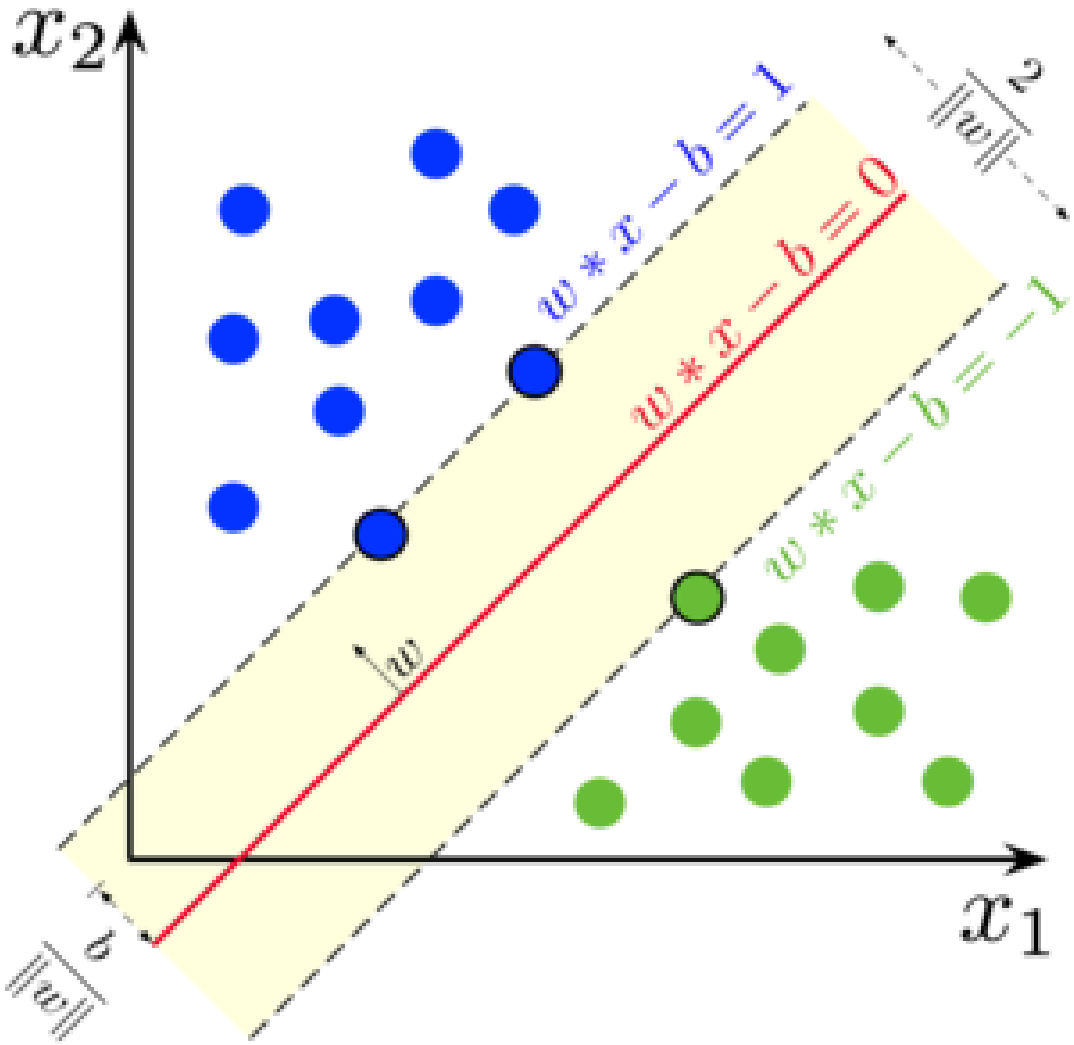
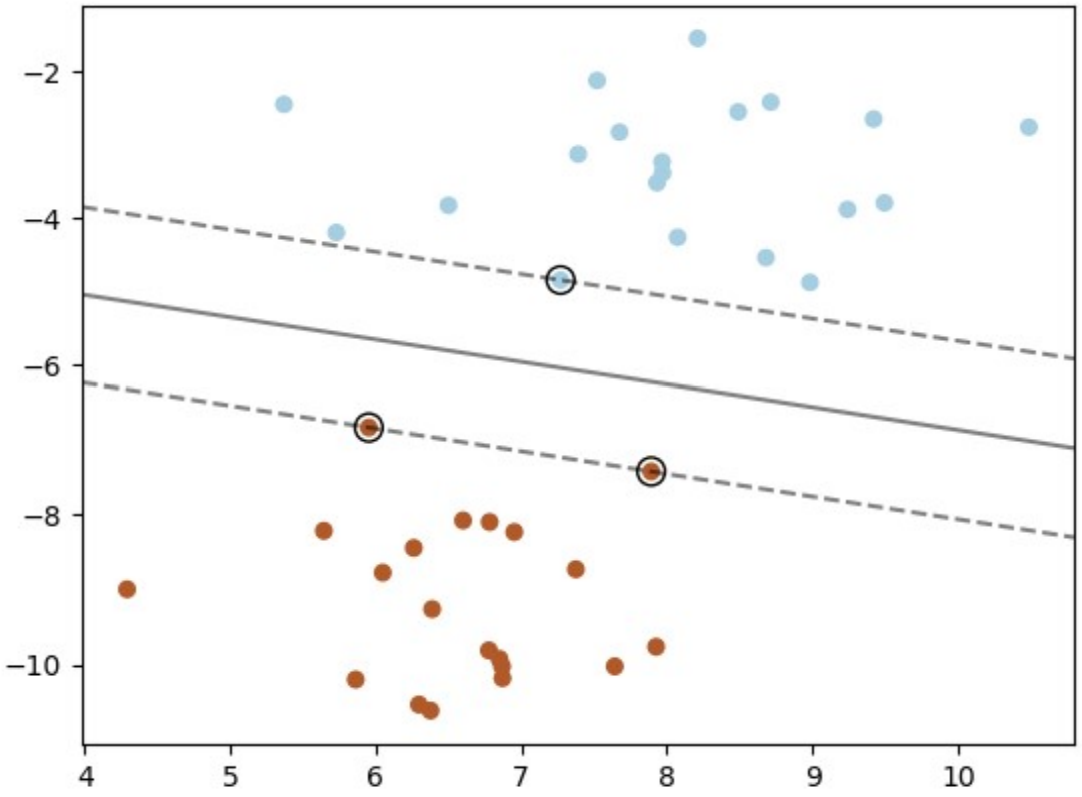
Support Vector Machine

