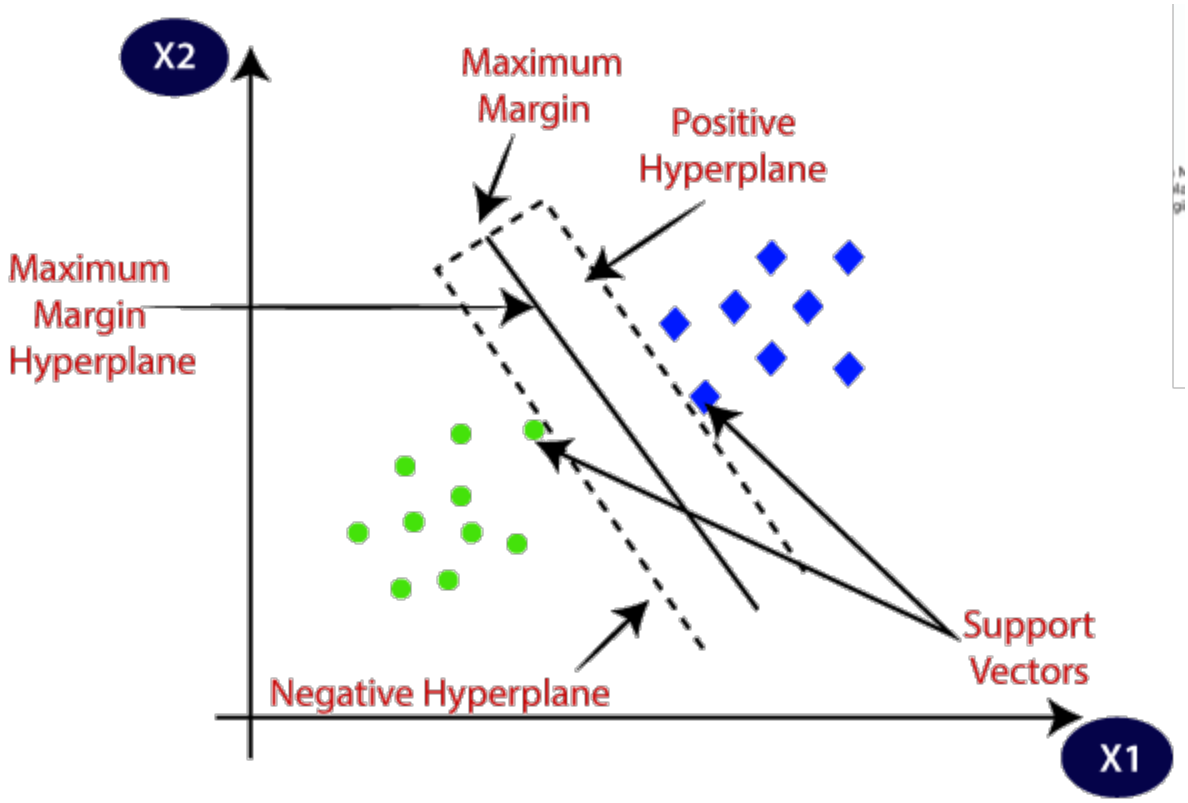


Support Vector Machine(with Numerical Example)

SVM is a one of the most popular supervised machine learning algorithm, which can be used for both classification and regression but mainly...

