Support vector Machine

SVM-Intuition

Outline

- Math behind SVM
- What is the equation of the hyperplane?
- How to compute the margin?
- SVM : Convex optimization problem
- SVM dual Problem
- Soft margin
- Kernel trick

SVM

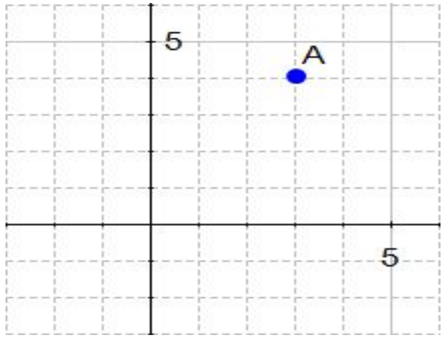
- A supervised machine learning technique
- classification or regression, however, it is most commonly employed for the classification problem.
- **four different Support Vector Machines**:
  - The original one : the Maximal Margin Classifier,
  - The kernelized version using the Kernel Trick,
  - The soft-margin version,
  - The soft-margin kernelized version (which combine 1, 2 and 3)
-

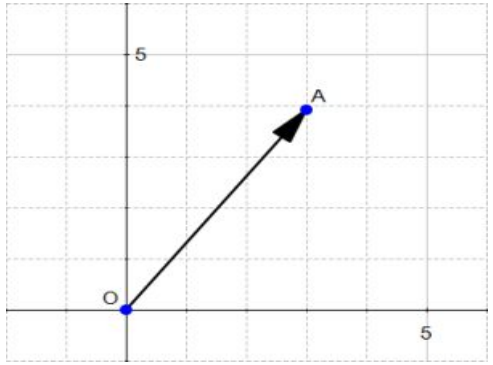
Linear Algebra fundamentals

- Vector
  - What is a vector?
    - its norm
    - its direction
  - How to add and subtract vectors ?
  - What is the dot product ?
  - How to project a vector onto another ?

Vector in R^2

A(3,4)





Any point $x=(x_1,x_2), x\neq 0$ x=(x1,x2),x≠0*, in* R^2 *specifies a vector in the plane, namely the vector starting at the origin and ending at x.*

Vector in R^2

Definition: A vector is an object that has both a magnitude and a direction.

The magnitude or length of a vector x is written $\|x\|$ and is called its norm.

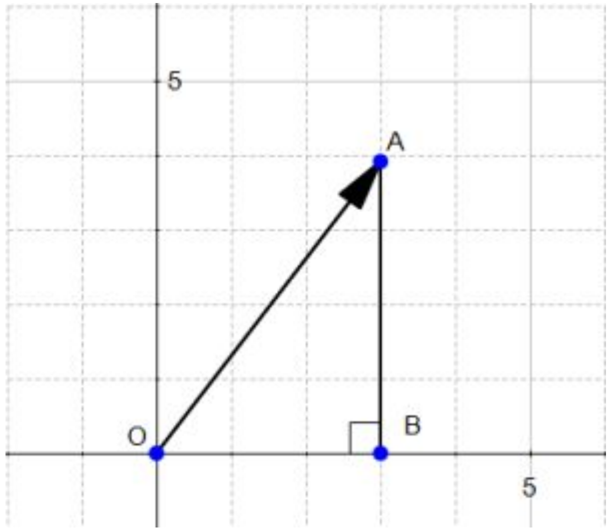
$$OA^2 = OB^2 + AB^2$$

$$OA^2 = 3^2 + 4^2$$

$$OA^2 = 25$$

$$OA = \sqrt{25}$$

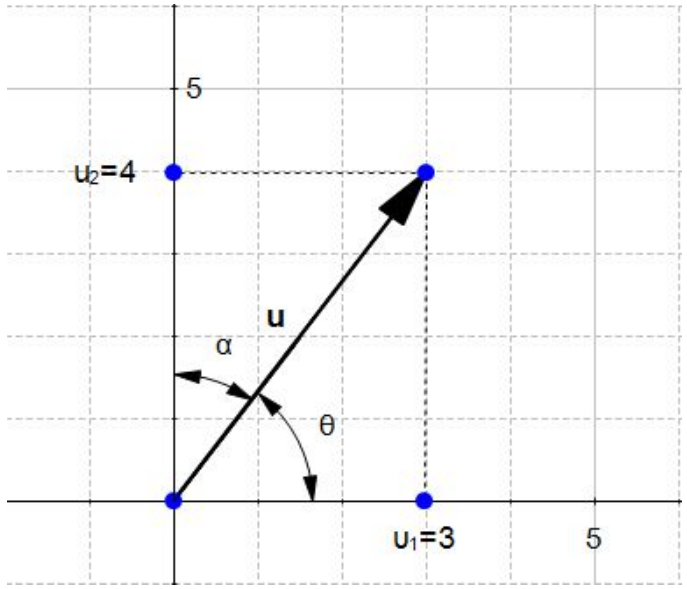
$$\|OA\| = OA = 5$$

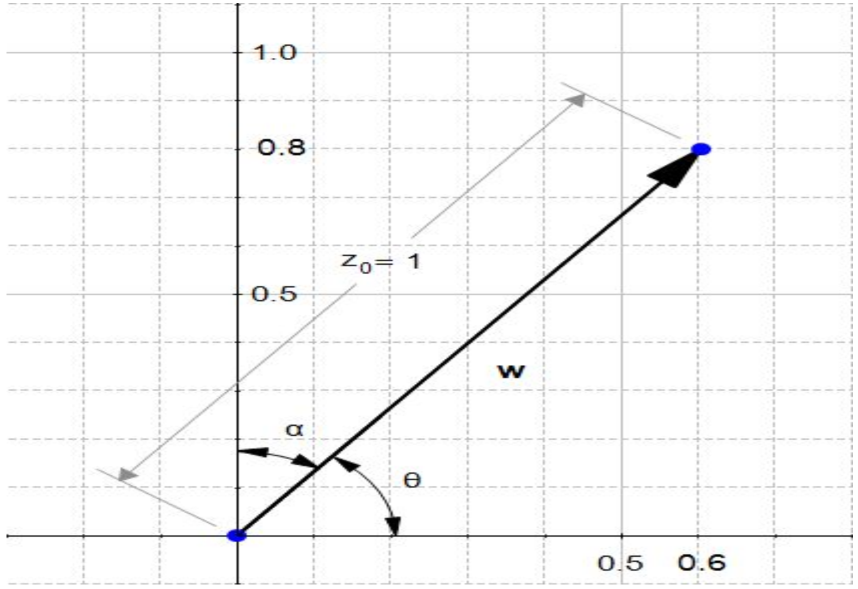


The direction

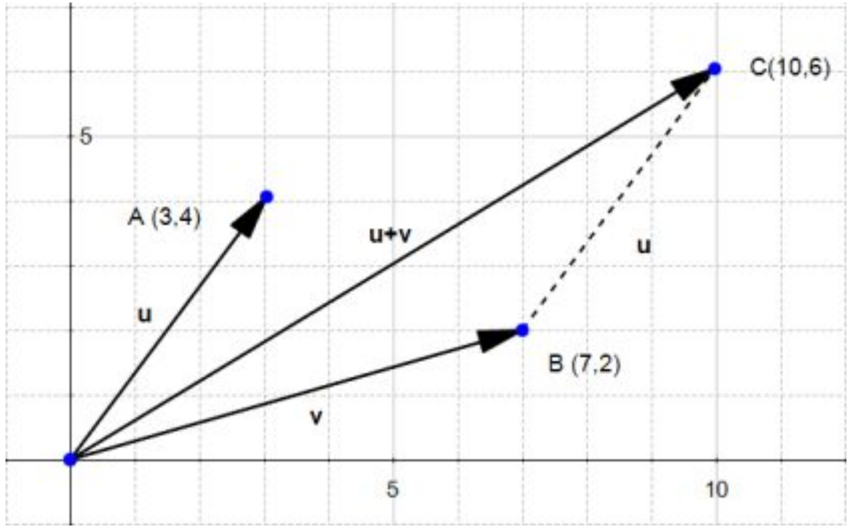
The direction is the second component of a vector.

Definition : The ***direction*** *of a vector* $\mathbf{u}(u_1, u_2)$ *is the vector* $\mathbf{w}(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|})$

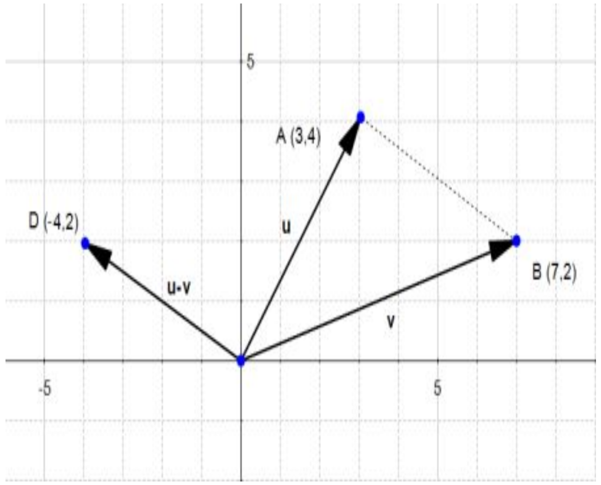




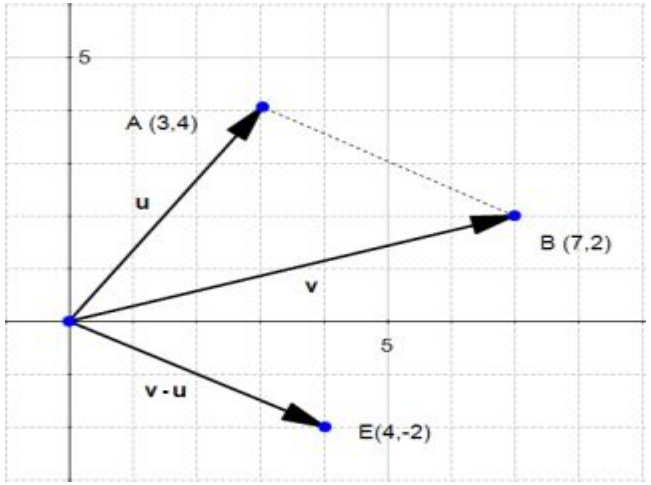
Sum of two vectors



The difference between two vectors : Not commutative



$$\mathbf{u} - \mathbf{v} = (u_1 - v_1, u_2 - v_2)$$



$$\mathbf{v} - \mathbf{u} = (v_1 - u_1, v_2 - u_2)$$

The dot product

)

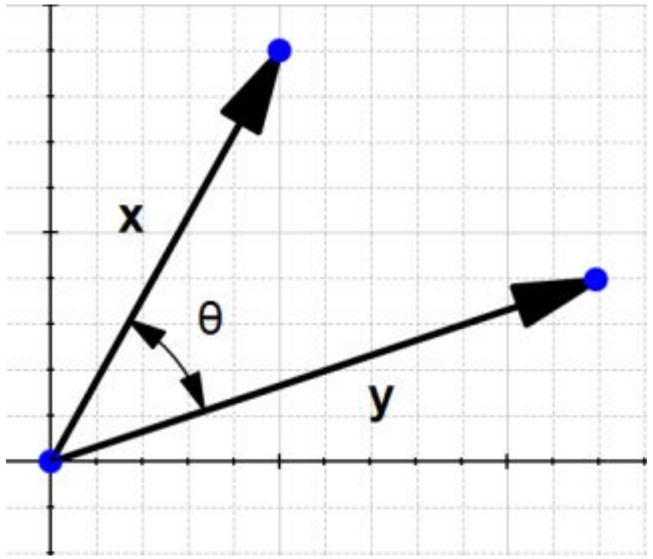
Definition: Geometrically, it is the product of the Euclidian magnitudes of the two vectors and the cosine of the angle between them

x · y= // x // // y // cos(θ)

$$\|x\|\|y\|cos(\theta) = x_1y_1 + x_2y_2$$

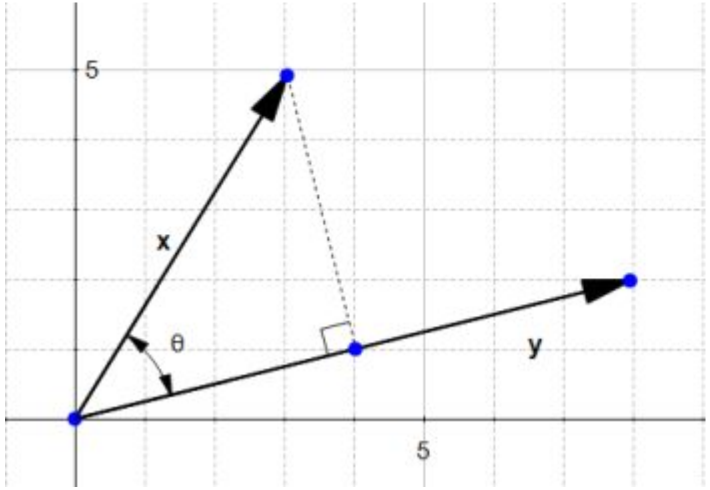
Which is the same as :