This PPT uses various online resources and is not my original work

SVM

- SVM is **trained** with the labeled data.
- It is a binary classifier.
- It learns a **linear model**.
- What is a linear model? In simple words: it is a line (in complicated words it is a hyperplane). If data is very simple and only has two dimensions, then the SVM will learn a line which will be able to separate the data.


x_2
x_1
$w * x - b = 1$
$w * x - b = 0$
$w * x - b = -1$
w
$\frac{2}{\|w\|}$
$\frac{b}{\|w\|}$


−2
−4
−6
−8
−10
4
5
6
7
8
9
10

1) We suppose that the data we want to classify can be separated by a line
2) We know that a line can be represented by the equation y=**wx**+b (this is our model)
3) We know that there is an infinity of possible lines obtained by changing the value of **w** and b
4) We use an algorithm to determine which are the values of **w** and b giving the "best" line separating the data.

How can we find the biggest margin ?

- have a dataset
- select two hyperplanes which separate the data with no points between them
- maximize their distance (the margin)
- The region bounded by the two hyperplanes will be the biggest possible margin.

- *Any hyperplane can be written as the set of points* x *satisfying* $w \cdot x + b = 0$

Given a hyperplane H_0 separating the dataset and satisfying:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

We can select two others hyperplanes H_1 and H_2 which also separate the data and have the following equations :

$$\mathbf{w} \cdot \mathbf{x} + b = \delta$$

and

$$\mathbf{w} \cdot \mathbf{x} + b = -\delta$$

so that H_0 is equidistant from H_1 and H_2.

However, here the variable δ is not necessary. So we can set $\delta = 1$ to simplify the problem.

$$\mathbf{w} \cdot \mathbf{x} + b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

Now we want to be sure that they have no points between them.

We won't select *any* hyperplane, we will only select those who meet the two following **constraints**:

For each vector $\mathbf{x_i}$ either :

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 \text{ for } \mathbf{x_i} \text{ having the class } 1 \tag{4}$$

or

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ for } \mathbf{x_i} \text{ having the class } -1 \tag{5}$$

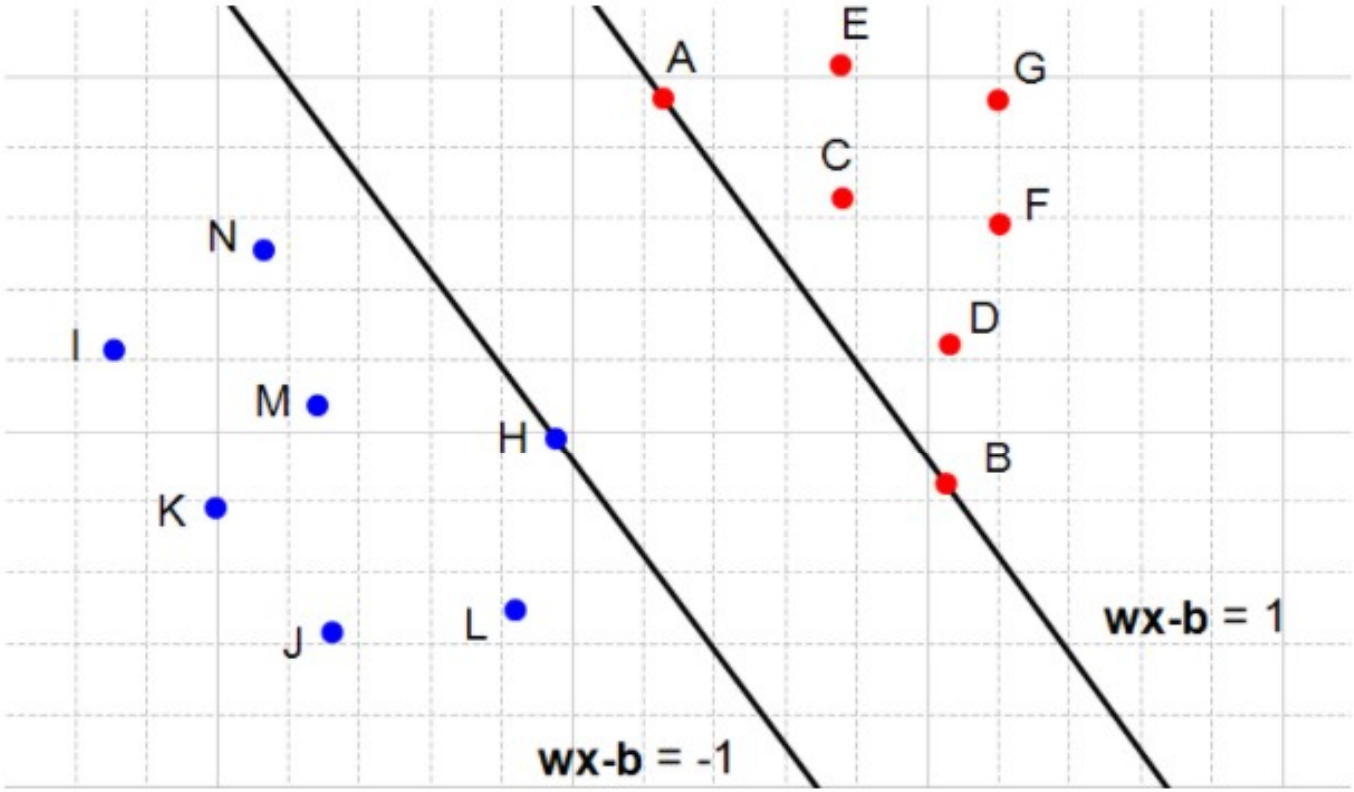


Figure 4: Two hyperplanes satisfying the constraints

On *Figure 5*, we see another couple of hyperplanes respecting the constraints:

