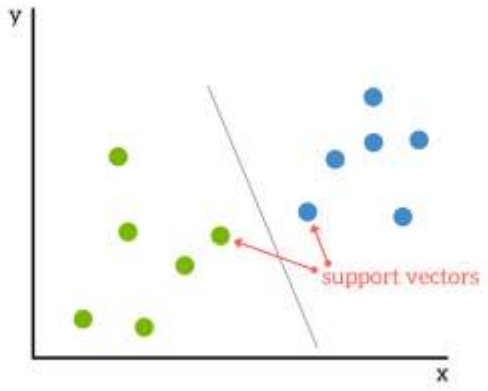


Image from: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

We can see that the line separates the two classes (green and blue) very well. These data points are called **support vectors**. Altering these points can alter the line. The black line is the **support vector machine** here.

When predicting classes, an observation is plotted, and if it falls on the left side of the line, it is classified as the green class, if it falls on the right side of the line, it is classified as the blue class.

Kernels

The example we saw above is too simple. Usually, classes are not easy to separate. Sometimes we cannot separate data easily by a line or a hyperplane. Therefore, SVMs are found by converting data into higher dimensions and finding hyper-planes on the higher dimensional data. This is known as the **kernel trick**. **Kernels** are functions that take low dimensional data and transform it into a higher dimensional space, i.e., they convert an inseparable problem to a separable problem. Simply put, they do some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs we have defined.

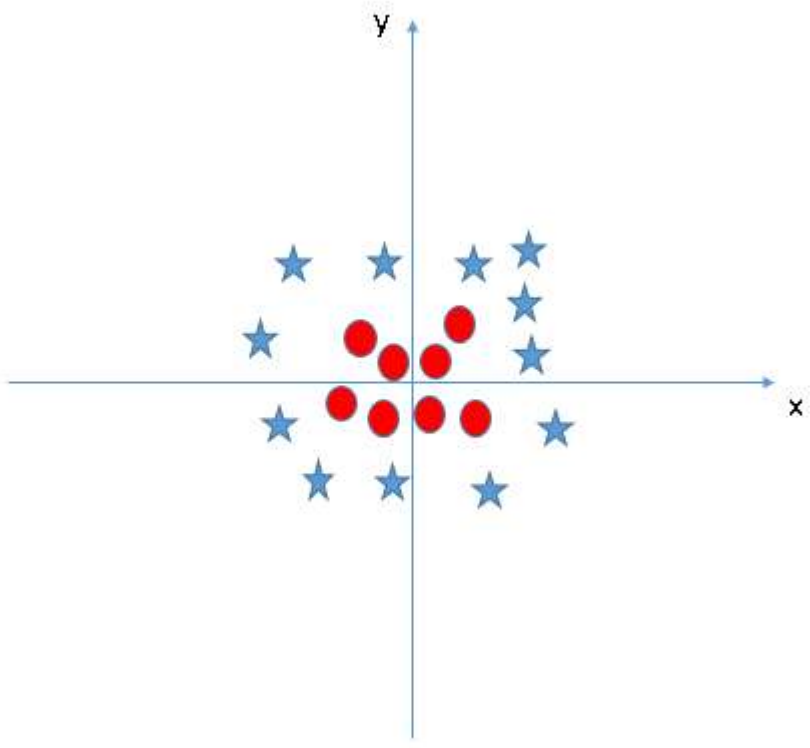


Image from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

In the example above, the data is two-dimensional (`x` and `y`). The points are plotted using the `x` and `y` values. We cannot draw a line to separate the two classes. Therefore, another dimension/feature is introduced. We call it `z`. It is defined as $z = x^2 + y^2$. This function that computes `z` is the kernel in this case. Now plotting the `x` and `z` values would give us:

