SVM implementation

- data=pd.read_csv('iris.csv’)
- d={'Iris-setosa':0,'Iris-versicolor':1, 'Iris-virginica':2}
- data['species']=data['species'].map(d)

- X=data[['sepal_length','sepal_width’]]

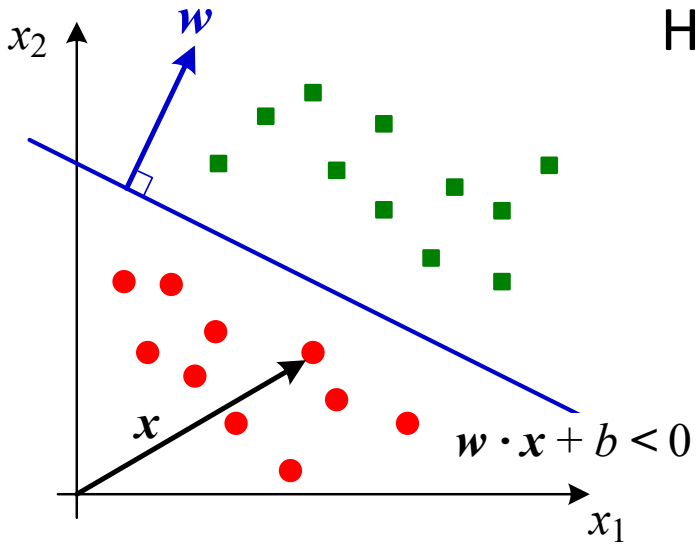
```
svc_model = SVC(kernel='linear)

svc_model.fit(X_train, y_train)
```

Linear Support Vector Machines

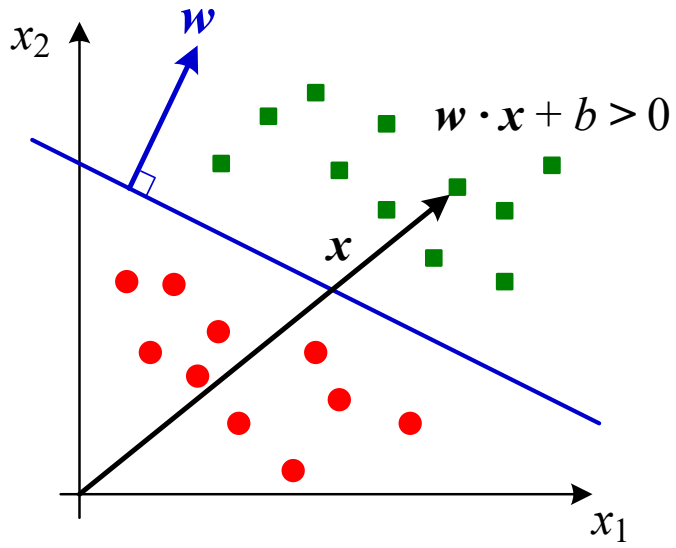
Hyperplane equation: $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$.

If $\boldsymbol{x}$ is the position vector of a negative point then $\boldsymbol{w} \cdot \boldsymbol{x} + b < 0$.

If $\boldsymbol{x}$ is the position vector of a positive point then $\boldsymbol{w} \cdot \boldsymbol{x} + b > 0$.



Hence the terminology.



Equations of the hyperplane

$$[x_1, x_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = 0$$

$$x_1 w_1 + x_2 w_2 + b = 0$$

$$x_2 w_2 = -x_1 w_1 - b$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Slope Intercept

$where,\ x_1$ and x_2 are found $svc\_model.$ coeff_

b is in scv_model.intercept_

w = svc_model.coef_[0] # w consists of 2 elements

b = svc_model.intercept_[0] # b consists of 1 element

x_points = np.linspace(4, 7) # generating x-points from 4 to 7

y_points = -(w[0] / w[1]) * x_points - b / w[1] # getting corresponding y-points

- # Plotting a red hyperplane
  plt.plot(x_points, y_points, c='r’);