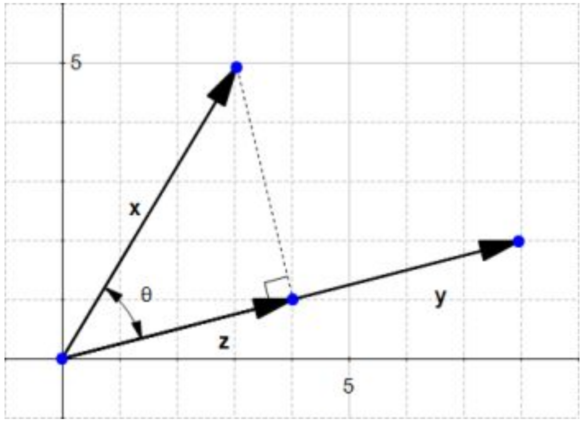
$$\|x\|\|y\|cos(\theta) = \mathbf{x} \cdot \mathbf{y}$$



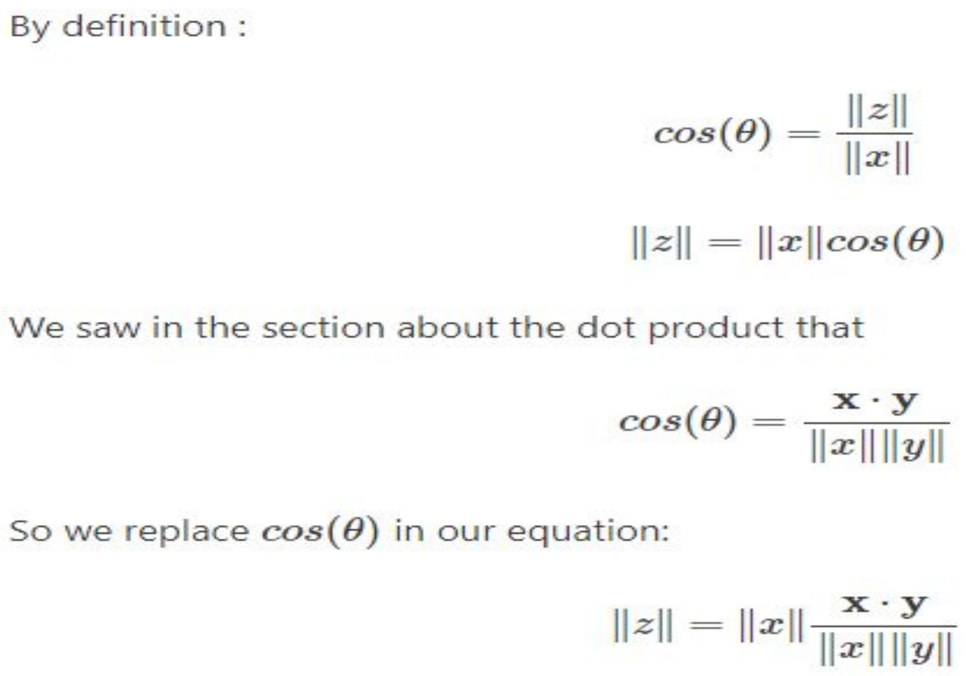
The orthogonal projection of a vector

project the vector x onto y





The orthogonal projection of a vector



By definition :

$$cos(\theta) = \frac{||z||}{||x||}$$

$$||z|| = ||x||cos(\theta)$$

We saw in the section about the dot product that

$$cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{||x||||y||}$$

So we replace $cos(\theta)$ in our equation:

$$||z|| = ||x||\frac{\mathbf{x} \cdot \mathbf{y}}{||x||||y||}$$

If we define the vector $\mathbf{u}$ as the **direction** of $\mathbf{y}$ then

$$\mathbf{u} = \frac{\mathbf{y}}{\|y\|}$$

and

$$\|z\| = \mathbf{u} \cdot \mathbf{x}$$

We now have a simple way to compute the norm of the vector $\mathbf{z}$.
Since this vector is in the same direction as $\mathbf{y}$ it has the direction $\mathbf{u}$

$$\mathbf{u} = \frac{\mathbf{z}}{\|z\|}$$

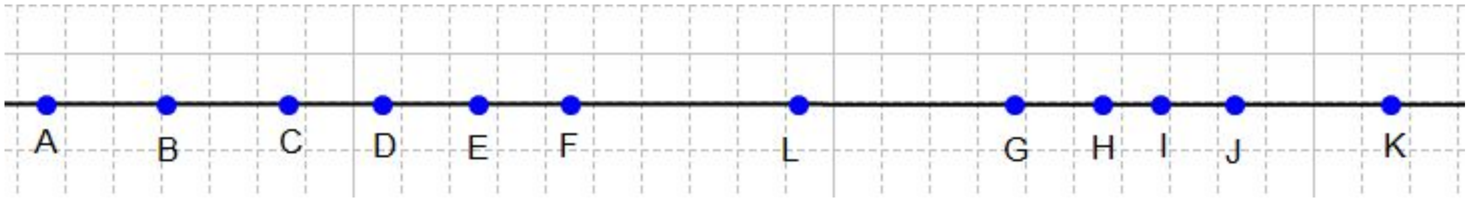
$$\mathbf{z} = \|z\|\mathbf{u}$$

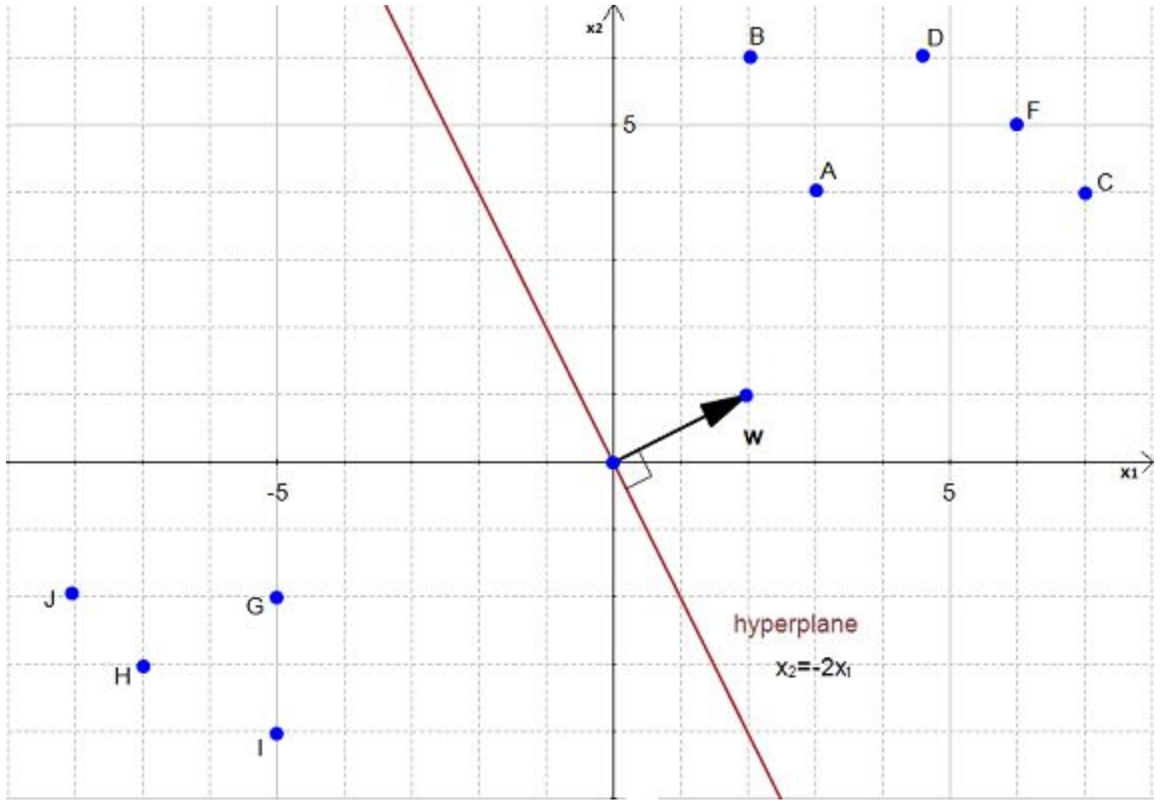
And we can say :

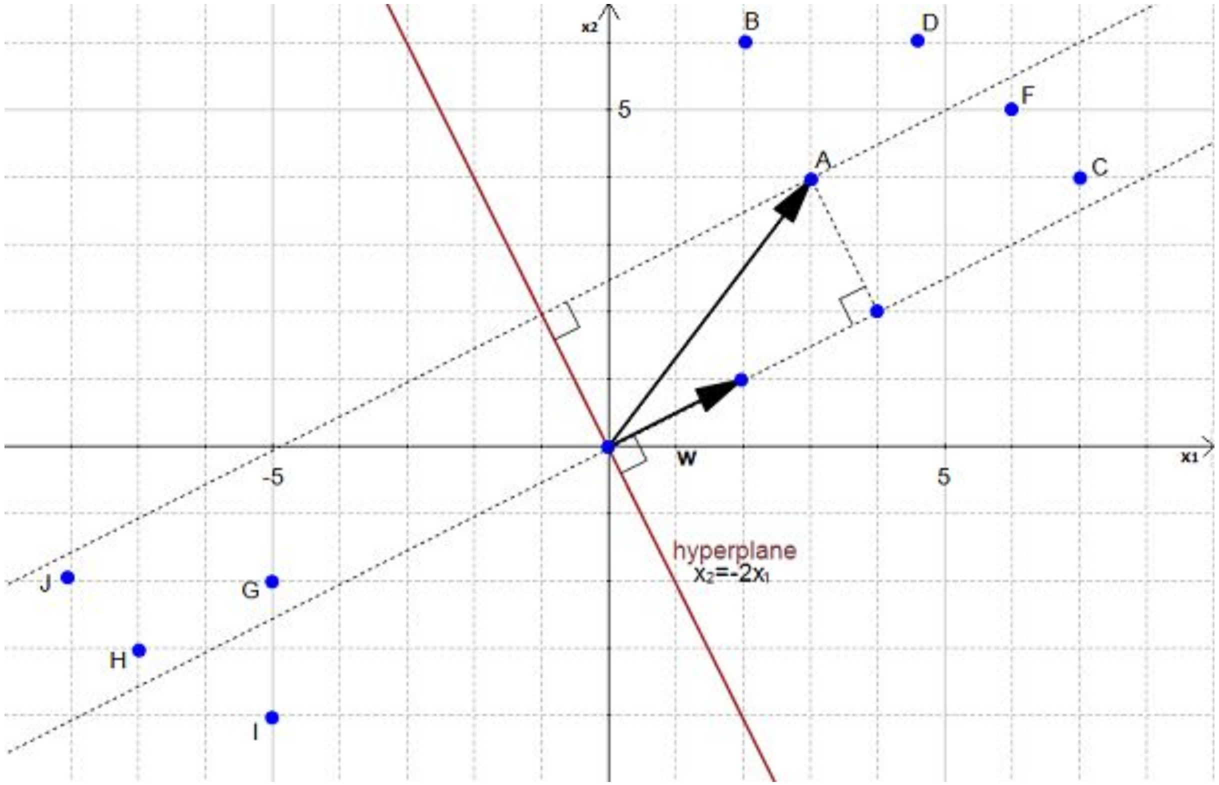
> *The vector $\mathbf{z} = (\mathbf{u} \cdot \mathbf{x})\mathbf{u}$ is the orthogonal projection of $\mathbf{x}$ onto $\mathbf{y}$.*

A hyperplane is a generalization of a plane.