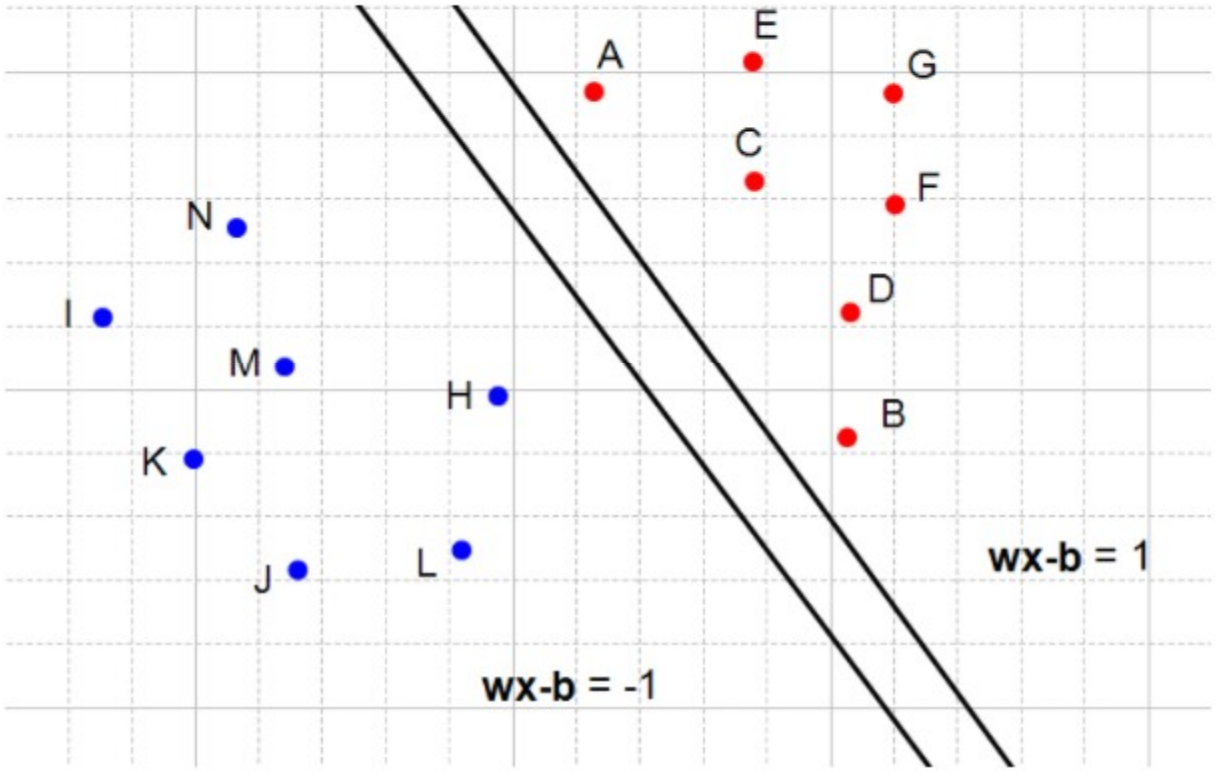


Figure 5: Two hyperplanes also satisfying the constraints

And now we will examine cases where the constraints are not respected:

