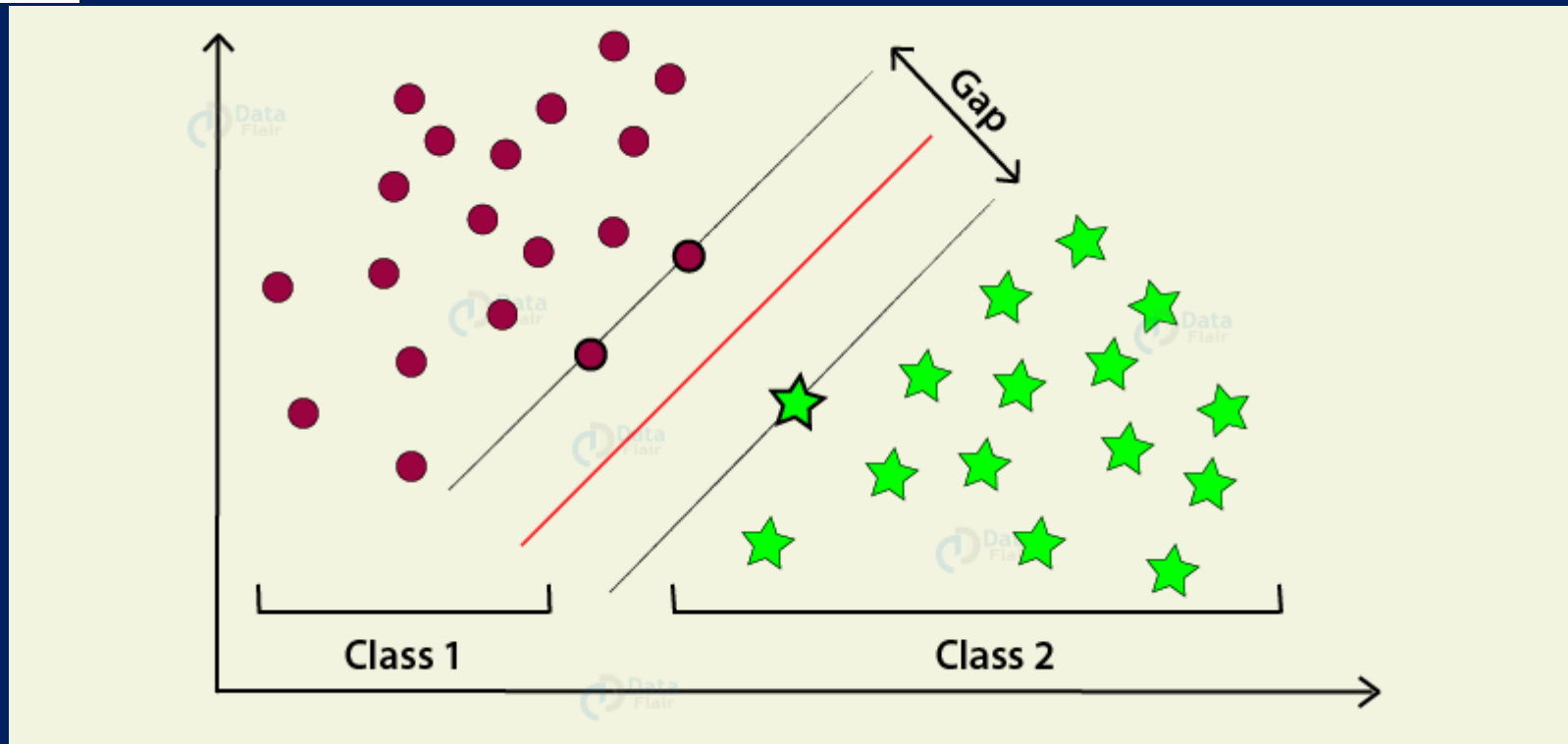


MCSE0007: Machine Learning



Support Vector Machines (SVM)

History

- Support Vector Machines (SVM) is arguably the most important & interesting discovery in Machine Learning.
- SVM were first introduced by Vladimir Vapnik



History ...