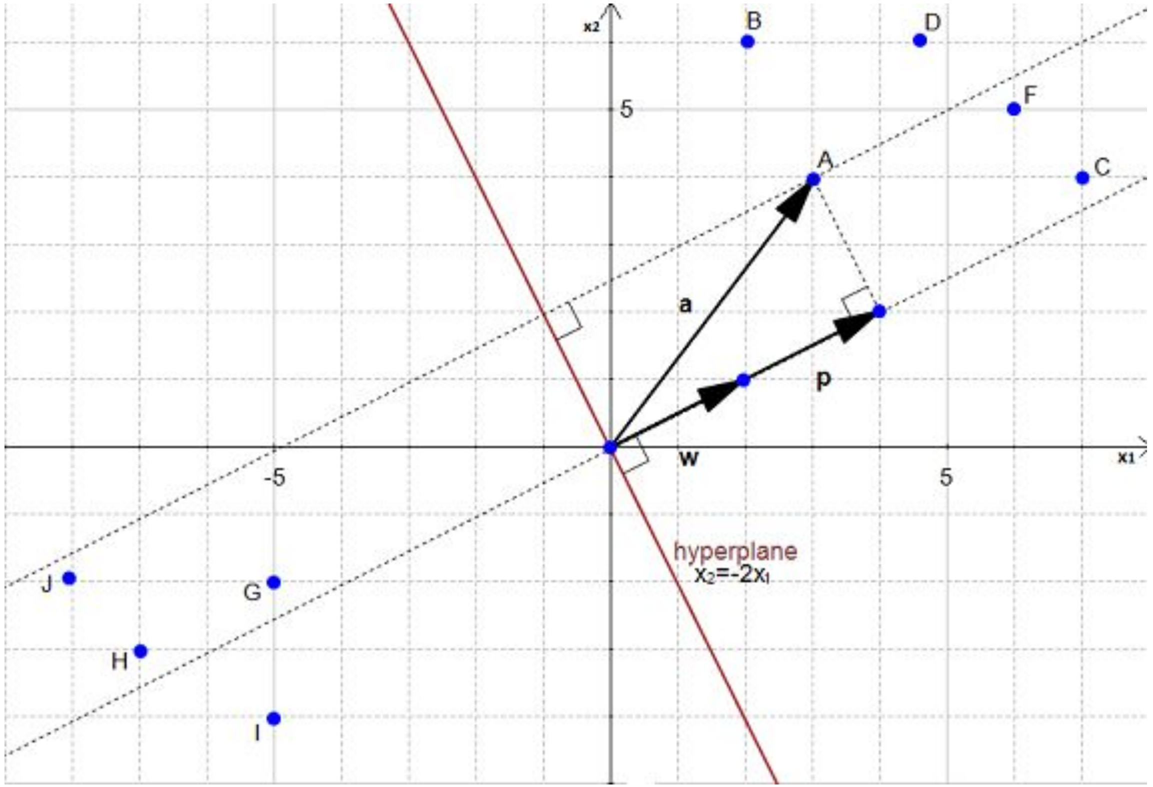
- in one dimension, a hyperplane is called a point
- in two dimensions, it is a line
- in three dimensions, it is a plane
- in more dimensions you can call it an hyperplane
-

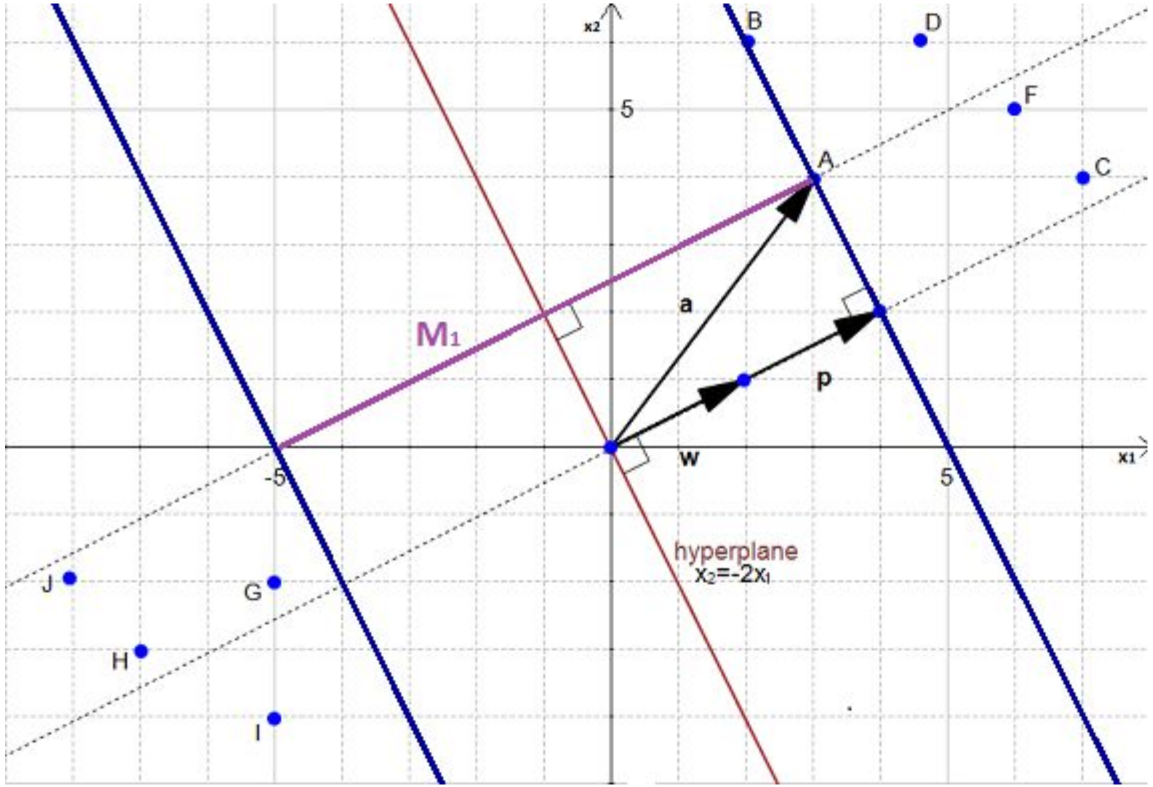



x2
x1
5
-5
5
B
D
F
A
C
w
J
G
H
I
hyperplane
$x_2=-2x_1$



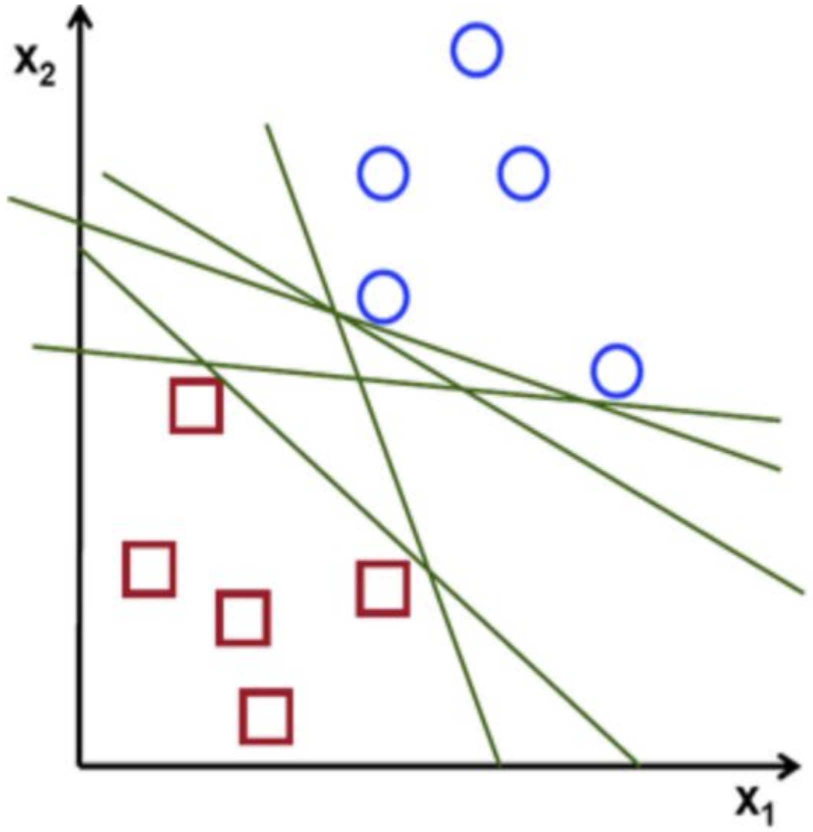
x2
x1
B
D
F
A
C
5
-5
5
W
hyperplane
x₂=-2x₁
J
G
H
I