Balaji C

Follow

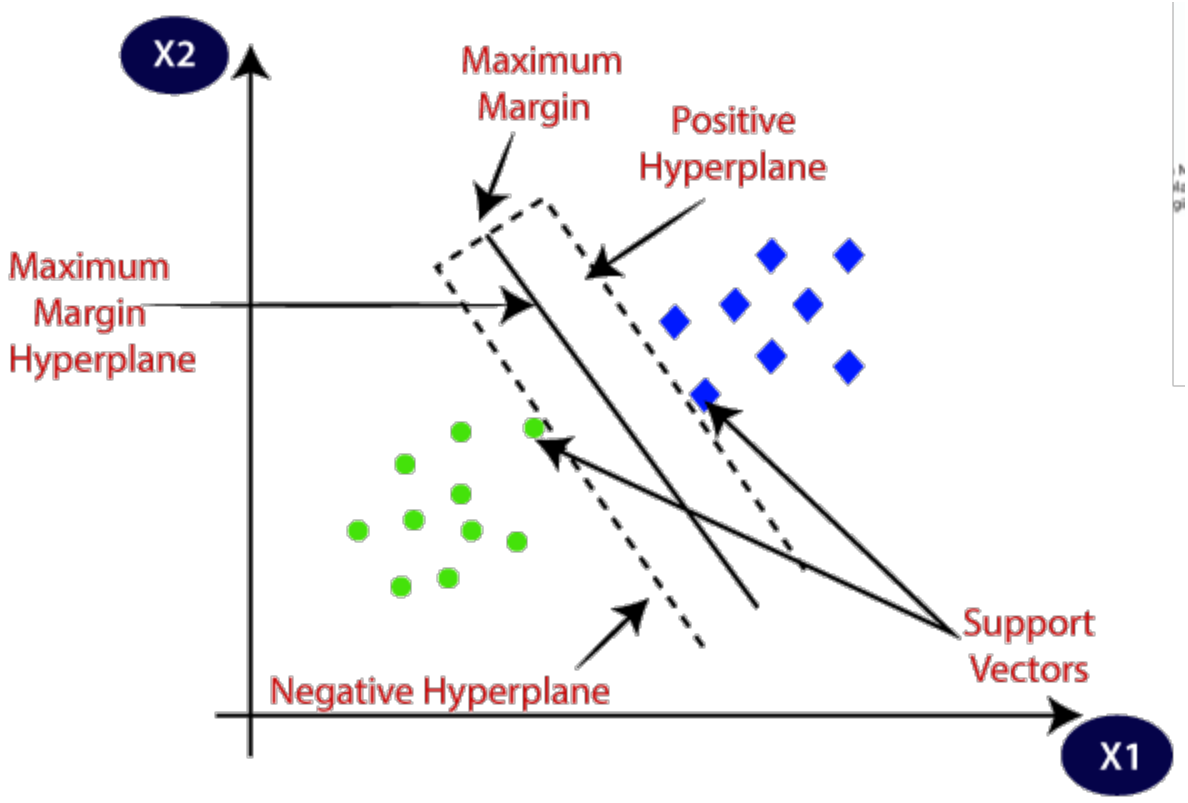
a11y-light · January 19, 2023 (Updated: January 20, 2023) · Free: Yes

SVM is a one of the most popular supervised machine learning algorithm, which can be used for both classification and regression

Main goal of SVM is to create best fit line or boundary that can segregate n-dimensional space into classes, so that we can put the data point in correct category for future prediction.

The best decision boundary is called **hyper-plane.**

SVM choses the extreme points/vectors that helps in creating a hyper-plane. These extreme cases are called as **support vectors.** Hence algorithm is termed as support vector machine.



Linear SVM

There are mainly two types of SVM

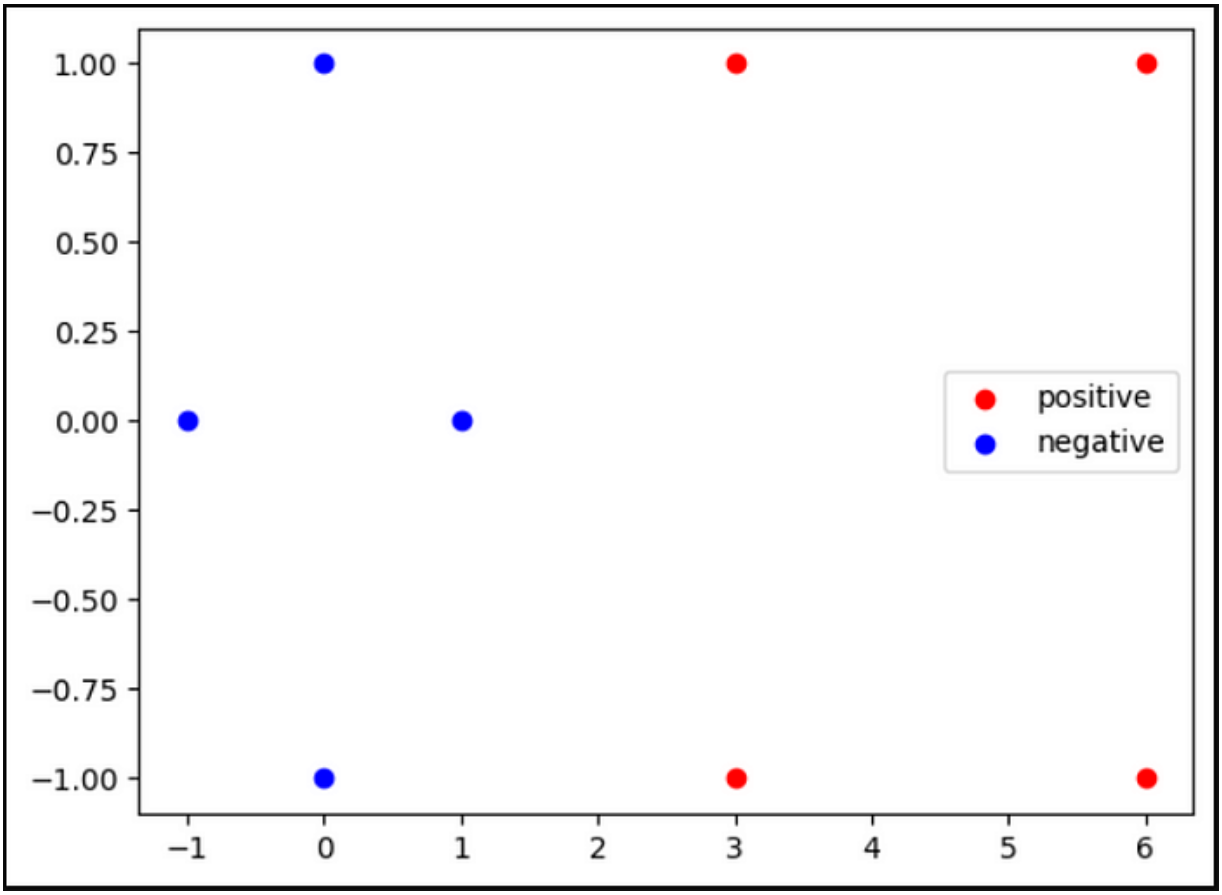
1. **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by