- The study on Statistical Learning Theory was started in the 1960s by Vapnik
- Statistical Learning Theory is the theory about Machine Learning Principle from a small sample size
- Support Vector Machine is a practical learning method based on Statistical Learning Theory and introduced by Boser, Guyon and Vapnik in COLT-92
- A simple SVM could beat a sophisticated neural network with elaborated features in a handwriting recognition task.

Overview: Supervised Learning

■ Example: Spam filtering

	viagra	learning	the	dating	nigeria	<i>spam?</i>
$\vec{x}_1 = ($	1	0	1	0	0	$y_1 = 1$
$\vec{x}_2 = ($	0	1	1	0	0	$y_2 = -1$
$\vec{x}_3 = ($	0	0	0	0	1	$y_3 = 1$

- **Instance space $\mathbf{x} \in \mathbf{X}$ ($|\mathbf{X}| = n$ data points)**
 - Binary or real-valued feature vector \mathbf{x} of word occurrences
 - d features (words + other things, $d \sim 100,000$)
- **Class $\mathbf{y} \in \mathbf{Y}$**
 - \mathbf{y} : Spam (+1), Ham (-1)

More generally: Supervised Learning

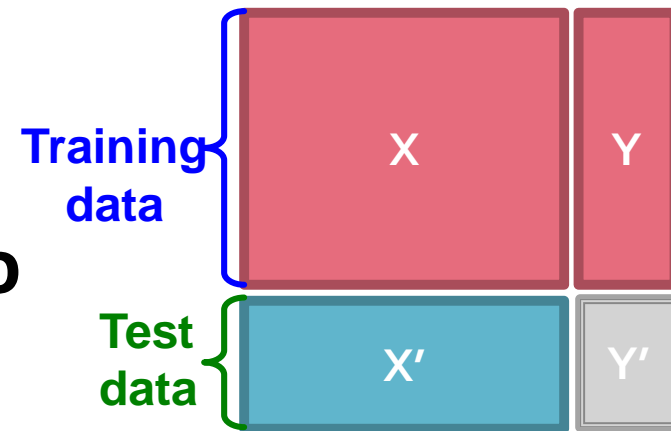
- Would like to do **prediction**:
estimate a function $f(x)$ so that $y = f(x)$
- Where y can be:
 - **Real number**: Regression
 - **Categorical**: Classification
 - **Complex object**:
 - Ranking of items, Parse tree, etc.
- Data is **labeled**:
 - Have many pairs $\{(x, y)\}$
 - x ... vector of binary, categorical, real valued features
 - y ... class ($\{+1, -1\}$, or a real number)

Overview: Supervised Learning

- **Task:** Given data (X, Y) build a model $f()$ to predict Y' based on X'

- **Strategy:** Estimate $y = f(x)$ on (X, Y) .

Hope that the same $f(x)$ also works to predict unknown Y'



- The “hope” is called **generalization**
 - **Overfitting:** If $f(x)$ predicts well Y but is unable to predict Y'
- **We want to build a model that generalizes well to unseen data**
 - But, how can we well on data we have never seen before?!?

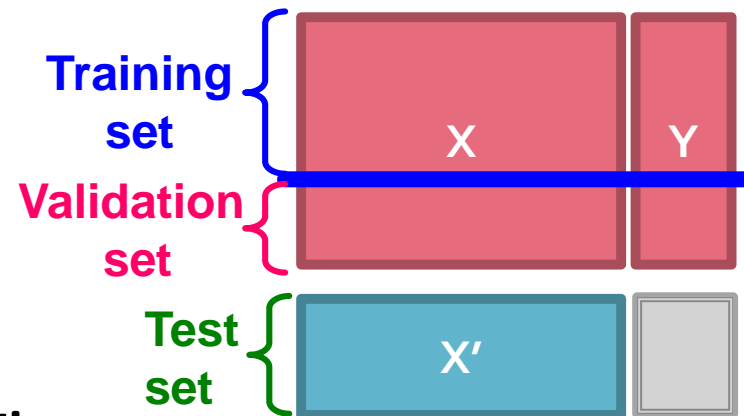


Overview: Supervised Learning

- **Idea:** Pretend we do not know the data/labels we actually do know

- Build the model $f(x)$ on the training data
See how well $f(x)$ does on the test data