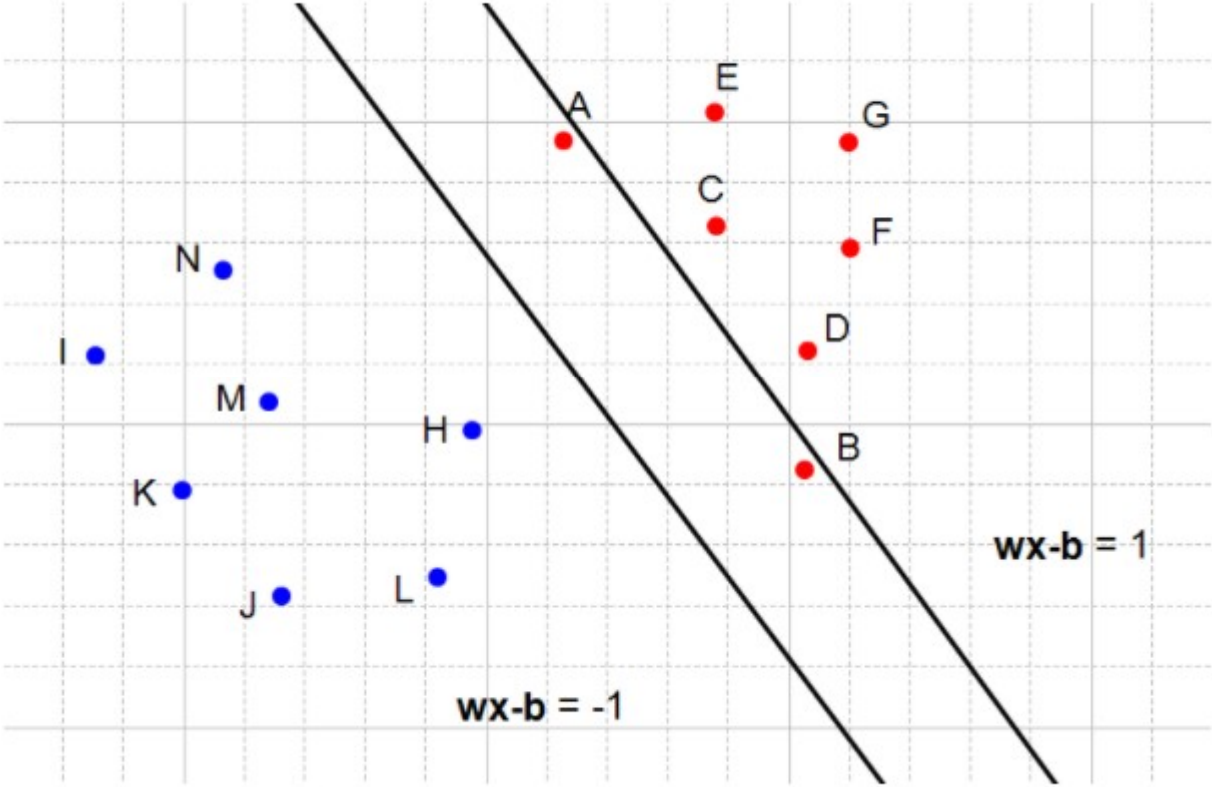


Figure 6: The right hyperplane does not satisfy the first constraint

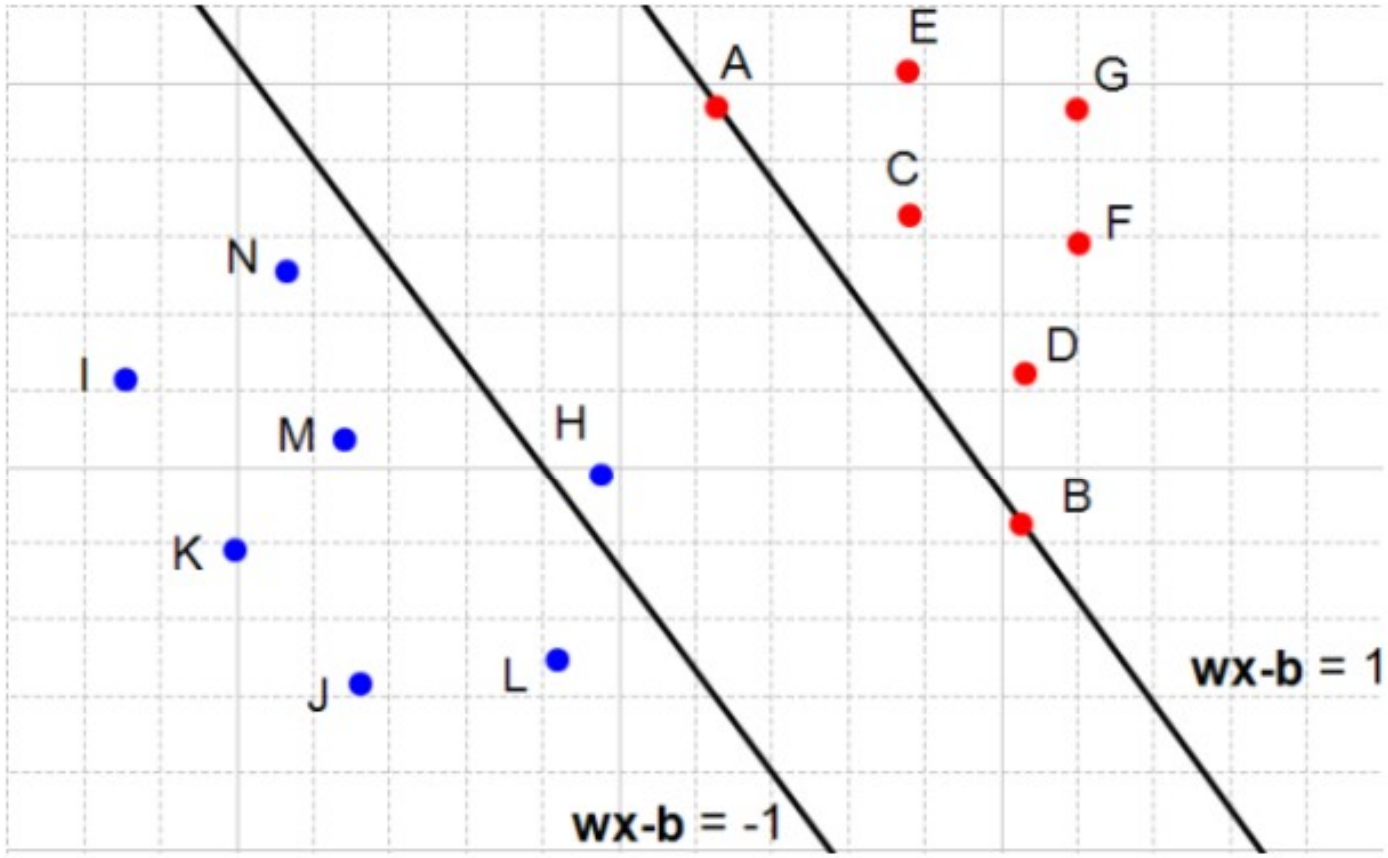


Figure 7: The left hyperplane does not satisfy the second constraint

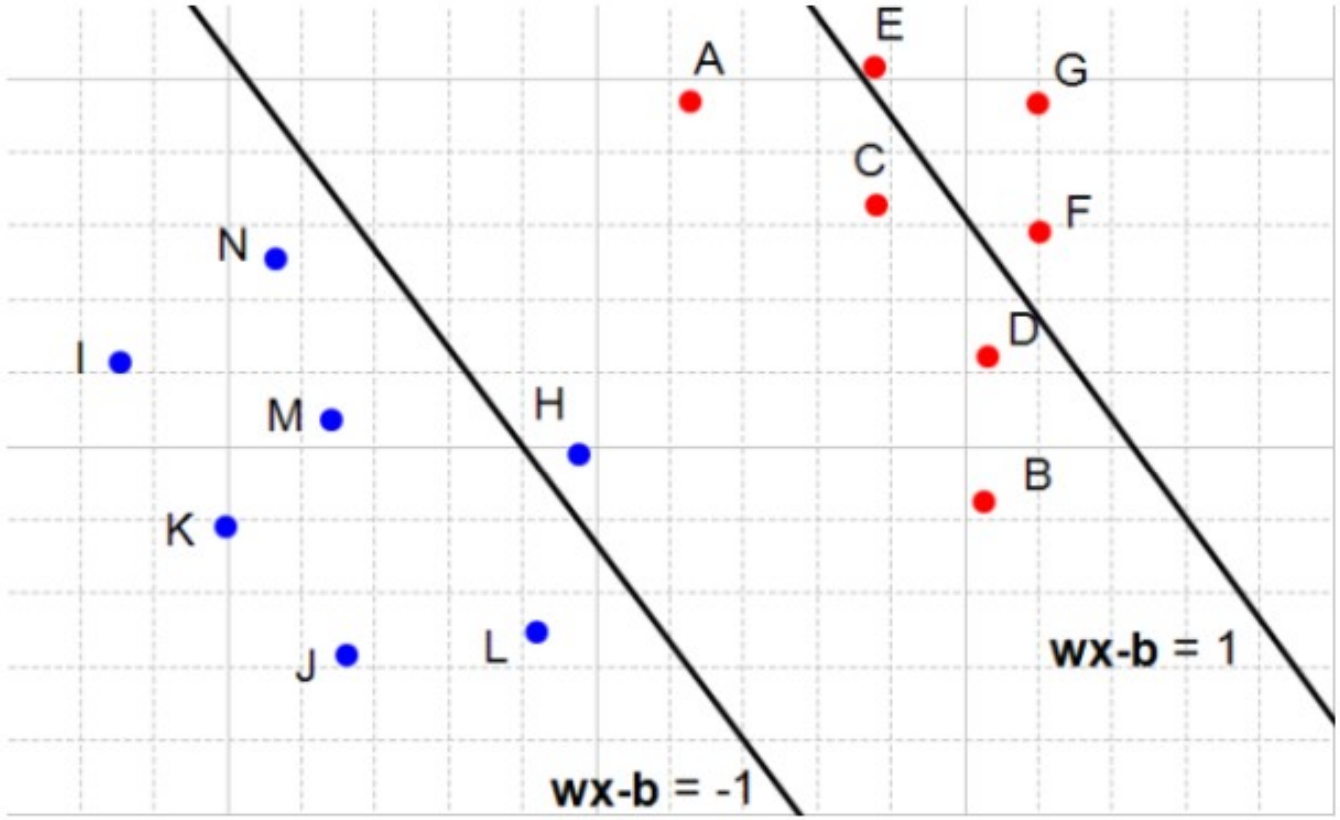


Figure 8: Both constraint are not satisfied

Equations **(4)** and **(5)** can be combined into a single constraint:

We start with equation **(5)**

$$\text{for } \mathbf{x_i} \text{ having the class } -1$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1$$

And multiply both sides by y_i (which is always -1 in this equation)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq y_i(-1)$$

Which means equation **(5)** can also be written:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \text{ for } \mathbf{x_i} \text{ having the class } -1 \tag{6}$$

In equation **(4)**, as $y_i = 1$ it doesn't change the sign of the inequation.