classifier.

Numerical example for Linear SVM:

Q. Positively labelled data points (3,1)(3,-1)(6,1)(6,-1) and Negatively labelled data points (1,0)(0,1)(0,-1)(-1,0)

Solution: for all negative labelled output is -1 and for all positive labelled output is 1.



graph for above table

Now adding 1 to all points

$s_6` = (0,1,1)$ $s_7 = (0,-1) => s_7 = (0,-1,1)$ $|s_8 = (-1,0) => s_8` = (-1,0,1)$

****

from the graph we can see there is one negative point $\alpha_1 = (1,0,1)$ and two positive points $\alpha_2 = (3,1,1)$ & $\alpha_3 = (3,-1,1)$ form support vectors

Generalized equation

$\alpha_1 * s_1` * s_1` + \alpha_2 * s_1` * s_2` + \alpha_3 * s_1` * s_3` = -1 \rightarrow 1$

$\alpha_1 * s_2` * s_1` + \alpha_2 * s_2` * s_2` + \alpha_3 * s_2` * s_3` = 1 \rightarrow 2$

$\alpha_1 * s_3` * s_1` + \alpha_2 * s_3` * s_2` + \alpha_3 * s_3` * s_3` = 1 \rightarrow 3$

****

On solving these equations

eq 1 => $\alpha_1 * (1,0,1) * (1,0,1) + \alpha_2 * (1,0,1) * (3,1,1) + \alpha_3 * (1,0,1) * (3,-1,1) = -1$

eq 2=> $\alpha_1 * (3,1,1) * (1,0,1) + \alpha_2 * (3,1,1) * (3,1,1) + \alpha_3 * (3,1,1) * (3,-1,1) = 1$

eq 3=> $\alpha_1 * (3,-1,1) * (1,0,1) + \alpha_2 * (3,-1,1) * (3,1,1) + \alpha_3 * (3,-1,1) * (3,-1,1) = 1$

****

$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1 \rightarrow 4$

$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = 1 \rightarrow 5$

$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = 1 \rightarrow 6$

on solving these equations 4,5 and 6 we get,

$\alpha_1 = -3.5$, $\alpha_2 = 0.75$ and $\alpha_3 = 0.75$

****

To find hyper-plane

$W` = \Sigma \alpha_i * s_i$

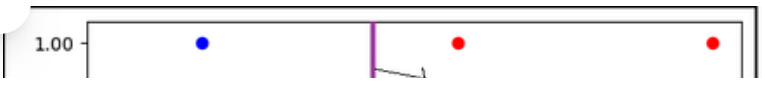
$W` = -3.5 * (1,0,1) + 0.75 * (3,1,1) + 0.75 * (3,-1,1)$