- If it does well, then apply it also to X'



- **Refinement: Cross validation**

- Splitting into training/validation set is brutal
- Let's split our data (X,Y) into 10-folds (buckets)
- Take out 1-fold for validation, train on remaining 9
- Repeat this 10 times, report average performance

Linear models for classification

■ Binary classification:

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } w_1 x_1 + w_2 x_2 + \dots + w_d x_d \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

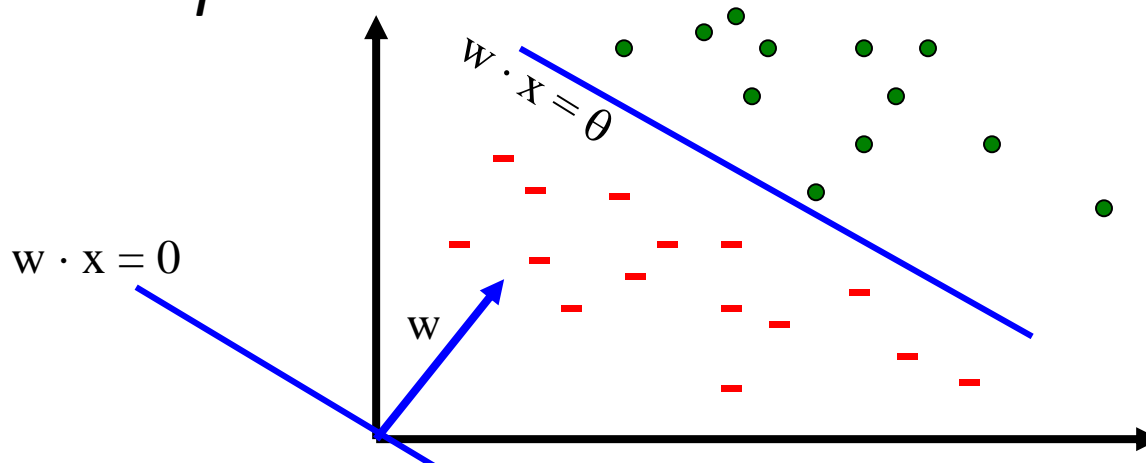
Decision
boundary
is linear

■ **Input:** Vectors $\mathbf{x}^{(j)}$ and labels $y^{(j)}$

- Vectors $\mathbf{x}^{(j)}$ are real valued where $\|\mathbf{x}\|_2 = 1$

■ **Goal:** Find vector $\mathbf{w} = (w_1, w_2, \dots, w_d)$

- Each w_i is a real number



Note:

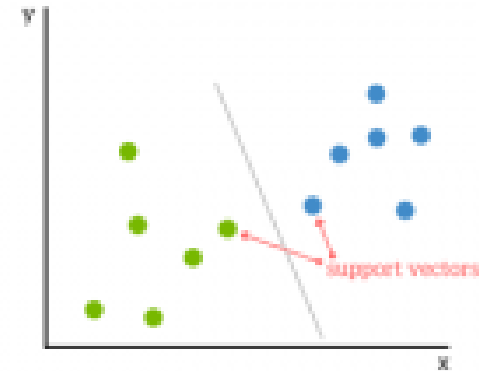
$$\mathbf{x} \Leftrightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$$