- # Encircle support vectors
- **plt.scatter(svc_model.support_vectors_[:, 0],**
  svc_model.support_vectors_[:, 1], s=50,
  facecolors='none’,
  edgecolors='k’,
  alpha=.5);

The hyperplane classifier

Recall that the hyperplane classifier has equation

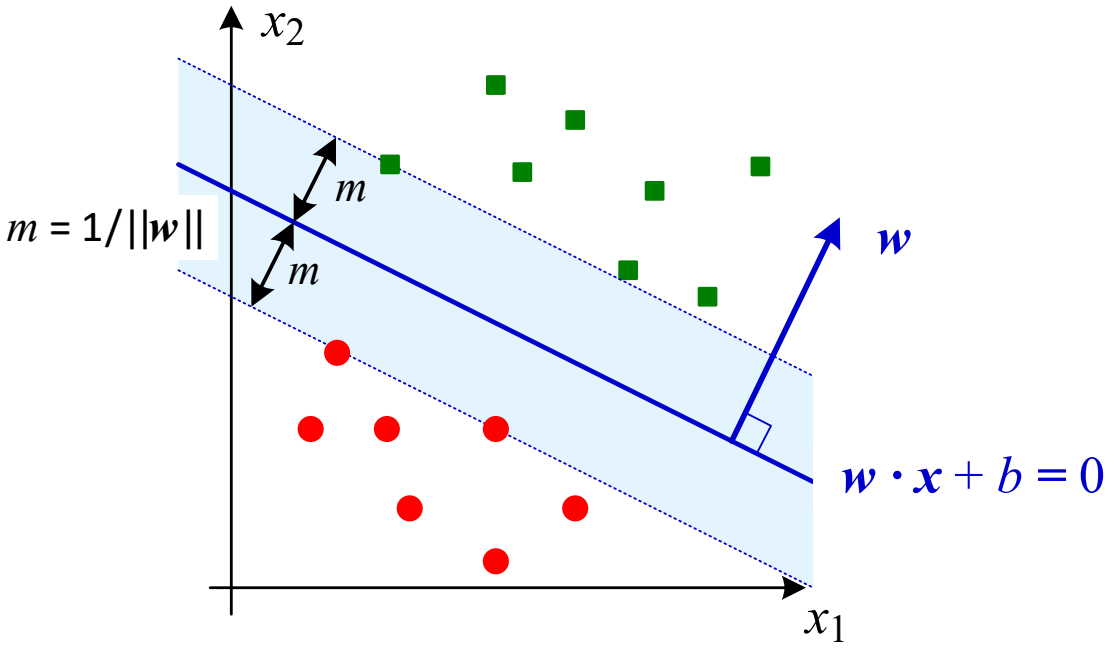
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0.$$

The width of the margin is

$$d = \frac{2}{\|\boldsymbol{w}\|}.$$

Hence the distance from the hyperplane classifier to the positive and negative barriers is