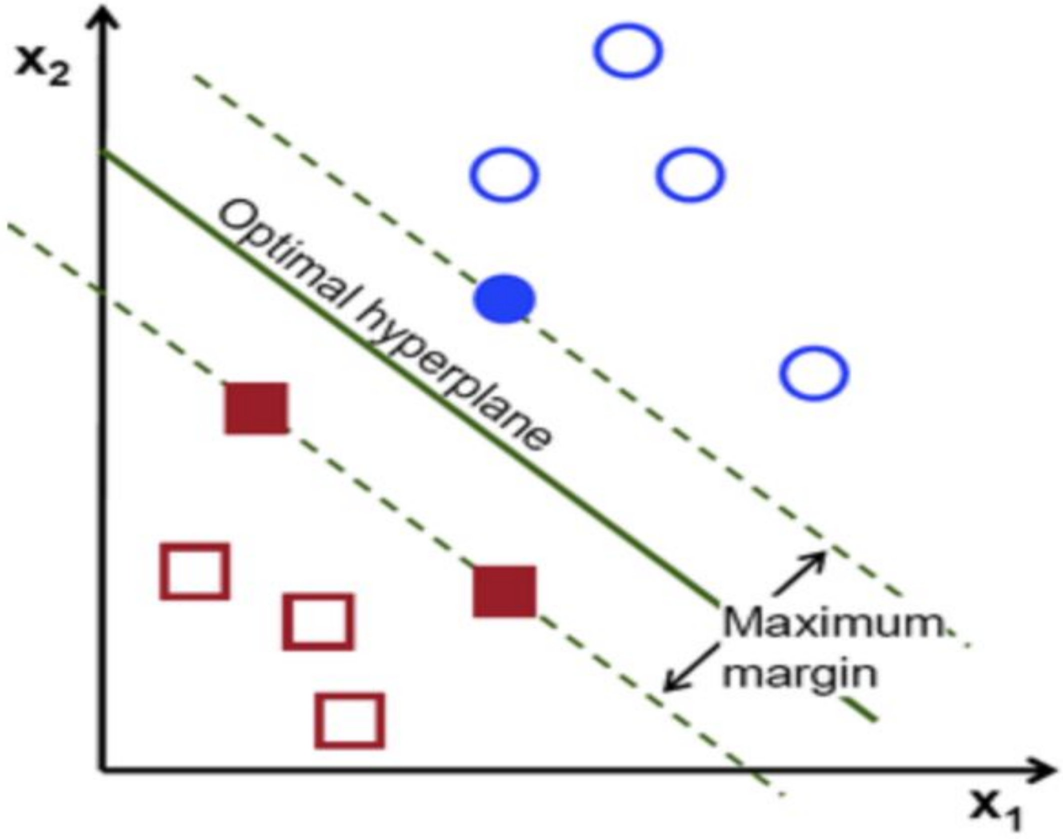

x2
x1
5
-5
5
A
B
C
D
F
G
H
I
J
a
p
w
hyperplane
$x_2=-2x_1$

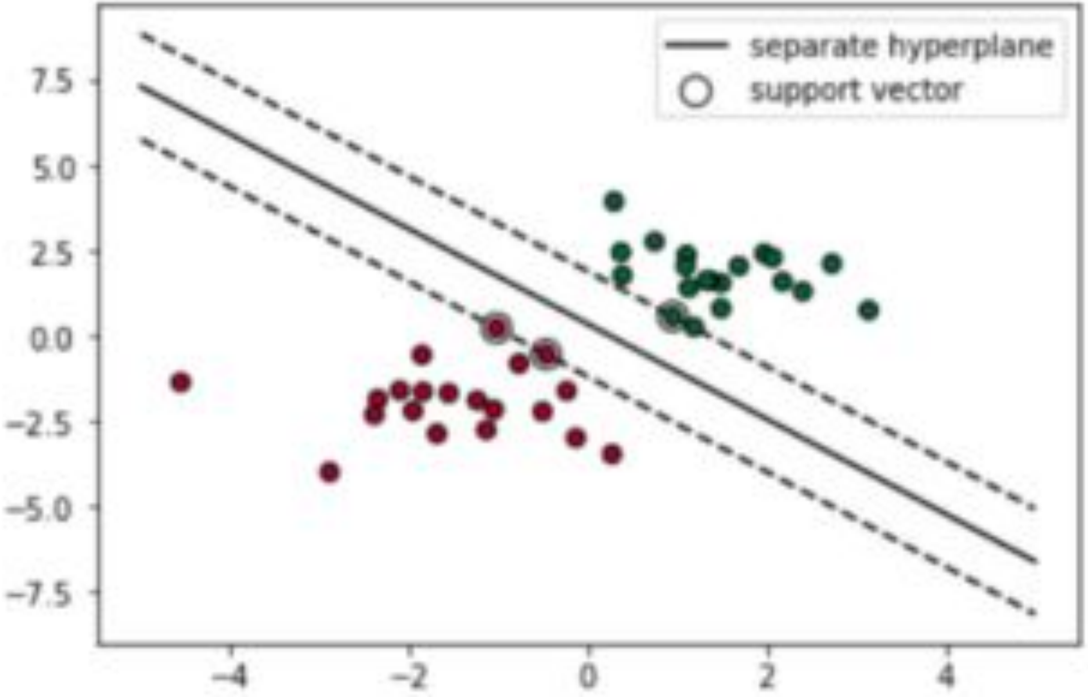

x2
x1
5
-5
5
B
D
F
A
C
a
p
w
M1
J
G
H
I
hyperplane
$x_2=-2x_1$

SVM : objective

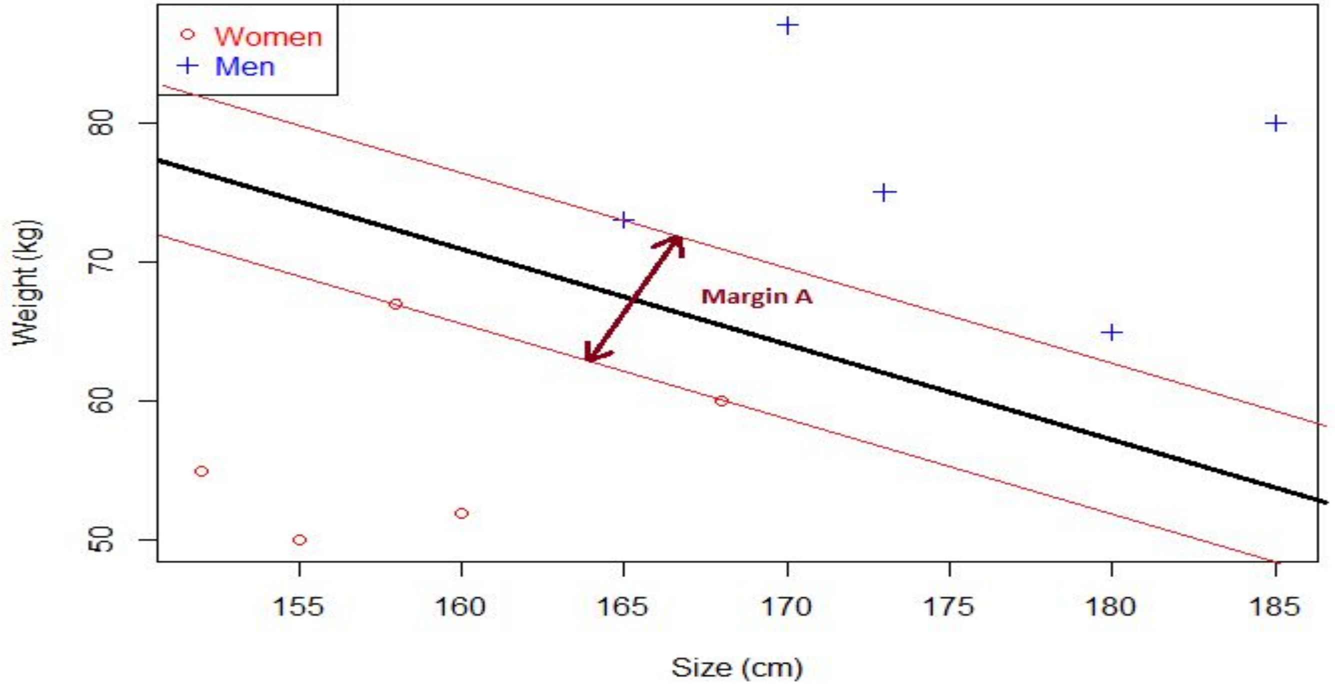
The main task of svm is to find the best separating hyperplane for the training data set which maximizes the margin.

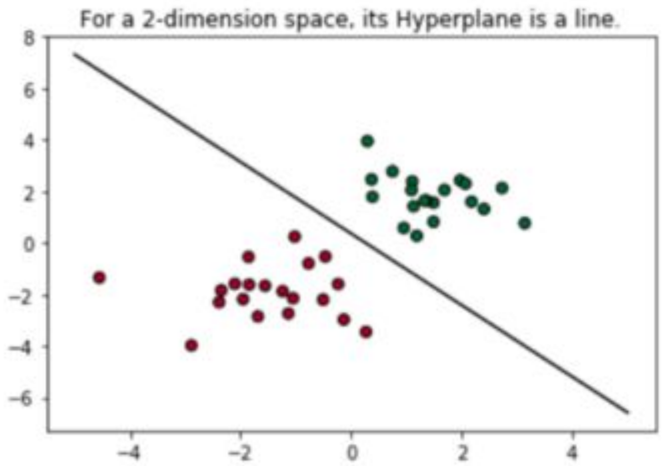
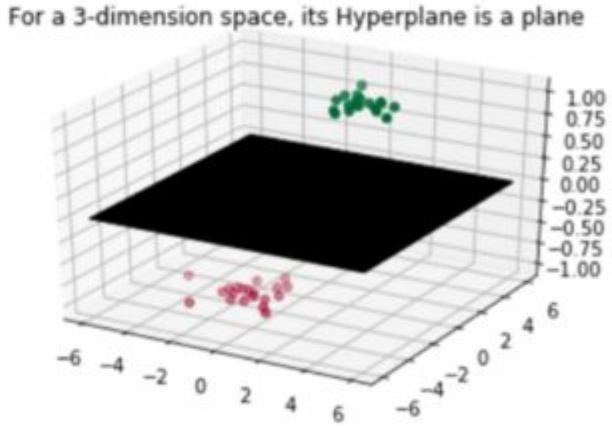



x_2
Optimal hyperplane
Maximum margin
x_1


separate hyperplane
support vector
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−4
−2
0
2
4

Basically the margin is a no man's land. There will never be any data point inside the margin.




For a 2-dimension space, its Hyperplane is a line.
8
6
4
2
0
−2
−4
−6
−4
−2
0
2
4

For a 3-dimension space, its Hyperplane is a plane
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−6
−4
−2
0
2
4
6
−6
−4
−2
0
2
4
6

SVM in linear separable cases

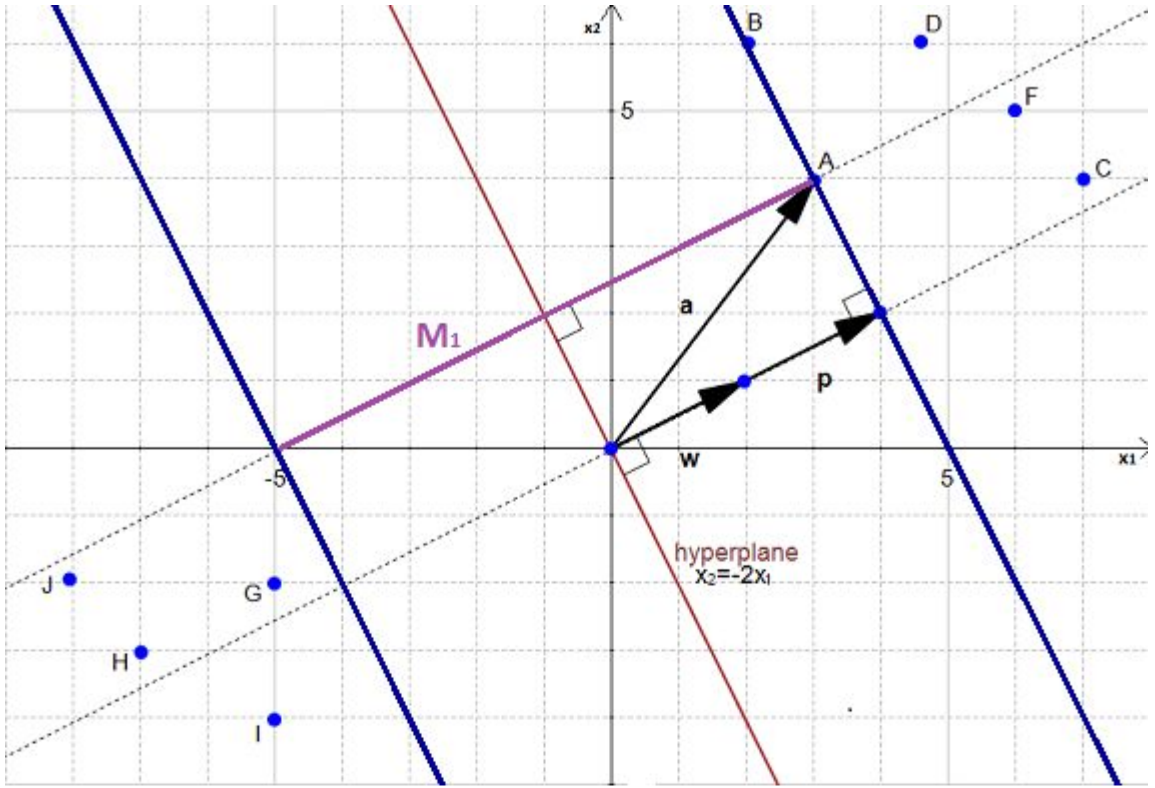
1. Follow the constraint: only look into the **separate hyperplanes**(e.g. separate lines), hyperplanes that classify classes correctly
2. Conduct optimization: pick up the one that maximizes the **margin**

Question?

How can we find the optimal hyperplane ?

How do we calculate the distance between two hyperplanes ?

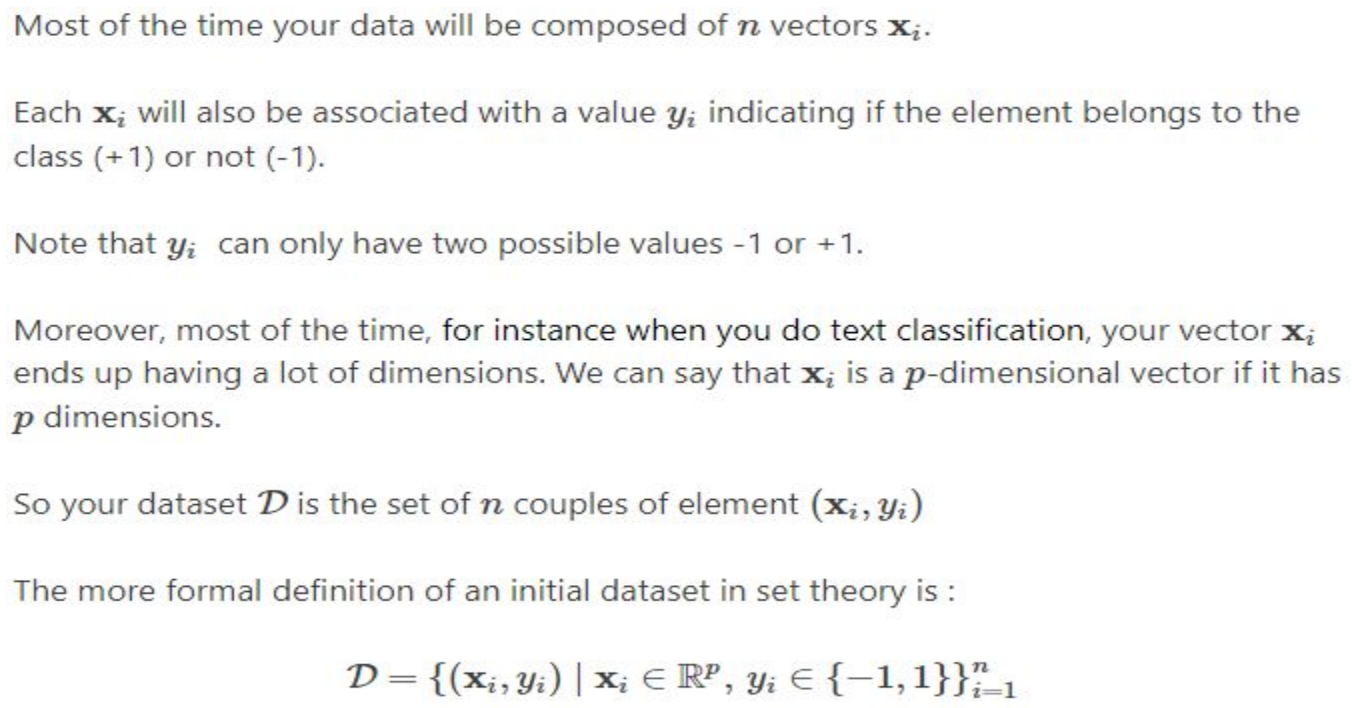
What is the SVM optimization problem ?


x2
x1
5
-5
5
A
B
C
D
F
G
H
I
J
a
p
w
M1
hyperplane
$x_2=-2x_1$

How can we find the biggest margin ?

