SUPPOT VECTOR MACHINE

MOHAN M J

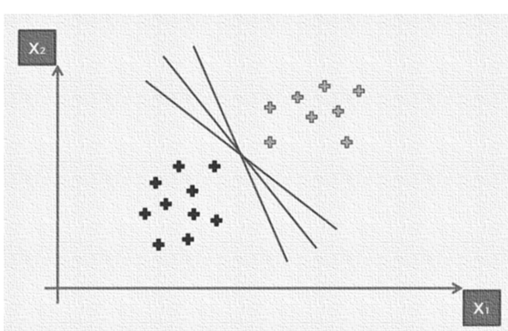
INTRODUCTION

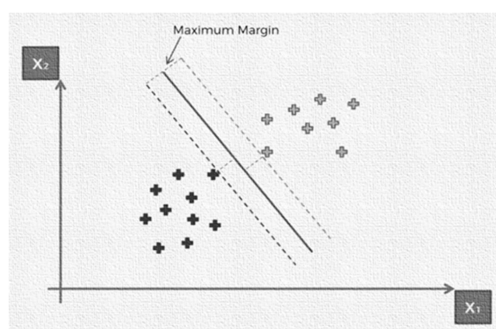
- One of the most popular and talked about machine learning algorithms
- They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning

MAXIMAL-MARGIN CLASSIFIER

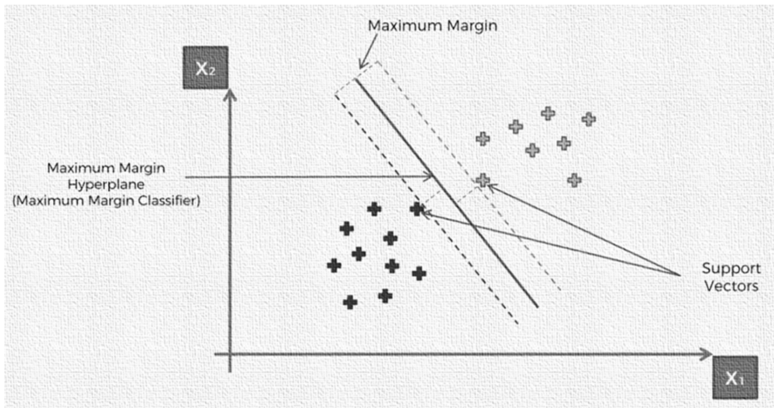
- The Maximal-Margin Classifier is a hypothetical classifier that best explains how SVM works in practice
- The numeric input variables (x) in your data (the columns) form an n-dimensional space. For example, if you had two input variables, this would form a two-dimensional space
- A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1.

CONTD..



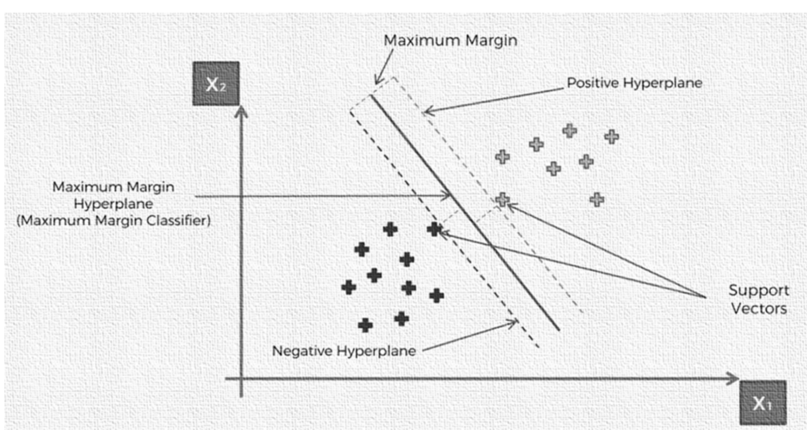


CONTD..



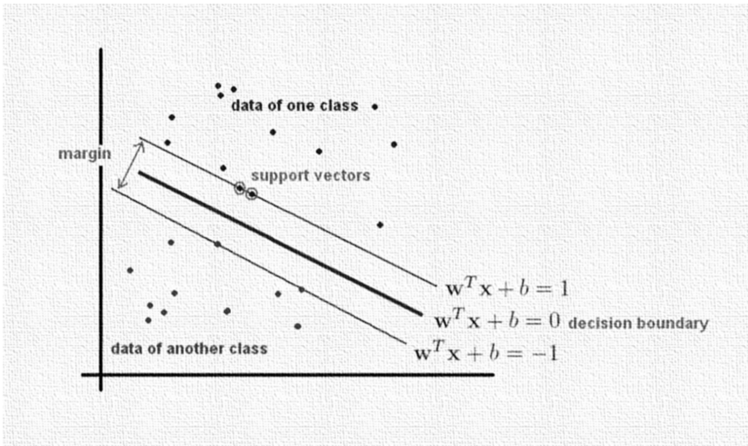
Maximum Margin
X₂
Maximum Margin
Hyperplane
(Maximum Margin Classifier)
Support
Vectors
X₁

CONTD..



Maximum Margin
Positive Hyperplane
X₂
Maximum Margin
Hyperplane
(Maximum Margin Classifier)
Support
Vectors
Negative Hyperplane
X₁

CONTD..



HYPERPLANE

- In two dimension, a hyperplane is a one dimension subspace namely a line
- In three dimension, a hyperplane is a flat two dimension subspace namely a plane
- In a p dimensional space, a hyperplane is a linear subspace of p-1 dimension
- Hyperplane in p dimension - Equation

  $w_0 + w_1 X_1 + w_2 X_2 + w_3 X_3 + \ldots\ldots + w_p X_p = 0$

CONTD..