$$m = \frac{1}{\|\boldsymbol{w}\|}.$$



The hyperplane classifier

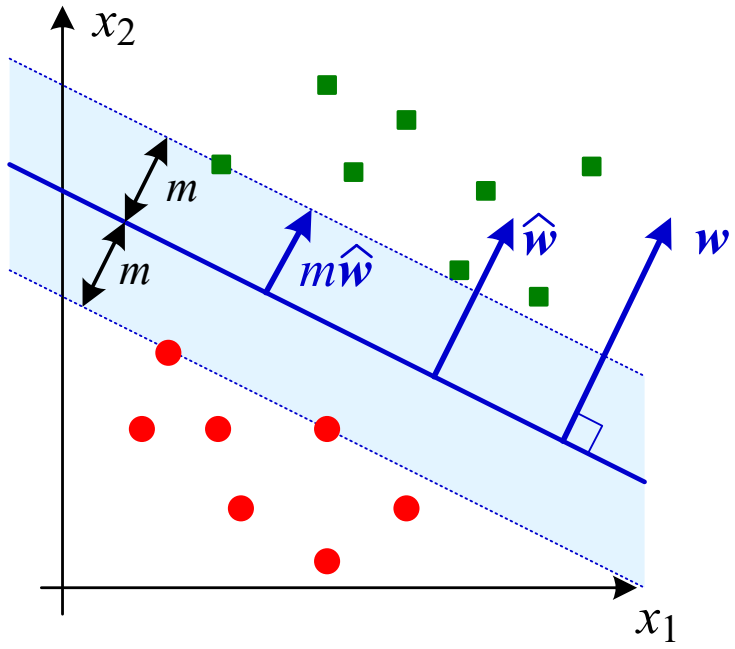
The vector $\boldsymbol{w}$ is perpendicular to the hyperplane classifier.

Then

$$\hat{\boldsymbol{w}} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

is a unit vector (length = 1) in the direction of $\boldsymbol{w}$.

Hence $m\hat{\boldsymbol{w}}$ has length m so is a vector from the hyperplane classifier to the positive barrier hyperplane.

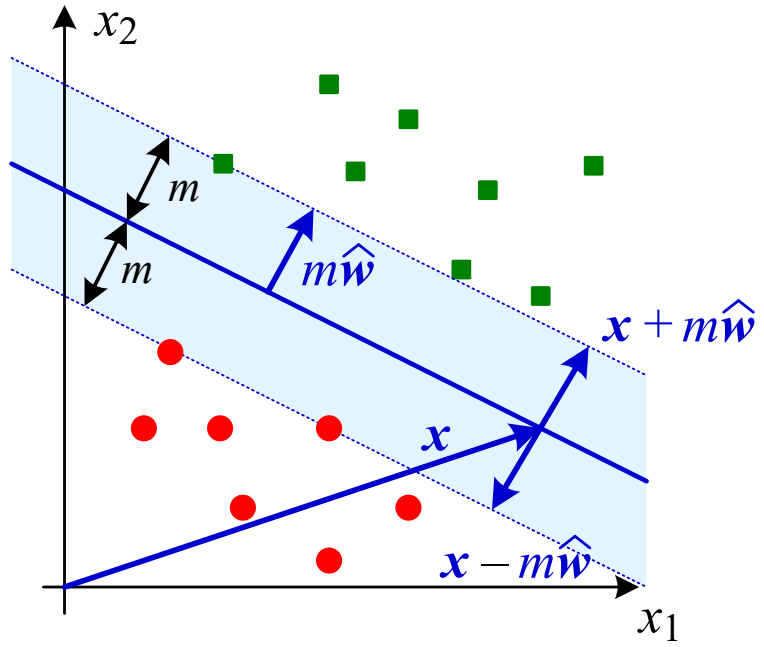


The hyperplane classifier

Let $\boldsymbol{x}$ be the position vector of a point on the hyperplane classifier.

Then $\boldsymbol{x} + m\hat{\boldsymbol{w}}$ is the position vector of a point on the positive barrier hyperplane.

And $\boldsymbol{x} - m\hat{\boldsymbol{w}}$ is the position vector of a point on the negative barrier hyperplane.



```
w_hat = svc_model.coef_[0] / (np.sqrt(np.sum(svc_model.coef_[0] ** 2)))

margin = 1 / np.sqrt(np.sum(svc_model.coef_[0] ** 2))

decision_boundary_points = np.array(list(zip(x_points, y_points)))

points_of_line_above = decision_boundary_points + w_hat * margin
points_of_line_below = decision_boundary_points - w_hat * margin
```

```
plt.plot(points_of_line_above[:, 0],
points_of_line_above[:, 1], 'b--', linewidth=2)
```

```
plt.plot(points_of_line_below[:, 0], points_of_line_below[:, 1], 'g--',
linewidth=2)
```

Output