1. You have a dataset
2. select two hyperplanes which separate the data with no points between them
3. maximize their distance (the margin)

Dataset



Most of the time your data will be composed of n vectors $\mathbf{x}_i$.

Each $\mathbf{x}_i$ will also be associated with a value y_i indicating if the element belongs to the class (+1) or not (-1).

Note that y_i can only have two possible values -1 or +1.

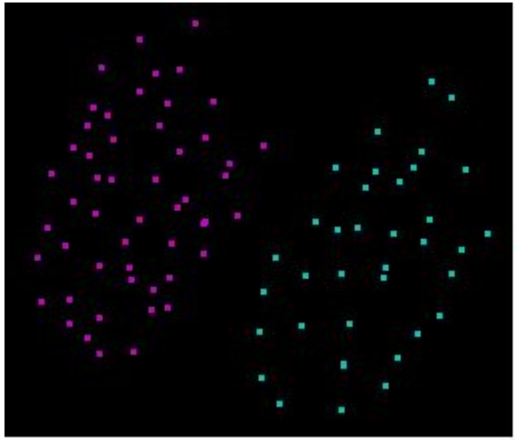
Moreover, most of the time, for instance when you do text classification, your vector $\mathbf{x}_i$ ends up having a lot of dimensions. We can say that $\mathbf{x}_i$ is a p-dimensional vector if it has p dimensions.

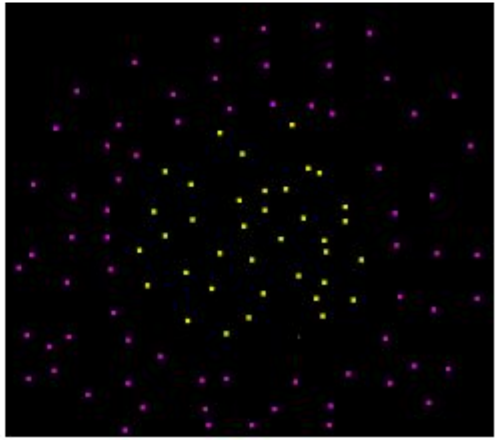
So your dataset $\mathcal{D}$ is the set of n couples of element $(\mathbf{x}_i, y_i)$

The more formal definition of an initial dataset in set theory is :

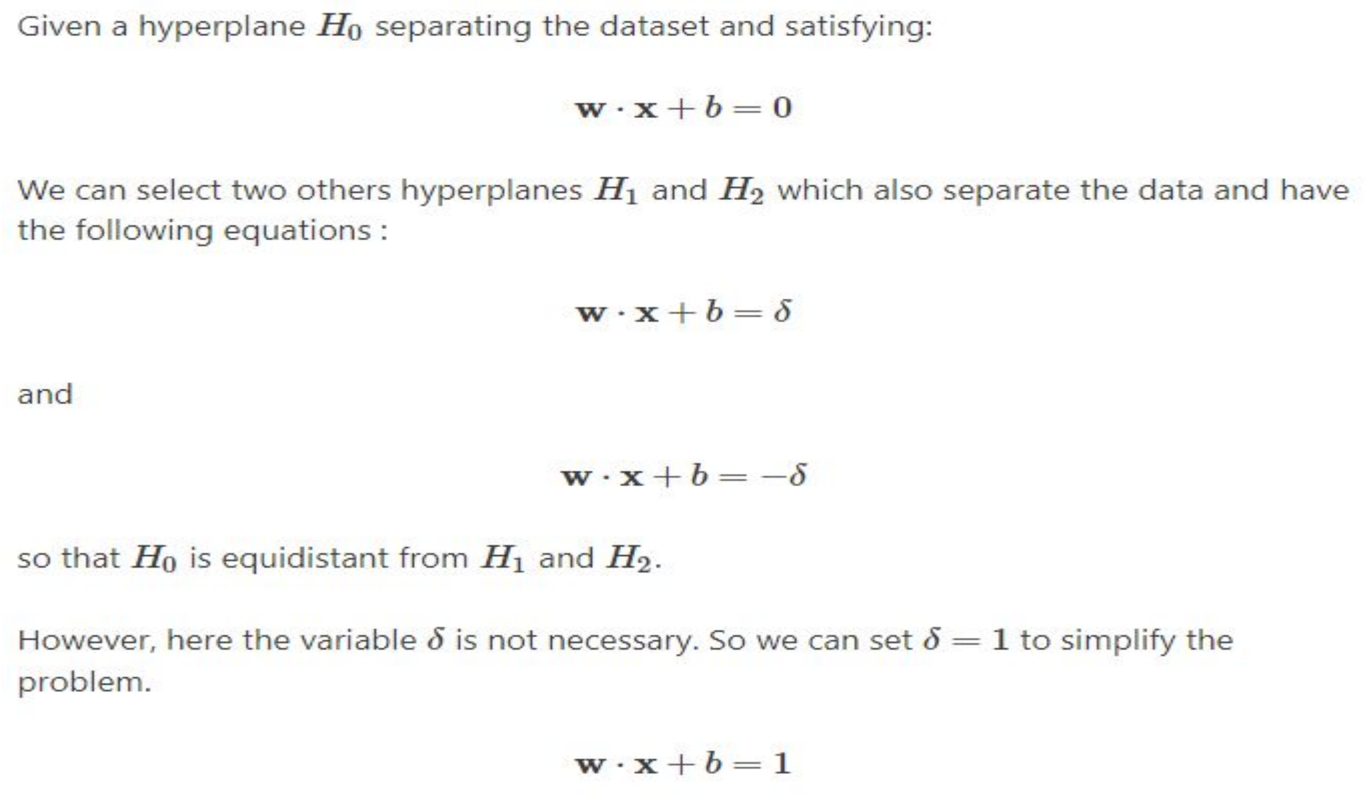
$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

SVM in linear non-separable cases


Linearly separable data


Non linearly separable data

Margin



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 :

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

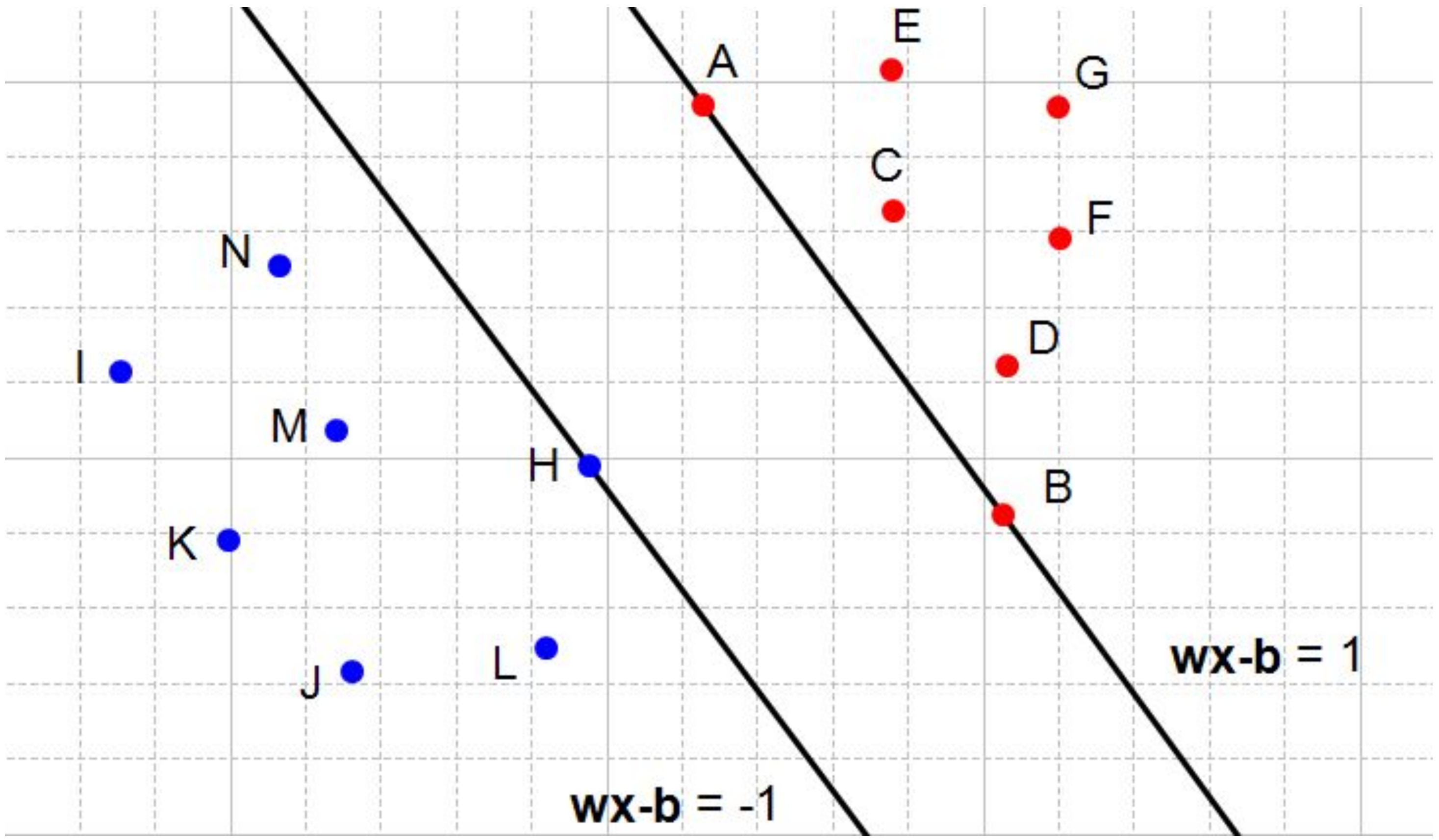
and

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

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.

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


E
A
G
C
F
N
D
I
M
H
B
K
wx-b = 1
L
J
wx-b = -1

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

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

Understanding the constraints

On the following figures, all red points have the class 1 and all blue points have the class -1.

So let's look at *Figure 4* below and consider the point A. It is red so it has the class 1 and we need to verify it does not violate the constraint $\mathbf{w} \cdot \mathbf{x_i} + b \geq 1$

When $\mathbf{x_i} = A$ we see that the point is on the hyperplane so $\mathbf{w} \cdot \mathbf{x_i} + b = 1$ and the constraint is respected. The same applies for B.

When $\mathbf{x_i} = C$ we see that the point is above the hyperplane so $\mathbf{w} \cdot \mathbf{x_i} + b > 1$ and the constraint is respected. The same applies for D, E, F and G.

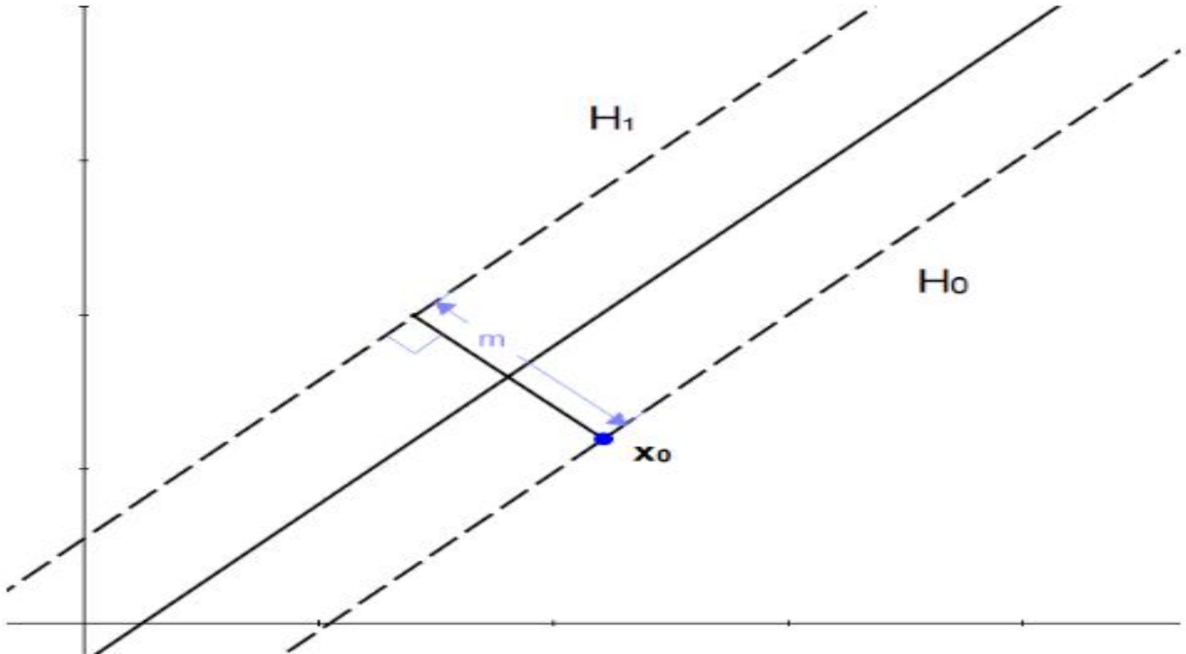
Let:

- $\mathcal{H}_0$ be the hyperplane having the equation $\mathbf{w} \cdot \mathbf{x} + b = -1$
- $\mathcal{H}_1$ be the hyperplane having the equation $\mathbf{w} \cdot \mathbf{x} + b = 1$
- $\mathbf{x}_0$ be a point in the hyperplane $\mathcal{H}_0$.