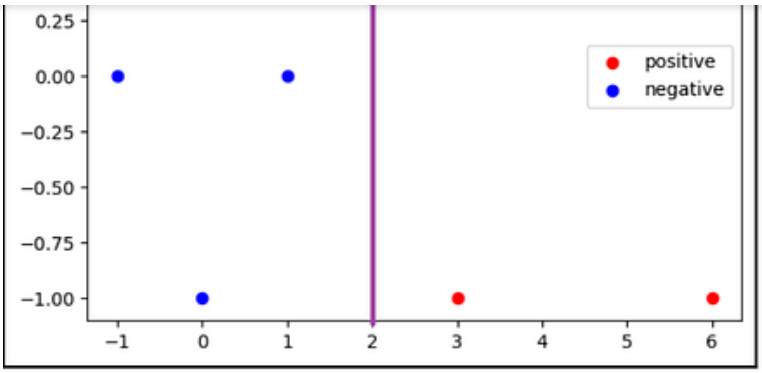
$W` = (1,0,-2)$

$y = W`x + b$

$W` = (1,0)$ and $b = 2$

so the best fit line or hyper plane is at (0,2) which splits the data points into two classes



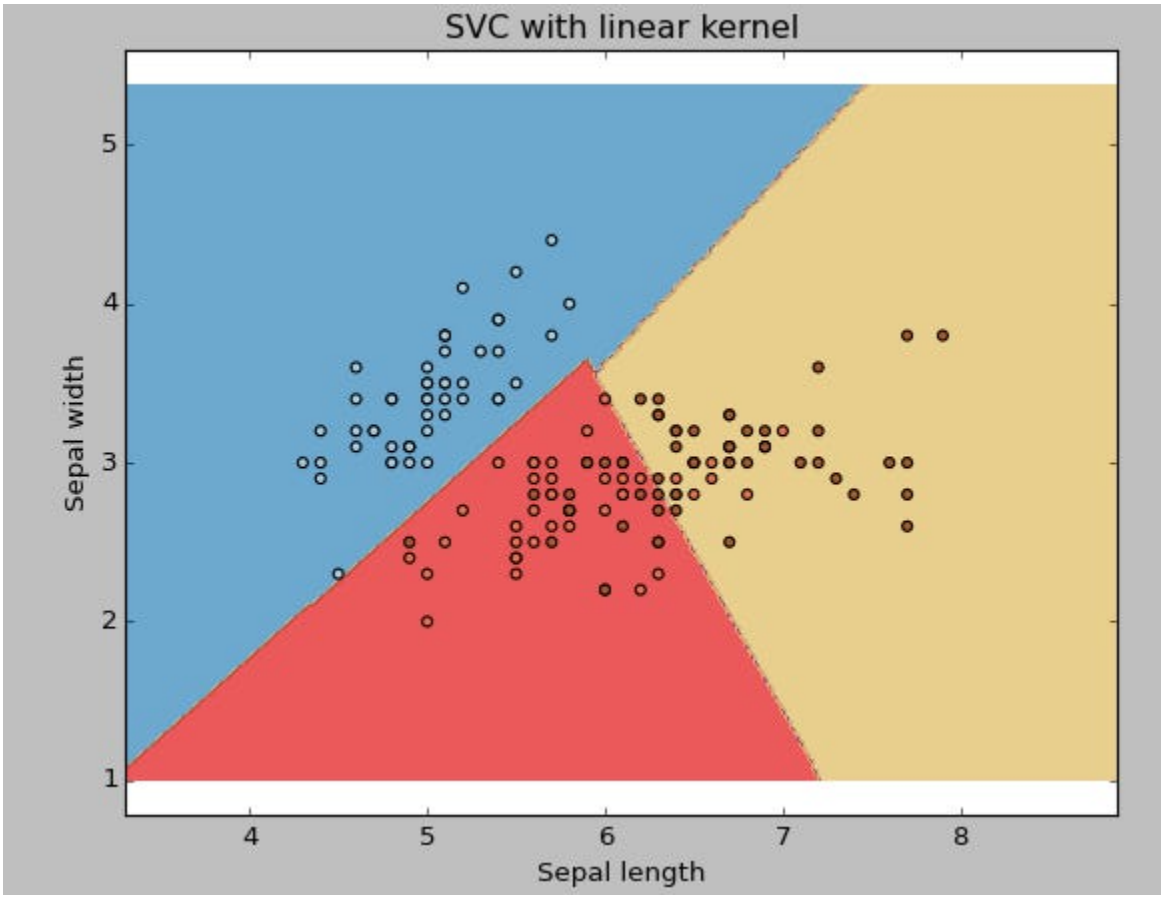


Final Graph for linear SVM

Copy

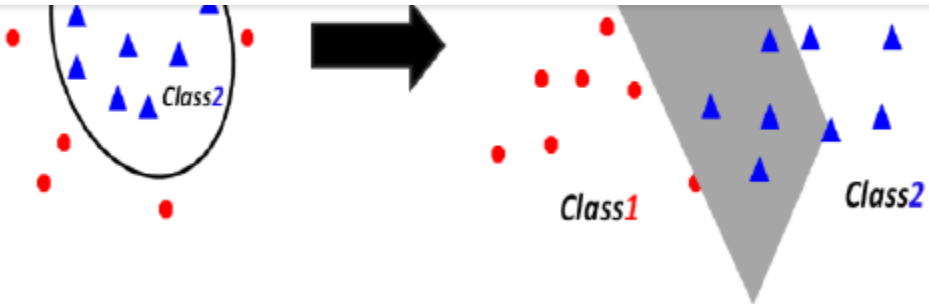
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target

C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=1,gamma=0).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
 np.arange(y_min, y_max, h))

plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
```

```
plt.title( 'SVC with linear kernel' )
plt.show()
```



2. Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

In non-linear SVM we will use Kernel SVM to convert low dimensional data into high dimensional data so that small hyper plane will be created.



Numerical example for Non-Linear SVM:

Q. Positively labelled data points (2,2)(2,-2)(-2,-2)(-2,2) and Negatively labelled data points (1,1)(1,-1)(-1,-1)(-1,1)

Solution: We have to find a hyper-plane i,e it divides data into two different classes. But this is of type non-linear SVM so we need to convert data from one feature space to another feature space using Kernal SVM.

Kernal SVM Condition:

$$\Phi_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 4 - x_2 + |x_1 - x_2| \\ 4 - x_1 + |x_1 - x_2| \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} > 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$

(1) For positive labelled data:

$\theta(2,2) \Rightarrow (\sqrt{2^2 + 2^2}) = (\sqrt{8}) > 2$

$\Rightarrow \theta(x_1) = 4 - 2 + |2 - 2| \Rightarrow \mathbf{\theta(x_1) = 2}$ and $\theta(x_2) = 4 - 2 + |2 - 2| \Rightarrow \mathbf{\theta(x_2) = 2}$

$\theta(2,-2) \Rightarrow (\sqrt{2^2+2^2}) = (\sqrt{8}) > 2$

so $\theta(x_1) = 4 + 2 + |2 + 2| \Rightarrow \mathbf{\theta(x_1) = 10}$ and $\theta(x_2) = 4 - 2 + |2 + 2| \Rightarrow \mathbf{\theta(x_2) = 6}$

****

$\theta(-2,-2) \Rightarrow (\sqrt{2^2+2^2}) = (\sqrt{8}) > 2$

so $\theta(x_1) = 4 + 2 + |-2 + 2| \Rightarrow \mathbf{\theta(x_1) = 6}$ and $\theta(x_2) = 4 + 2 + |-2 + 2| \Rightarrow \mathbf{\theta(x_2) = 6}$

****

$\theta(-2,2) \Rightarrow (\sqrt{2^2+2^2}) = (\sqrt{8}) > 2$