$$\mathbf{w} \Leftrightarrow \langle \mathbf{w}, -\theta \rangle$$

Objectives

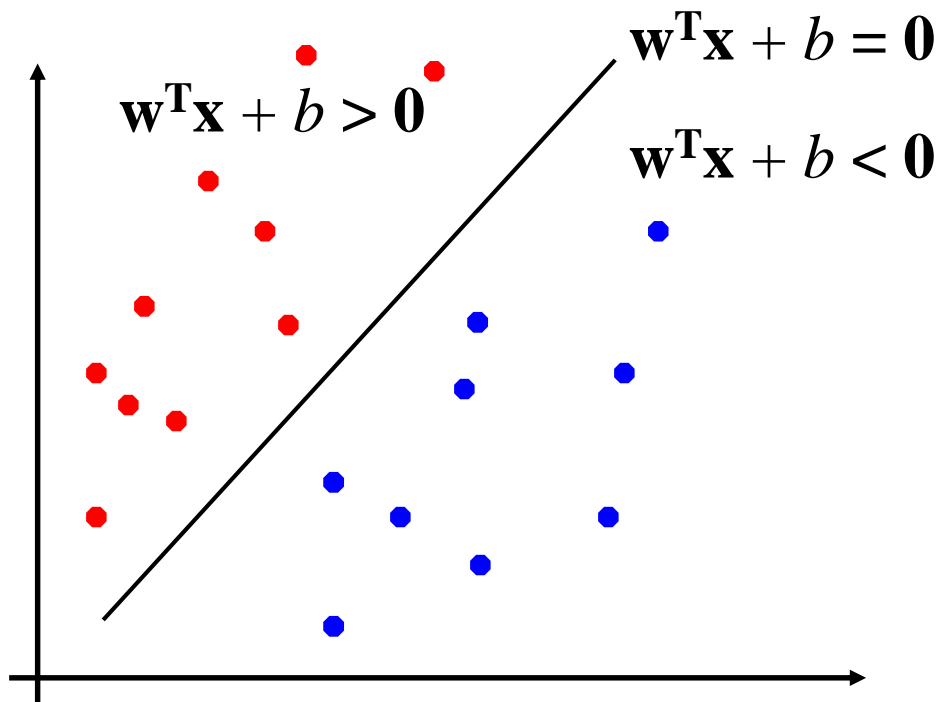
- **Support vector machines (SVM)** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- They analyze the large amount of data to identify patterns from them.

SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes



Two Class Problem: Linear Separable Case

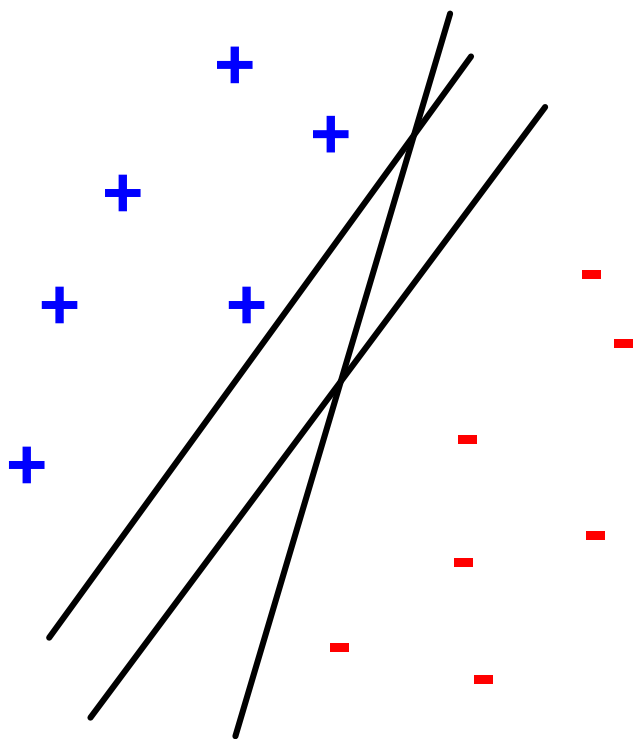
- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