We will call m the perpendicular distance from $\mathbf{x}_0$ to the hyperplane $\mathcal{H}_1$. By definition, m is what we are used to call **the margin**.

- As $\mathbf{x}_0$ is in $\mathcal{H}_0$, m is the distance between hyperplanes $\mathcal{H}_0$ and $\mathcal{H}_1$.

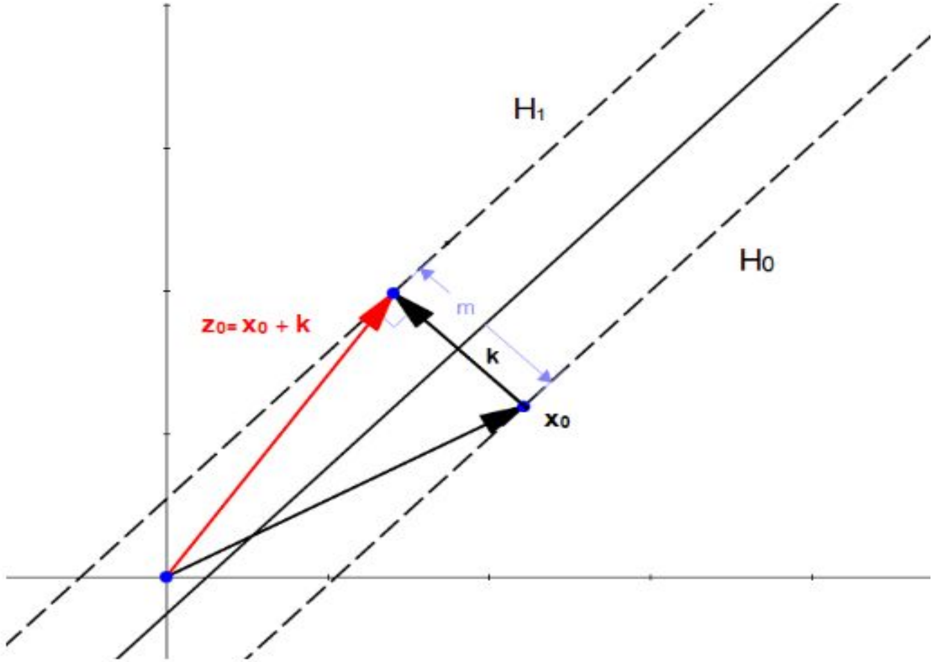
We will now try to find the value of m.



$$\mathbf{k} = m\mathbf{u} = m\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

We did it ! We transformed our scalar m into a vector $\mathbf{k}$ which we can use to perform addition with the vector $\mathbf{x}_0$.

If we start from the point $\mathbf{x}_0$ and add k we find that the point $\mathbf{z}_0 = \mathbf{x}_0 + \mathbf{k}$ is in the hyperplane $\mathcal{H}_1$ as shown on *Figure 14.*



$$\mathbf{w} \cdot \mathbf{z_0} + b = 1$$

We can replace $\mathbf{z_0}$ by $\mathbf{x_0} + \mathbf{k}$ because that is how we constructed it.

$$\mathbf{w} \cdot (\mathbf{x_0} + \mathbf{k}) + b = 1$$

We can now replace $\mathbf{k}$ using equation (9)

$$\mathbf{w} \cdot (\mathbf{x_0} + m\frac{\mathbf{w}}{\|\mathbf{w}\|}) + b = 1$$

We now expand equation (12)

$$\mathbf{w} \cdot \mathbf{x_0} + m\frac{\mathbf{w} \cdot \mathbf{w}}{\|\mathbf{w}\|} + b = 1$$

The dot product of a vector with itself is the square of its norm so :

$$\mathbf{w} \cdot \mathbf{x_0} + m\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x_0} + m\|\mathbf{w}\| + b = 1$$

$$\mathbf{w} \cdot \mathbf{x_0} + b = 1 - m\|\mathbf{w}\|$$

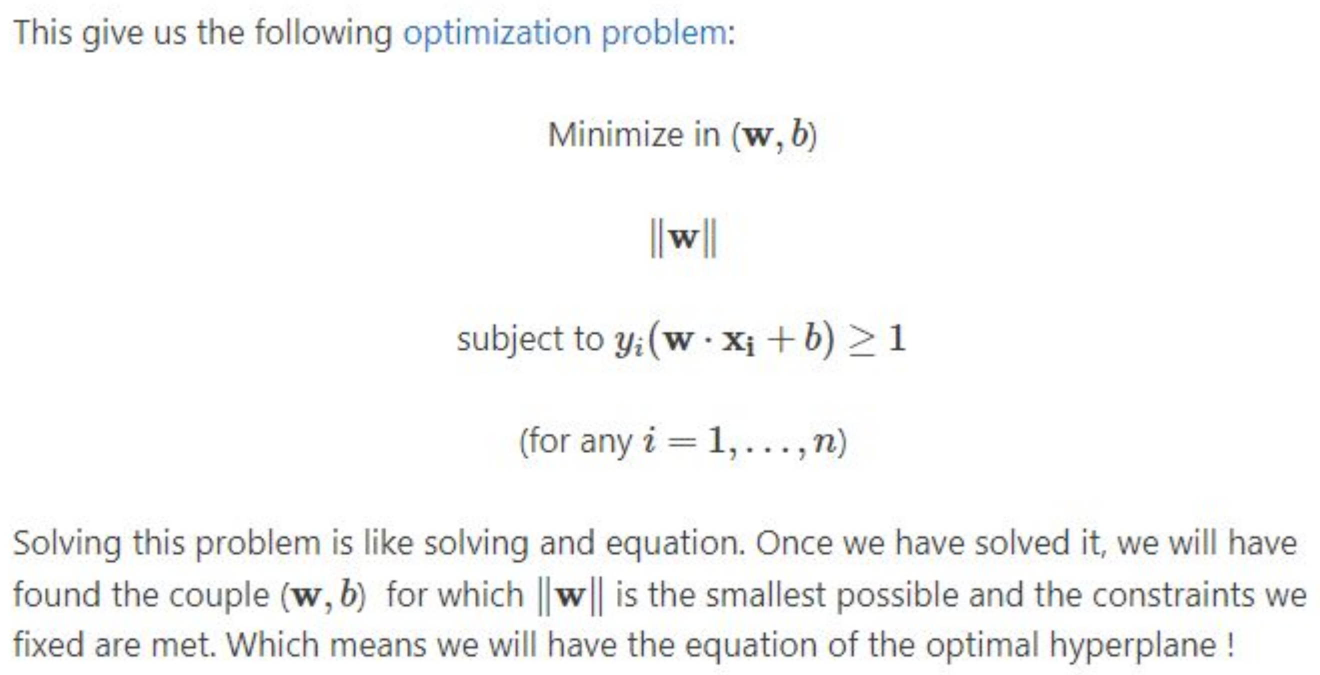
As $\mathbf{x_0}$ is in $\mathcal{H}_0$ then $\mathbf{w} \cdot \mathbf{x_0} + b = -1$

$$-1 = 1 - m\|\mathbf{w}\|$$

$$m\|\mathbf{w}\| = 2$$

$$m = \frac{2}{\|\mathbf{w}\|}$$

SVM optimization problem



This give us the following optimization problem:

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

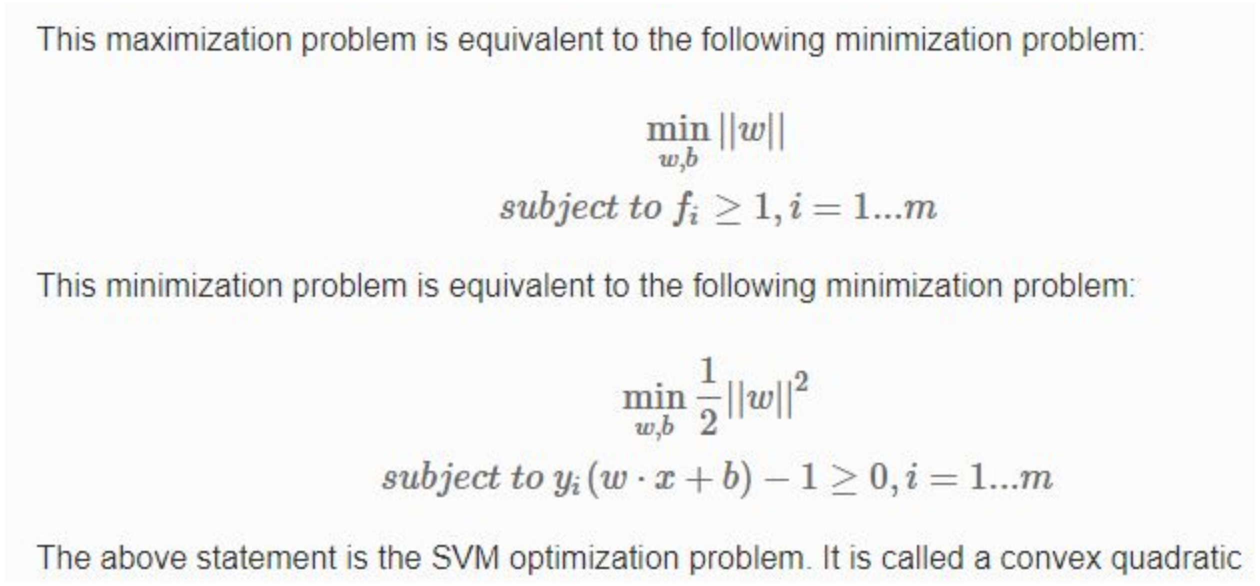
$$\|\mathbf{w}\|$$

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

(for any $i = 1, \dots, n$)

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 !

SVM optimization problem



This maximization problem is equivalent to the following minimization problem:

$$\min_{w,b} ||w||$$
$$subject\ to\ f_i \geq 1, i = 1...m$$

This minimization problem is equivalent to the following minimization problem:

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$subject\ to\ y_i(w \cdot x + b) - 1 \geq 0, i = 1...m$$

The above statement is the SVM optimization problem. It is called a convex quadratic

Solving SVM optimization problem – Hard Margin SVM

Lagrange stated that if we want to find the minimum of f under the equality constraint g, we just need to solve for:

$$\nabla f(x) - \alpha \nabla g(x) = 0$$

α is called the Lagrange multiplier.

In terms of the SVM optimization problem, $f(w) = \frac{1}{2}\|w\|^2$, $g(w,b) = y_i(w \cdot x + b) - 1, i = 1...m$. The Lagrangian function is then $\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x + b) - 1]$.

To solve for $\nabla \mathcal{L}(w,b,\alpha) = 0$ analytically, the number of examples has to be small. Thus, we will rewrite the problem using the duality principle.

Equivalently, we need to solve the following Lagrangian primal problem:

$$\min_{w,b} \ \max \mathcal{L}(w,b,\alpha)$$
$$subject\ to\ \alpha_i \geq 0, i = 1...m$$

More accurately, α should be KKT (Karush-Kuhn-Tucker) multipliers because we are dealing with inequality constraints here. But we will use the term Lagrangian multipliers for continuity.