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \text{ for } \mathbf{x_i} \text{ having the class } 1 \tag{7}$$

We combine equations **(6)** and **(7)** :

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \text{ for all } 1 \leq i \leq n \tag{8}$$

We now have a unique constraint (equation **8**) instead of two (equations **4** and **5**) , but they are mathematically equivalent. So their effect is the same (there will be no points between the two hyperplanes).

Step 3: Maximize the distance between the two hyperplanes

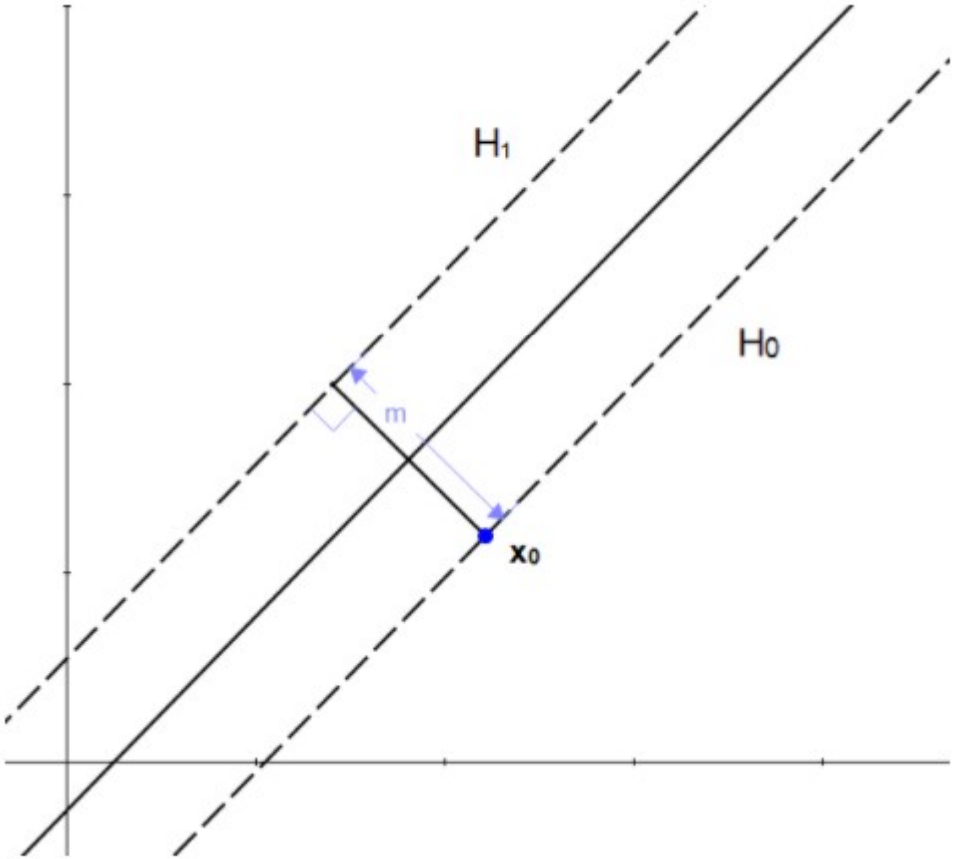


Figure 9: m is the distance between the two hyperplanes

- Maximizing the margin is the same thing as minimizing the norm of w
- Our goal is to maximize the margin. Among all possible hyperplanes meeting the constraints, we will choose the hyperplane with the smallest $\|w\|$ because it is the one which will have the biggest margin.