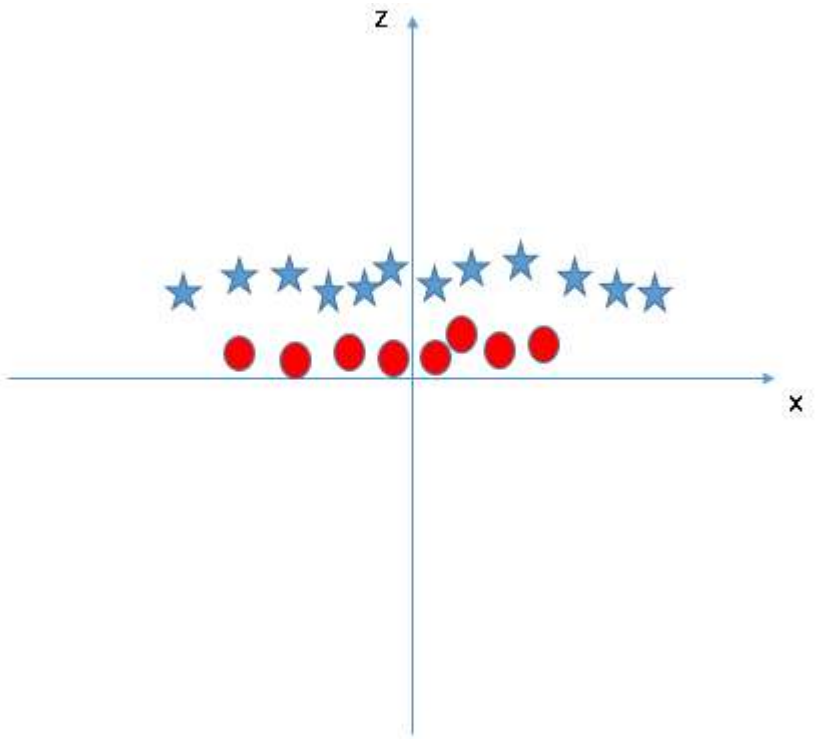


Image from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

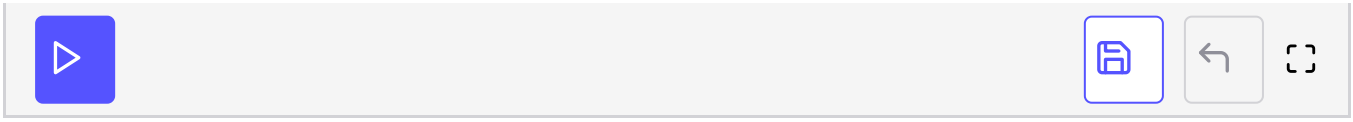
We can easily draw a line to separate the two classes. In this way, using a *kernel*, we transformed 2-dimensional data to 3-dimensional data and separated the classes.

SVM in Python

SVMs are available in `sklearn.svm` module. This module has many different SVMs based on different kernels. We will be using the `SVC` model. SVC stands for Support Vector Classification. We will be using the Audit Risk Dataset of different firms. We will be performing the binary classification task of predicting whether a company is fraudulent or not.

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score,classification_report

df = pd.read_csv('audit_data.csv')


# MAKE DATA
X = df.drop(columns = ['Risk','LOCATION_ID'])
Y = df[['Risk']]

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2, random_state = 2)

# MAKE MODEL
svm = SVC()
svm.fit(X_train,Y_train)

# CALCULATE AND PRINT RESULTS
preds = svm.predict(X_test)
acc = accuracy_score(y_true = Y_test,y_pred = preds)
print(acc)
print(classification_report(y_true = Y_test,y_pred = preds))
```



We import `SVC` in **line 2**. After we read the data in **line 6**, we separate our target variable as `Y`. To split the data into training and test sets, we use the function `train_test_split`. We provide our inputs `X` and the labels `Y` to the function in **line 12**. We also provide the test set size as `test_size`. 0.2 implies that 20% data will be included in the testing set, while the remaining 80% will form the training set. The function outputs 4 items that we store in `X_train`, `X_test`, `Y_train`, `Y_test`.

In **line 15**, we initialize our SVM classifier object just like we did in the last lesson. We call `SVC` without any arguments. Then in the next line, we call the `fit` function of the model. We provide the training examples and labels to the function. After this, we need to evaluate our model. Therefore, we use the `predict` function of the model in **line 19** to store predictions in `preds`. We give the testing inputs `X_test` to `predict` as an argument. We use `accuracy_score` function to measure the accuracy of the predictions. We print the accuracy with the classification report, which we obtained by using the `classification_report` function, in the last two lines.

We see that the model is approximately 98% accurate which is a greater prediction accuracy.

In the next lesson, we will look at an *ensemble* machine learning method

which is known as *boosting*.