There is an α for each example, we need to maximize $\mathcal{L}(w,b,\alpha)$ for all examples. And there is a (**w**,b) for each hyperplane, we need to minimize the $max \mathcal{L}(w,b,\alpha)$ in the meantime. □

The Lagrangian function is:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{m} \alpha_i [y_i (w \cdot x + b) - 1]$$

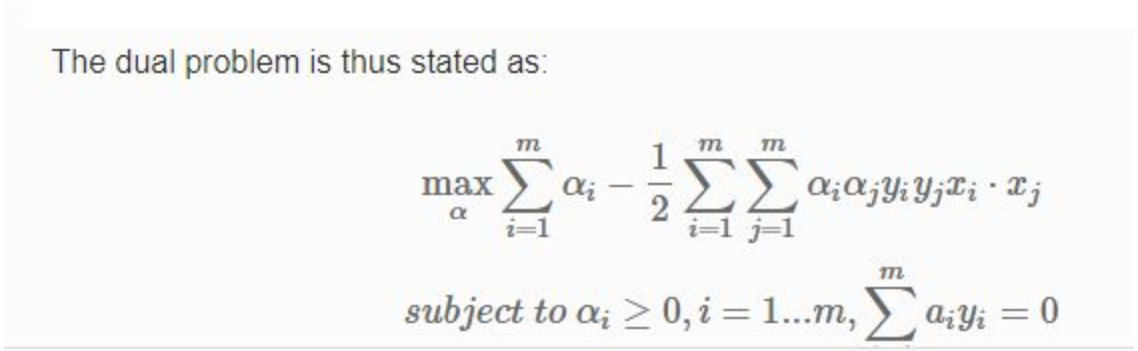
For the dual problem, we have that:

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

$$\nabla_b \mathcal{L}(w,b,\alpha) = - \sum_{i=1}^{m} \alpha_i y_i = 0$$

From the above two equations, we get $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$. We substitute them into the Lagrangian function $\mathcal{L}$ and get:

$$W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

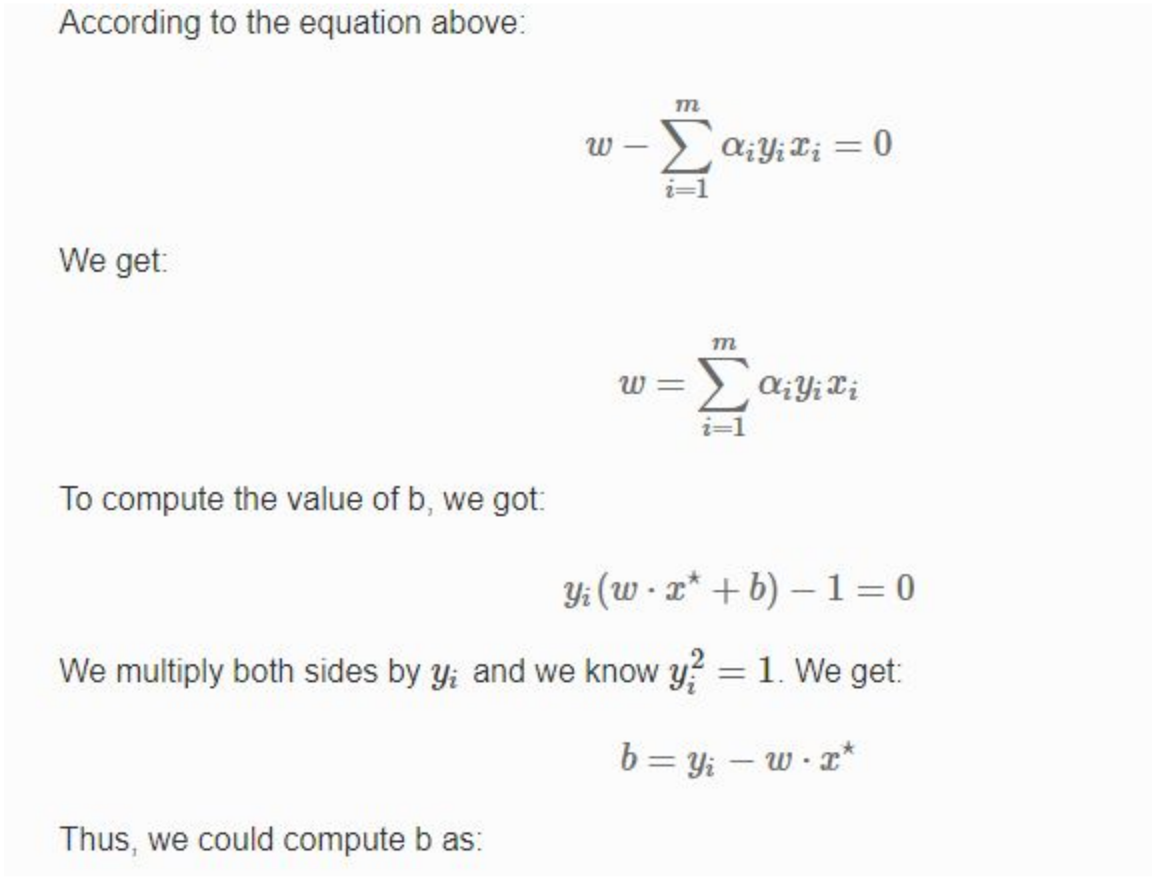


The dual problem is thus stated as:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

$$subject\ to\ \alpha_i \geq 0, i = 1...m, \sum^{m} a_i y_i = 0$$

Compute **w** and b



According to the equation above:

$$w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

We get:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

To compute the value of b, we got:

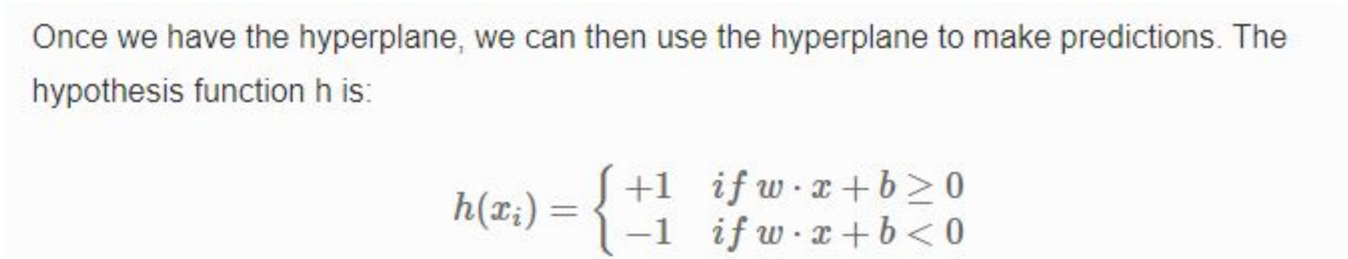
$$y_i(w \cdot x^\star + b) - 1 = 0$$

We multiply both sides by y_i and we know $y_i^2 = 1$. We get:

$$b = y_i - w \cdot x^\star$$

Thus, we could compute b as:

Classifier



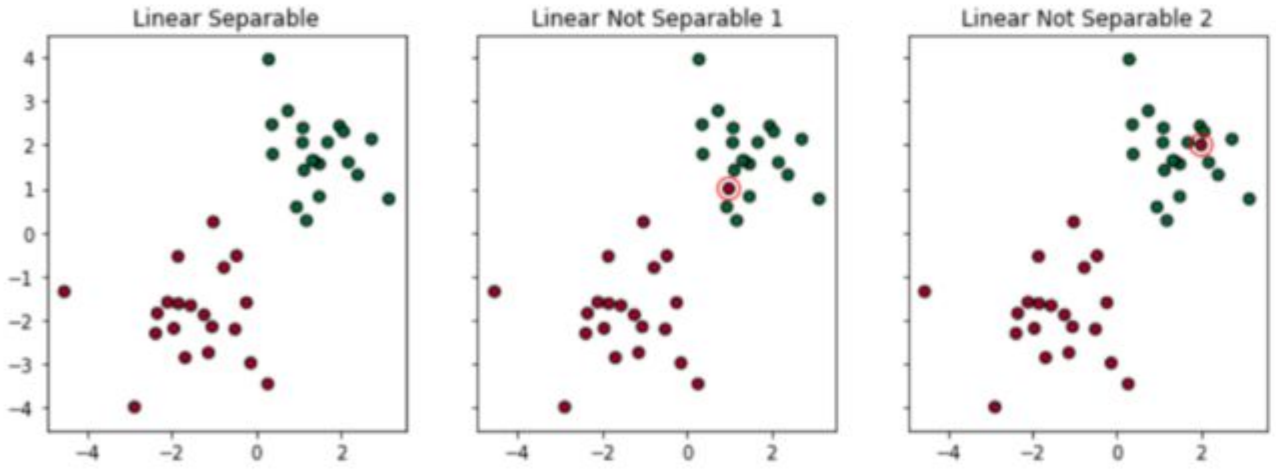
Once we have the hyperplane, we can then use the hyperplane to make predictions. The hypothesis function h is:

$$h(x_i) = \begin{cases} +1 & if\, w \cdot x + b \geq 0 \\ -1 & if\, w \cdot x + b < 0 \end{cases}$$

In the linearly separable case, SVM is trying to find the hyperplane that maximizes the margin, with the condition that both classes are classified correctly. But in reality, datasets are probably never linearly separable, so the condition of 100% correctly classified by a hyperplane will never be met.

SVM address non-linearly separable cases by introducing two concepts: **Soft Margin** and **Kernel Tricks.**

SVM in linear non-separable cases



SVM in linear non-separable cases

1. **Soft Margin:** try to find a line to separate, but tolerate one or few misclassified dots

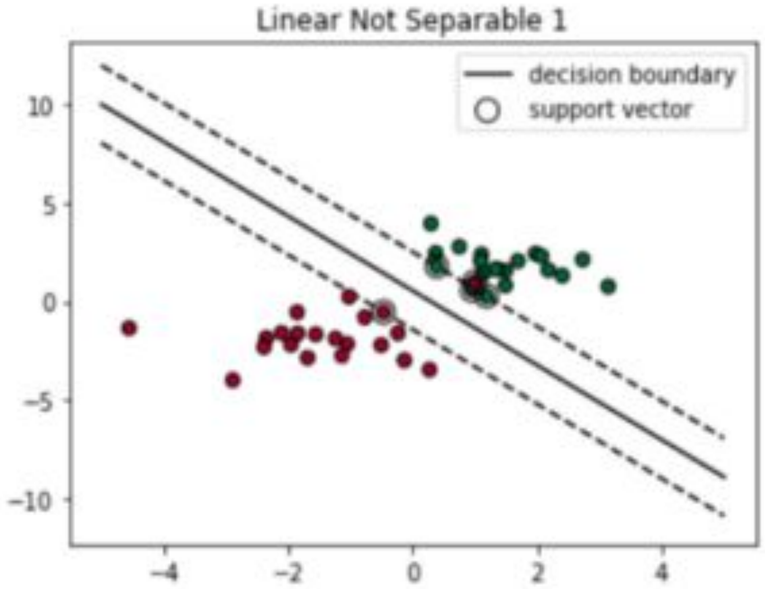
   (e.g. the dots circled in red)

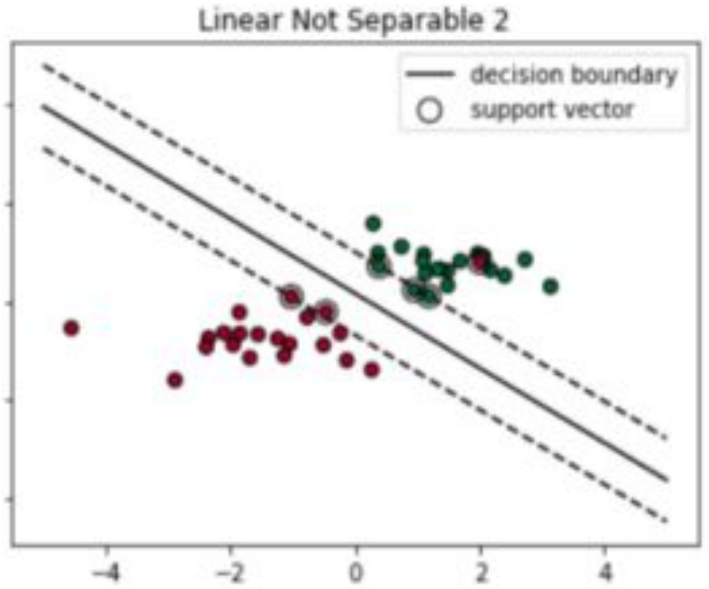
2. **Kernel Trick:** try to find a non-linear decision boundary

Soft Margin

Two types of misclassifications are tolerated by SVM under soft margin:

- The dot is on the wrong side of the decision boundary but on the correct side/ on the margin (shown in left)
- The dot is on the wrong side of the decision boundary and on the wrong side of the margin (shown in right)


Linear Not Separable 1
decision boundary
support vector
10
5
0
−5
−10
−4
−2
0
2
4


Linear Not Separable 2
decision boundary
support vector
−4
−2
0
2
4

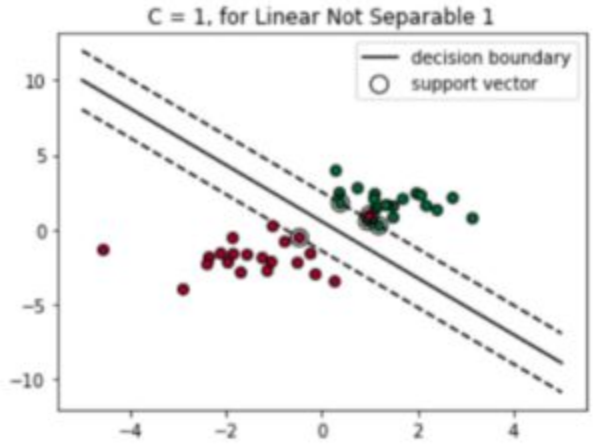
Soft Margin

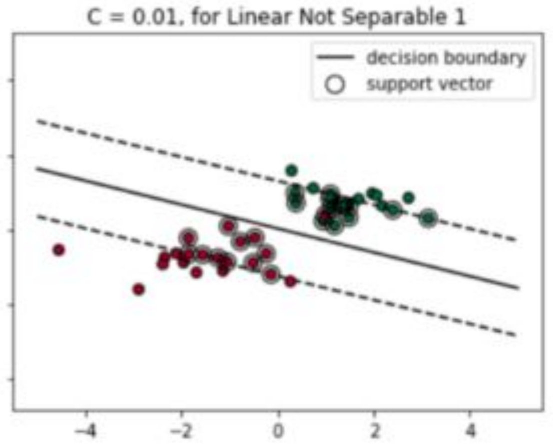
Applying Soft Margin, SVM tolerates a few dots to get misclassified and tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.