This give us the following optimization problem:

$$\text{Minimize in } (\mathbf{w}, b)$$

$$\|\mathbf{w}\|$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1$$

$$\text{(for any } i = 1, \ldots, n)$$

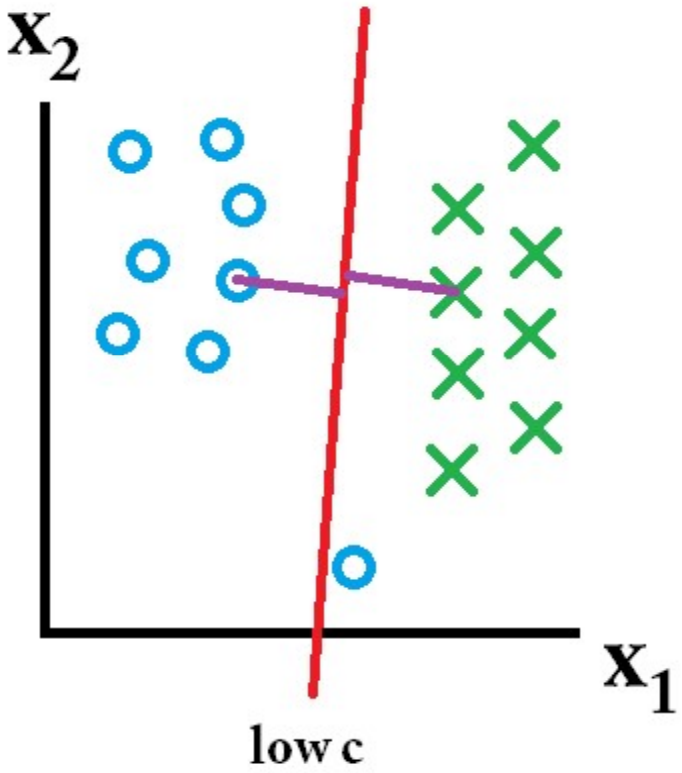
Solving this problem is like solving and equation. Once we have solved it, we will have found the couple $(\mathbf{w}, b)$ for which $\|\mathbf{w}\|$ is the smallest possible and the constraints we fixed are met. Which means we will have the equation of the optimal hyperplane !