so $\theta(x_1) = 4 - 2 + |-2 - 2| \Rightarrow \mathbf{\theta(x_1) = 6}$ and $\theta(x_2) = 4 + 2 + |-2 - 2| \Rightarrow \mathbf{\theta(x_2) = 10}$

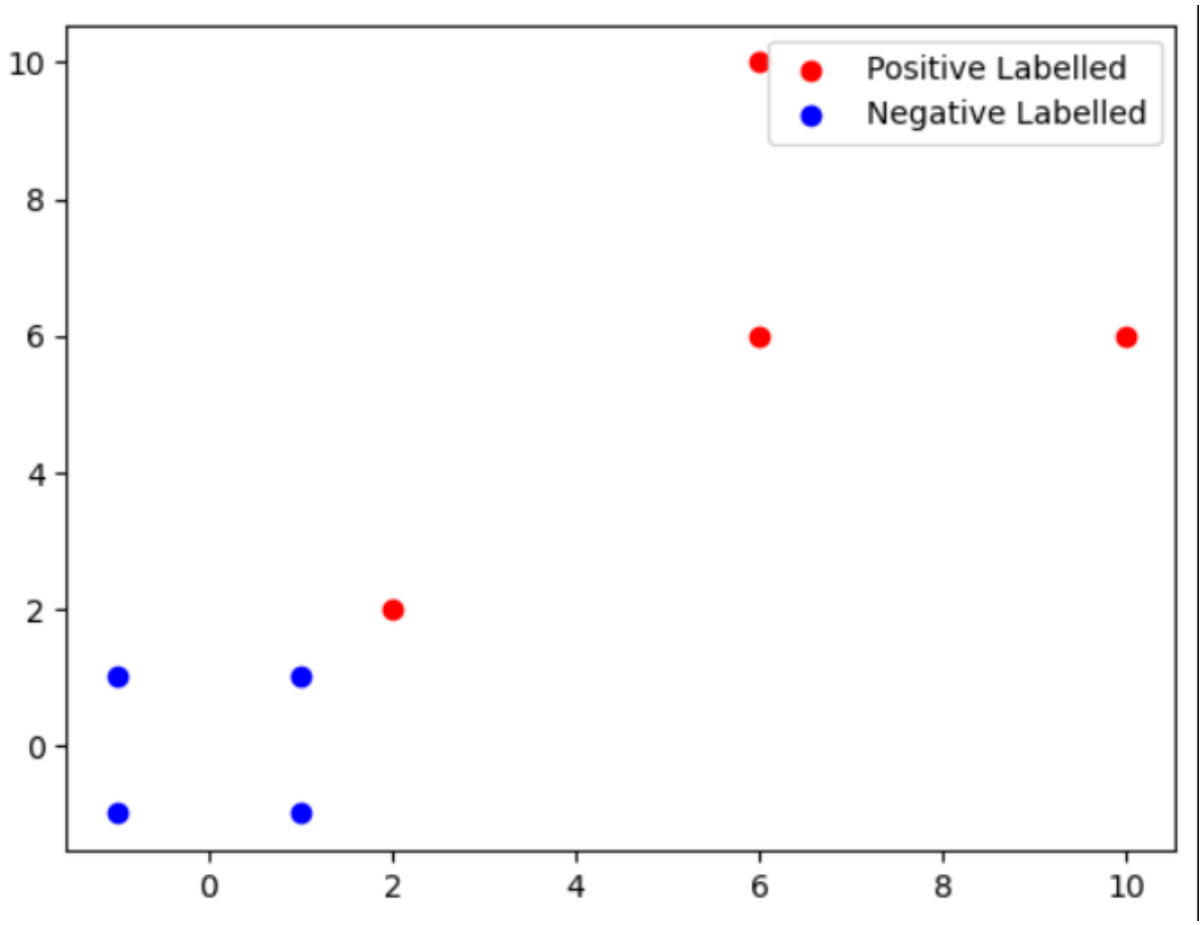
****

Hence overall positive labelled data is rechanged as (2,2), (10,6), (6,6), (6,10)

(2) For negative labelled data points:

Here all values are (1, 1) only thing is negative and positive sign changes, so if we square root them then all values will be less than 2.

Final plot is



From the graph we can say support vectors are S_1= (1,1) and S_2 = (2,2) so adding 1 to support vectors, $S_1`$ = (1,1,1) and $S_2`$ = (2,2,1).

$\alpha_1 * s_1` * s_1` + \alpha_2 * s_1` * s_2` = -1 \rightarrow 1$

$\alpha_1 * s_2` * s_1` + \alpha_2 * s_2` * s_2` = 1 \rightarrow 2$

$\alpha_1 * (1,1,1) * (1,1,1) + \alpha_2 (1,1,1) * (2,2,1) = -1 \rightarrow 1$

$_1 * (2,2,1) * (1,1,1) + \alpha_2 * (2,2,1) * (2,2,1) = 1 \rightarrow 2$

On solving equations $\alpha_1 = -7$ and $\alpha_2 = 4$

To find hyper-plane

$W` = \Sigma \alpha_i * s_i$

$W` = -7 * (1,1,1) + 4 * (2,2,1)$

$W` = (1,1,-3)$

$y = W`x + b$

$W` = 1,1$ and $b = 3$

on drawing graph

None

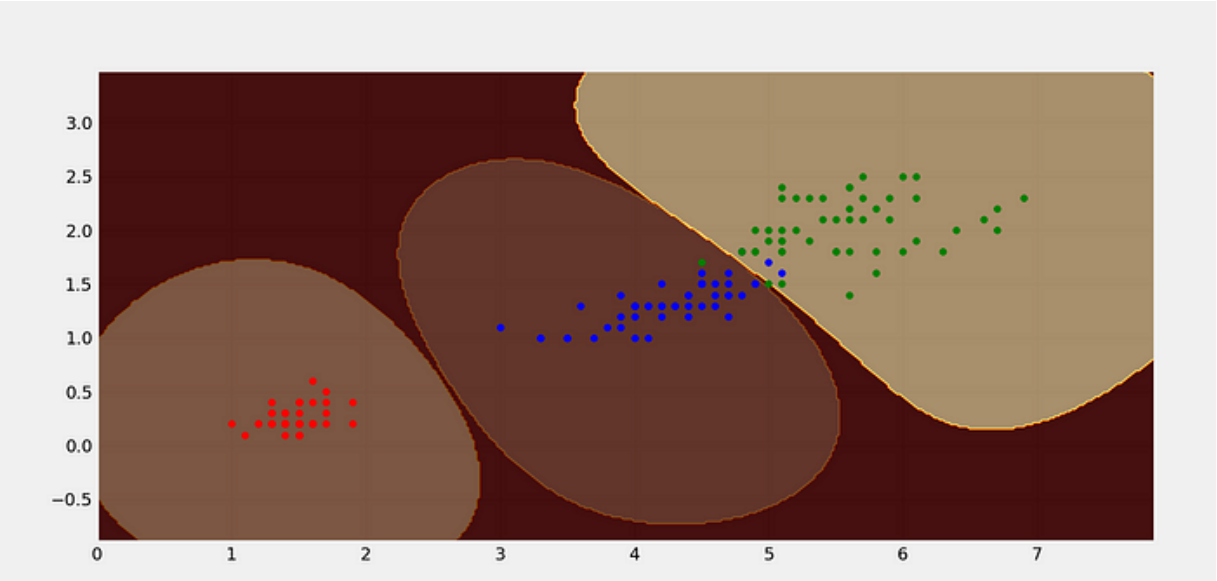
At 3 hyper-plane is drawn.

Copy

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn.svm import SVC

style.use('fivethirtyeight')

# create mesh grids
def make_meshgrid(x, y, h =.02):
```

```
 xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max
 return xx, yy

# plot the contours
def plot_contours(ax, clf, xx, yy, **params):
 Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
 Z = Z.reshape(xx.shape)
 out = ax.contourf(xx, yy, Z, **params)