Degree of tolerance

How much tolerance(soft) we want to give when finding the decision boundary is an important hyper-parameter for the SVM (both linear and nonlinear solutions). In Sklearn, it is represented as the penalty term — 'C'. The bigger the C, the more penalty SVM gets when it makes misclassification. Therefore, the narrower the margin is and fewer support vectors the decision boundary will depend on.

```
# Default Penalty/Default Tolerance

clf = svm.SVC(kernel='linear', C=1)

# Less Penalty/More Tolearance

clf2 = svm.SVC(kernel='linear', C=0.01)
```

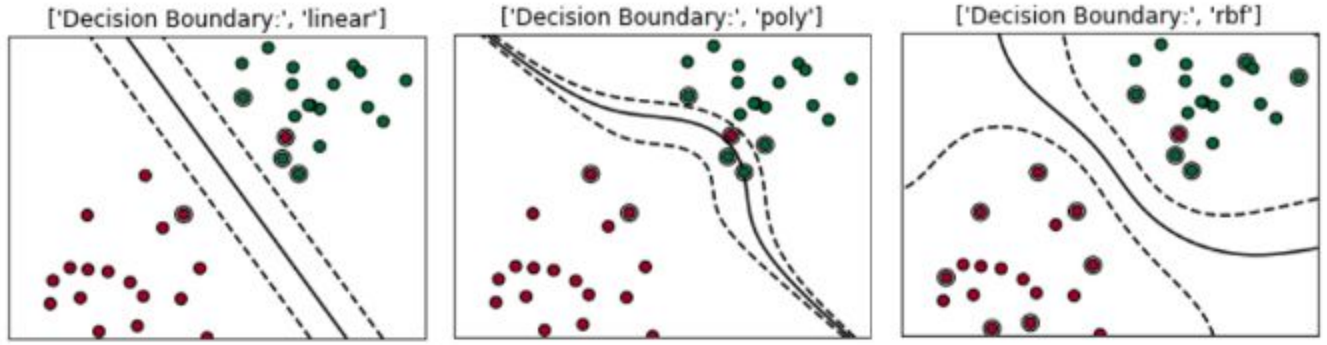

C = 1, for Linear Not Separable 1
decision boundary
support vector
10
5
0
−5
−10
−4
−2
0
2
4


C = 0.01, for Linear Not Separable 1
decision boundary
support vector
−4
−2
0
2
4

Kernel Trick

What Kernel Trick does is it utilizes existing features, applies some transformations, and creates new features. Those new features are the key for SVM to find the nonlinear decision boundary.

In Sklearn — svm.SVC(), we can choose ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable as our kernel/transformation. I will give examples of the two most popular kernels — Polynomial and Radial Basis Function(RBF).


['Decision Boundary:', 'linear']
['Decision Boundary:', 'poly']
['Decision Boundary:', 'rbf']

Polynomial kernel

Think of the polynomial kernel as a transformer/processor to generate new features by applying the polynomial combination of all the existing features.

To illustrate the benefit of applying a polynomial transformer, let’s use a simple example:

Polynomial kernel

Existing Feature: X = np.array([-2,-1,0, 1,2])

Label: Y = np.array([1,1,0,1,1])

it’s impossible for us to find a line to separate the yellow (1)and purple (0) dots (shown on the left).

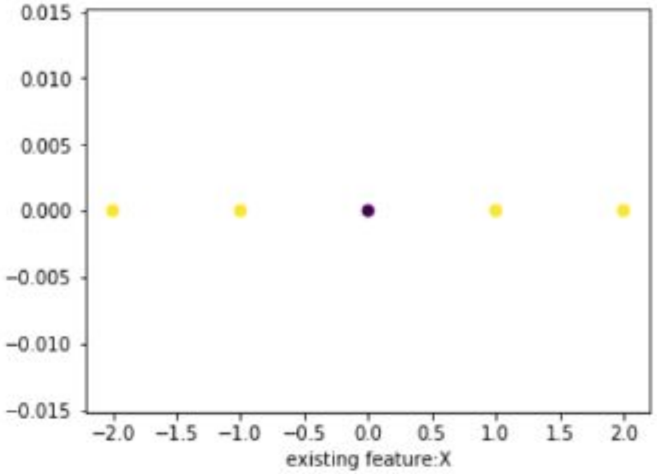
But, if we apply transformation X^2 to get:

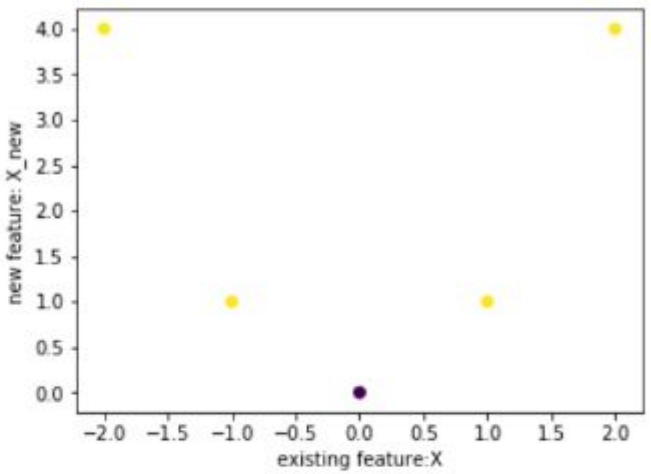
New Feature: X = np.array([4,1,0, 1,4])

By combing the existing and new feature, we can certainly draw a line to separate the yellow purple dots (shown on the right).

Support vector machine with a polynomial kernel can generate a non-linear decision boundary using those polynomial features.

Polynomial kernel



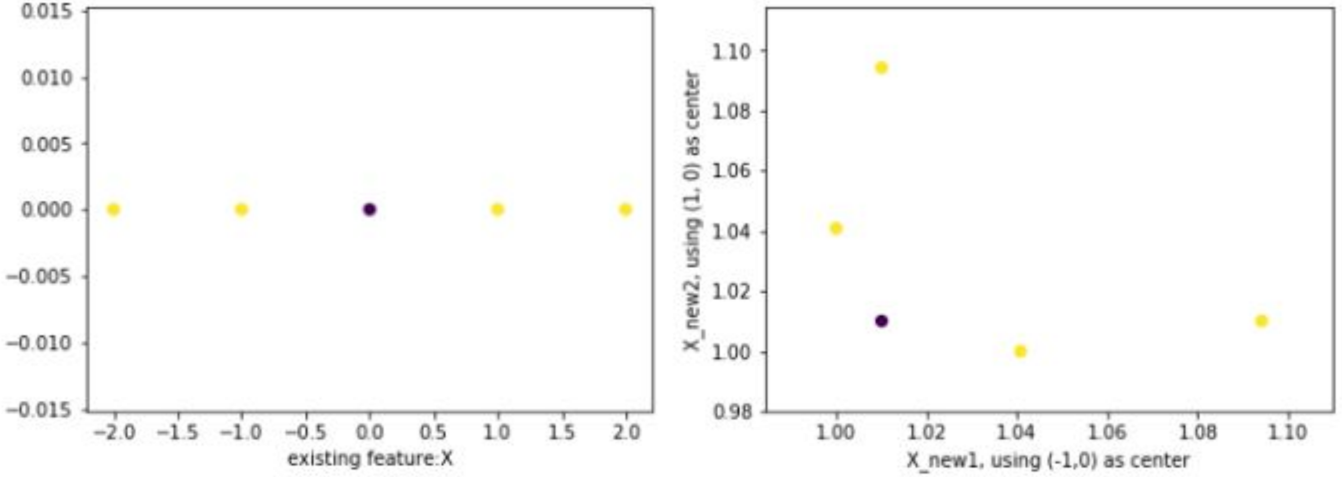


Radial Basis Function (RBF) kernel

Think of the Radial Basis Function kernel as a transformer/processor to generate new features by measuring the distance between all other dots to a specific dot/dots — centers. The most popular/basic RBF kernel is the Gaussian Radial Basis Function:

$$\phi(x, center) = exp(-\gamma \|x - center\|^2)$$

gamma (γ) controls the influence of new features — **Φ(x, center)** on the decision boundary. The higher the gamma, the more influence of the features will have on the decision boundary, more wiggling the boundary will be.



Existing Feature: X = np.array([-2,-1,0, 1,2])

Label: Y = np.array([1,1,0,1,1])

Again,But, if we apply Gaussian RBF transformation using two centers (-1,0) and (2,0) to get new features, we will then be able to draw a line to separate the yellow purple dots (on the right):

New Feature 1: X_new1 = array([1.01, 1.00, 1.01, 1.04, 1.09])

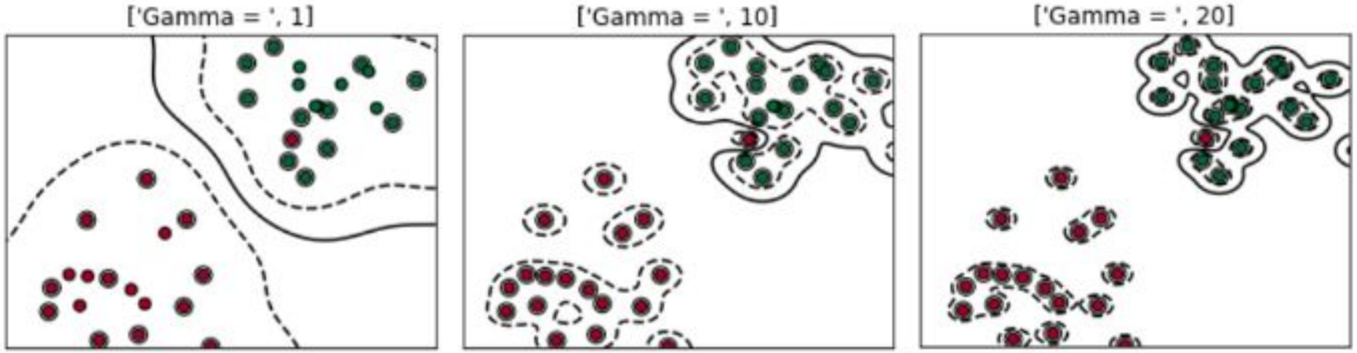
New Feature 2: X_new2 = array([1.09, 1.04, 1.01, 1.00, 1.01])

it’s impossible for us to find a line to separate the dots (on left hand).

```
# Note for how we get New Feature 1, e.g. 1.01:

Φ(x1,center1) = np.exp(np.power(-(gamma*(x1-center1)),2)) = 1.01

# gamma = 0.1

# center1 = [-1,0]

# x1 = [-2,0]
```

Similar to the penalty term — C in the soft margin, Gamma is a hyperparameter that we can tune for when we use SVM with kernel.

```
# Gamma is small, influence is small

clf = svm.SVC(kernel='rbf', Gamma=1)

# Gamma gets bigger, influence increase, the decision boundary get wiggled

clf2 = svm.SVC(kernel='rbf', Gamma=10)

# Gamma gets too big, influence too much, the decision boundary get too wiggled

clf3 = svm.SVC(kernel='rbf', Gamma=20)
```


['Gamma = ', 1]
['Gamma = ', 10]
['Gamma = ', 20]

To sum up, SVM in the linear nonseparable cases:

By combining the soft margin (tolerance of misclassifications) and kernel trick together, Support Vector Machine is able to structure the decision boundary for linear non-separable cases.

Hyper-parameters like C or Gamma control how wiggling the SVM decision boundary could be.

the higher the C, the more penalty SVM was given when it misclassified, and therefore the less wiggling the decision boundary will be

the higher the gamma, the more influence the feature data points will have on the decision boundary, thereby the more wiggling the boundary will be