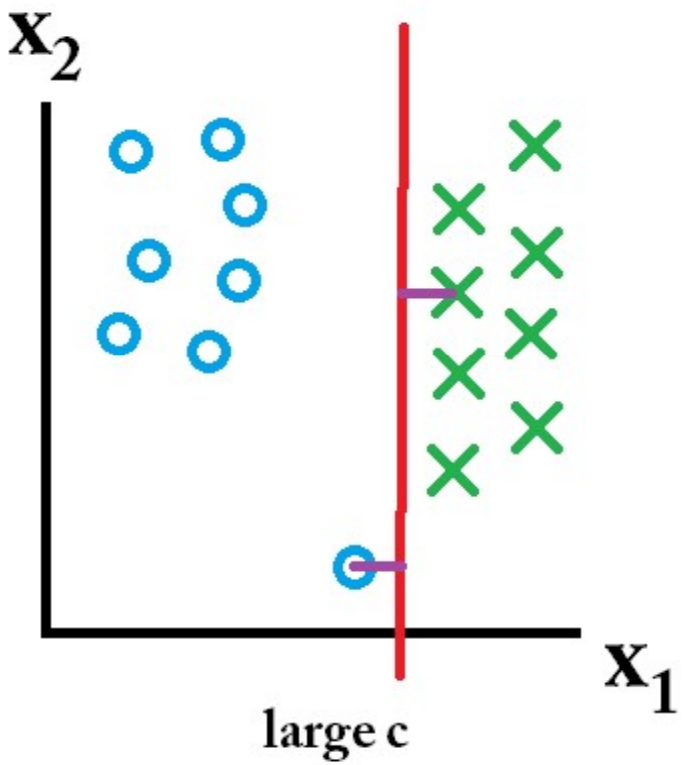
Vanilla SVM

Regularization parameter C

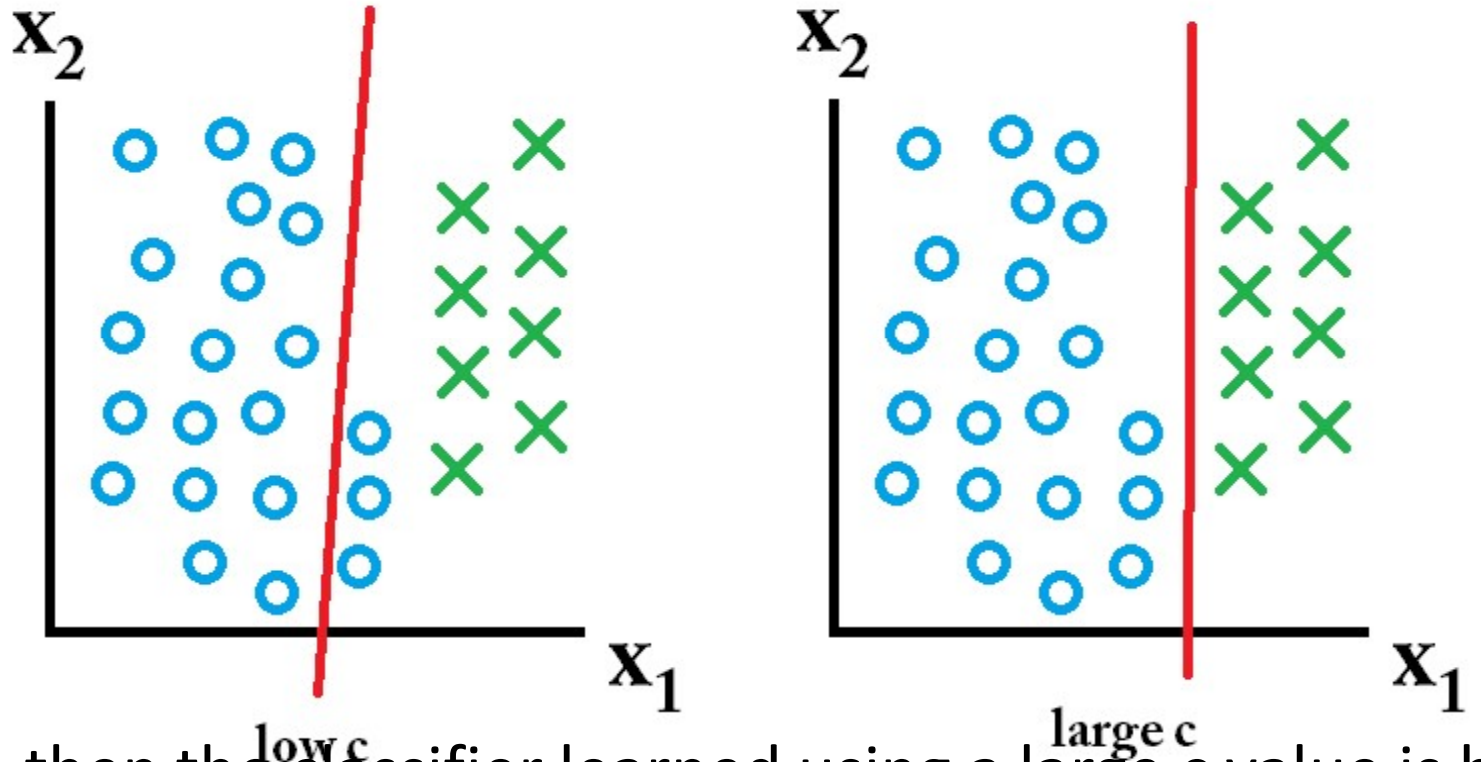
- The C parameter trades off correct classification of training examples against maximization of the decision function's margin.
- For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly.
- A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy.

- The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.


x_2
x_1
low c


x_2
x_1
large c

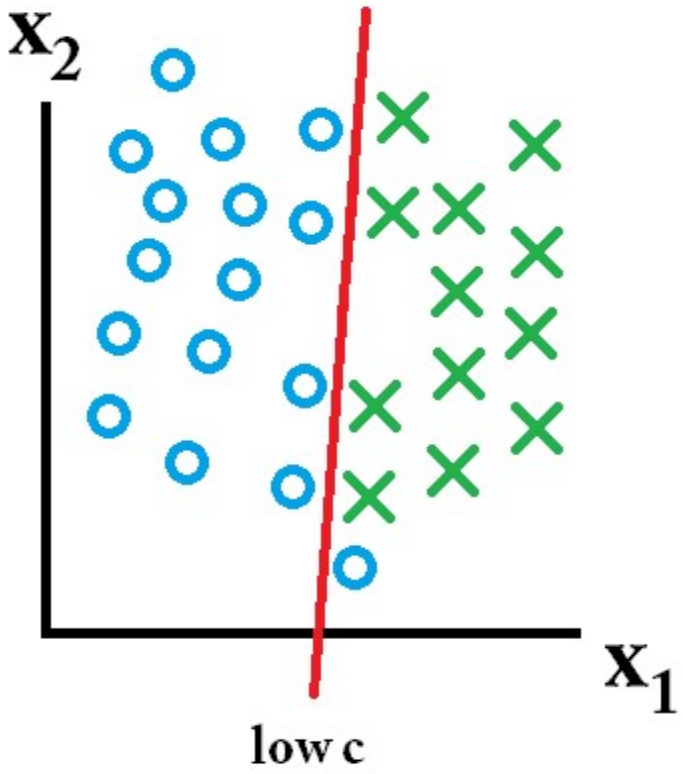
So which of these classifiers are the best? That depends on what the future data you will predict looks like, and most often you don't know that of course. If the future data looks like this:

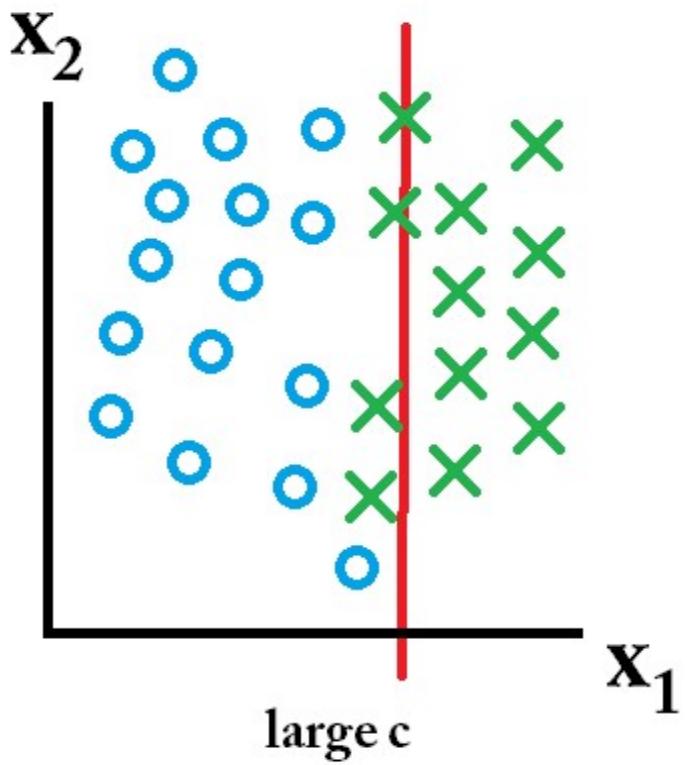


- then the classifier learned using a large c value is best.