 return out

color = ['r', 'b', 'g', 'k']

iris = pd.read_csv("iris-data.txt").values


features = iris[0:150, 2:4]
level1 = np.zeros(150)
level2 = np.zeros(150)
level3 = np.zeros(150)

# level1 contains 1 for class1 and 0 for all others.
# level2 contains 1 for class2 and 0 for all others.
# level3 contains 1 for class3 and 0 for all others.
for i in range(150):
 if i>= 0 and i<50:
  level1[i] = 1
 elif i>= 50 and i<100:
  level2[i] = 1
 elif i>= 100 and i<150:
  level3[i]= 1

# create 3 svm with rbf kernels
svm1 = SVC(kernel ='rbf')
svm2 = SVC(kernel ='rbf')
svm3 = SVC(kernel ='rbf')

# fit each svm's
svm1.fit(features, level1)
svm2.fit(features, level2)
svm3.fit(features, level3)
```

```
X0, X1 = iris[:, 2], iris[:, 3]
xx, yy = make_meshgrid(X0, X1)

# plot the contours
plot_contours(ax, svm1, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.8)
plot_contours(ax, svm2, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.3)
plot_contours(ax, svm3, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.5)

color = ['r', 'b', 'g', 'k']

for i in range(len(iris)):
 plt.scatter(iris[i][2], iris[i][3], s = 30, c = color[int(iris[i][4])])
plt.show()
```



Advantages:

- It works really well with a clear margin of separation
- It is effective in high dimensional spaces.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called

Disadvantages:

- It doesn't perform well when we have large data set because the required training time is higher
- It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
- SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is included in the related SVC method of Python scikit-learn library.