Suppose for $X = X_1$, X_2 , X_3 …….. X_p

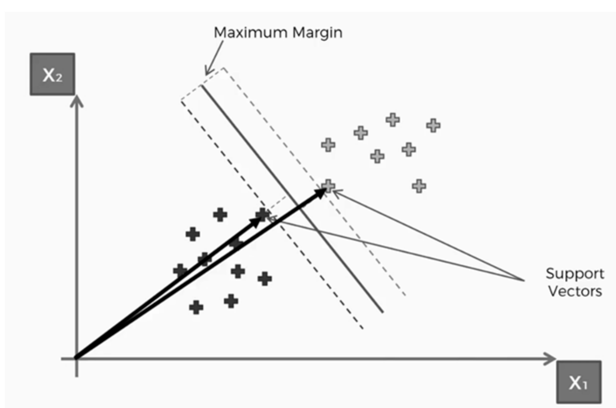
- If $w_0 + w_1 X_1 + w_2 X_2 + w_3 X_3 + \ldots\ldots + w_p X_p > 0$

  Then the X lies in one side of the hyperplane
- If $w_0 + w_1 X_1 + w_2 X_2 + w_3 X_3 + \ldots\ldots + w_p X_p < 0$

  Then the X lies in one the other side of the hyperplane
- Hyperplane divide p dimensional space into 2 halves
- A value close to the line returns a value close to zero and the point may be difficult to classify
- If the magnitude of the value is large, the model may have more confidence in the prediction

CONTD..

- Distance between the line and the closest data points is referred to as the margin
- Best or optimal line that can separate the two classes is the line that has the largest margin. This is called Maximal-Margin hyperplane.



SVM - QUADRATIC PROGRAMMING PROBLEM

$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ y_i are either 1 or −1

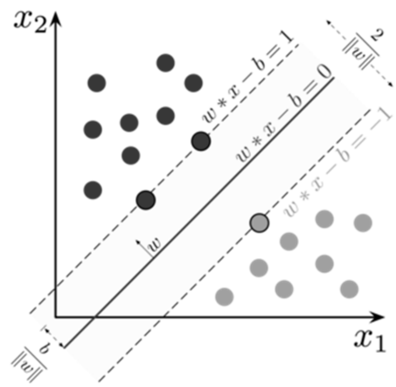
$\vec{w} \cdot \vec{x} - b = 0,$

$\vec{w} \cdot \vec{x} - b = 1$ or $\vec{w} \cdot \vec{x}_i - b \geq 1$, if $y_i = 1$

$\vec{w} \cdot \vec{x} - b = -1$ or $\vec{w} \cdot \vec{x}_i - b \leq -1$, if $y_i = -1$.

$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \quad$ for all $1 \leq i \leq n$.

"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$, for $i = 1, \ldots, n$"



EXERCISE

- Develop a model to predict the Iris plant class (1 : Iris-setosa, 2:Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length, petal width using SVM. The data is given in Iris_data.csv file.
- Validate the model with Iris_test.csv data

PYTHON CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, svm
iris = datasets.load_iris()
X = iris.data
print(X)
y = iris.target
print(y)
X = X[y != 0, :2]
y = y[y != 0]
n_sample = len(X)
```

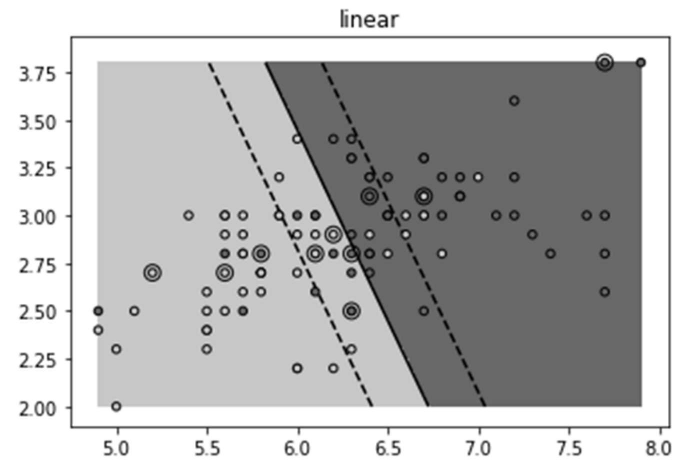
```
np.random.seed(0)
order = np.random.permutation(n_sample)
X = X[order]
y = y[order].astype(np.float)
X_train = X[:int(.9 * n_sample)]
y_train = y[:int(.9 * n_sample)]
X_test = X[int(.9 * n_sample):]
y_test = y[int(.9 * n_sample):]
```

CONTD..

```
classifier = svm.SVC(kernel='linear', gamma=10)
classifier.fit(X_train, y_train)
plt.figure()
plt.clf()
plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=plt.cm.Paired, edgecolor='k', s=20)
# Circle out the test data
plt.scatter(X_test[:, 0], X_test[:, 1], s=80, facecolors='none',  zorder=10, edgecolor='k')
plt.axis('tight')
x_min = X[:, 0].min()
x_max = X[:, 0].max()
```

```
y_min = X[:, 1].min()
y_max = X[:, 1].max()
XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
Z = classifier.decision_function(np.c_[XX.ravel(), YY.ravel()])
# Put the result into a color plot
Z = Z.reshape(XX.shape)
plt.pcolormesh(XX, YY, Z > 0, cmap=plt.cm.Paired)
plt.contour(XX, YY, Z, colors=['k', 'k', 'k'],
               linestyles=['--', '-', '--'], levels=[-.5, 0, .5])
plt.title('linear')
plt.show()
```

VISUALIZATION



THANK YOU