On the other hand, if the future data looks like this:

- the





Nonlinearly separable data: use kernels

- Different Kernel Functions
  - linear, nonlinear, polynomial, Gaussian kernel, Radial basis function (RBF), sigmoid etc

Polynomial kernel: degree *d*

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

Gaussian kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Gaussian Radial Basis Function (RBF)

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

$$\gamma > 0$$

$$\gamma = 1/2\sigma^2$$

Gamma

- Gamma is a parameter of the RBF kernel and can be thought of as the ‘spread’ of the kernel and therefore the decision region.
- When gamma is low, the ‘curve’ of the decision boundary is very low and thus the decision region is very broad.
- When gamma is high, the ‘curve’ of the decision boundary is high, which creates islands of decision-boundaries around data points.

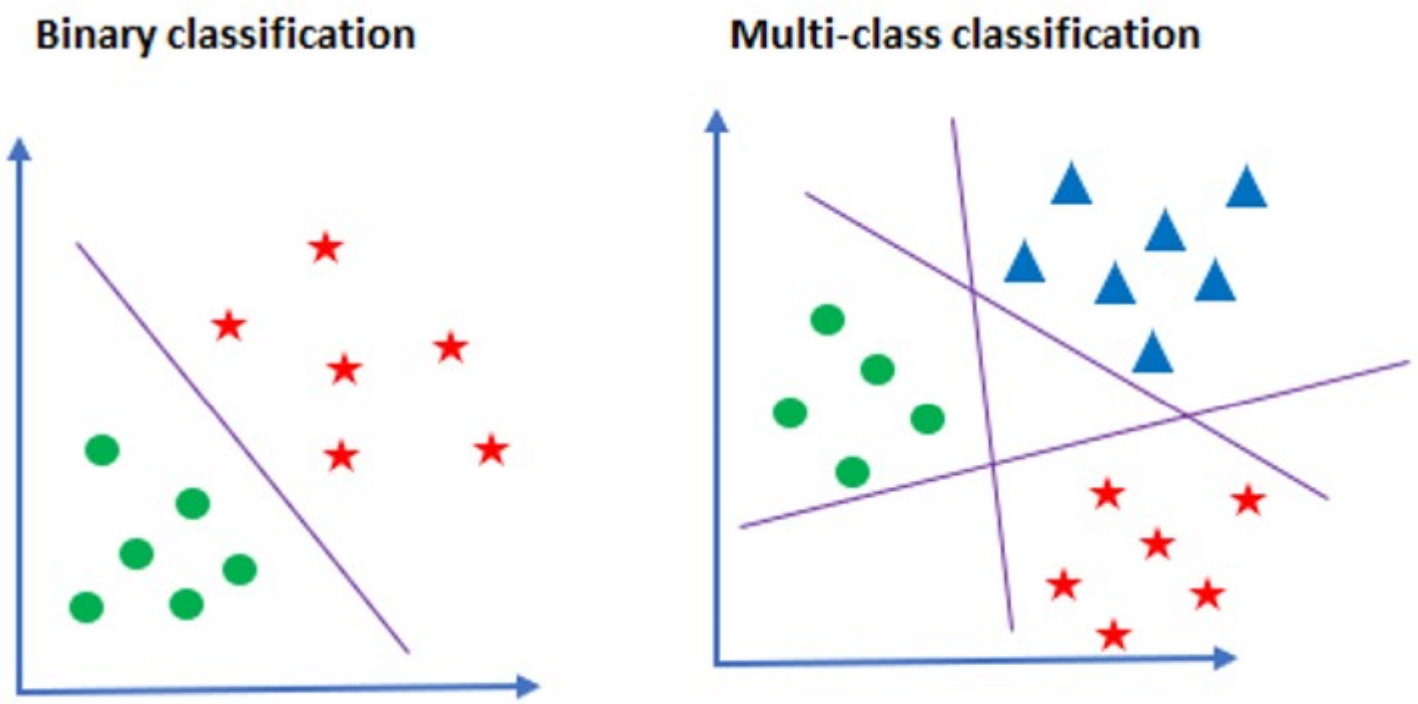
C

- C is a parameter of the SVC learner and is the penalty for misclassifying a data point.
- When C is small, the classifier is okay with misclassified data points (high bias, low variance).
- When C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards avoid any misclassified data points (low bias, high variance).

Multiple classes

- OVO (Libsvm – SVC)
- OVR (LinearSVC)

OVR



One-vs-One (OvO)

- Hereby the number of generated models depending on the number of classes where N is the number of classes.
- $N=N(N-1)/2$
- If N is 10 as shown in our example below the total of the learned model is 45 according to the mentioned formula.
- In this method, every single class will be paired one by one with other class.
- At the end of the classification training, each classification is given one vote for the winning class.
- The highest votes will determine which class the test dataset belongs to.

One-vs-Rest (OvR)

- One-vs-Rest produced the same amount of learned models with the number of classes.
- If number of classes is10, the number of learned models is also 10.
- In this method, every class is paired with the remaining classes.
- Run N binary classifiers.

- from sklearn.multiclass import OneVsRestClassifier
- from sklearn.multiclass import OneVsOneClassifier

Reference

- https://www.svm-tutorial.com/2017/02/svms-overview-support-vector-machines/