Two Class Problem: Linear Separable Case

- Want to separate “+” from “-” using a line



Data:

- Training examples:

- $(x_1, y_1) \dots (x_n, y_n)$

- Each example i :

- $x_i = (x_i^{(1)}, \dots, x_i^{(d)})$

- $x_i^{(j)}$ is real valued

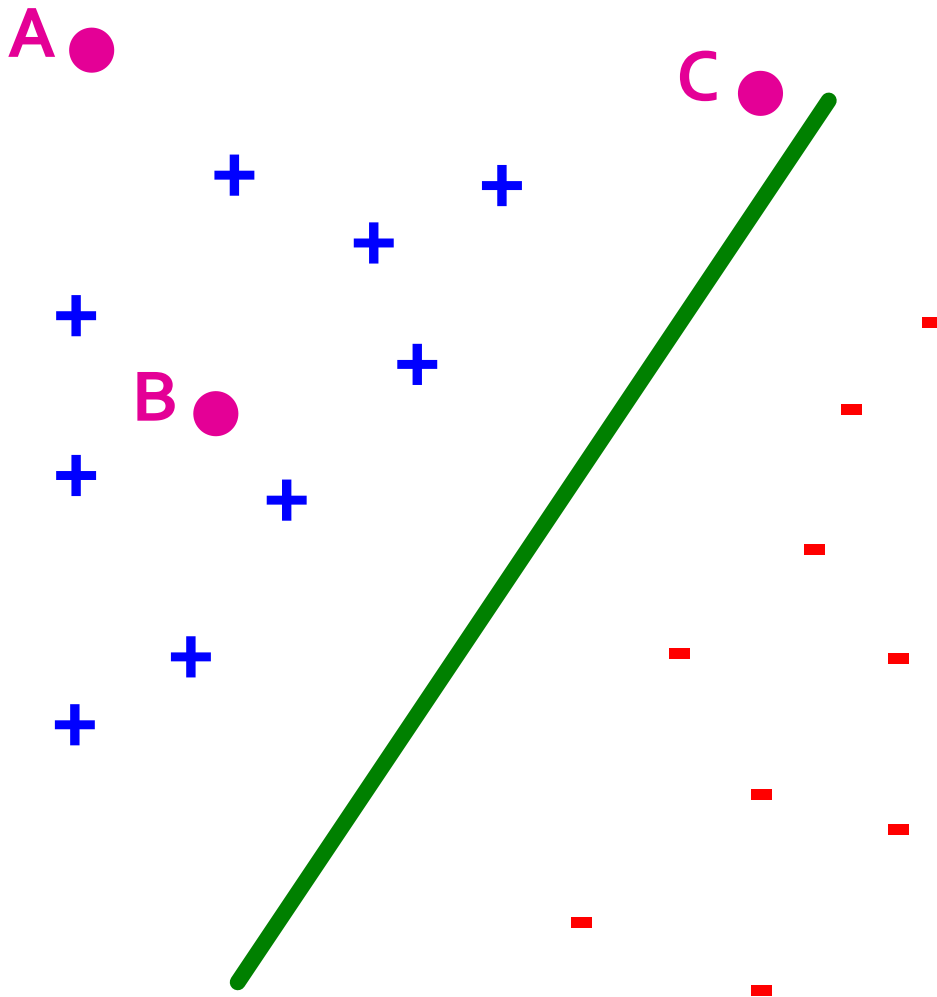
- $y_i \in \{-1, +1\}$

- Inner product:

$$w \cdot x = \sum_{j=1}^d w^{(j)} \cdot x^{(j)}$$

Which is best linear separator (defined by w)?

Two Class Problem: Linear Separable Case



- Distance from the separating hyperplane corresponds to the “confidence” of prediction
- Example:
 - We are more sure about the class of **A** and **B** than of **C**

Support Vectors

- Support Vectors are simply the co-ordinates of individual observation.
- Support vectors are the data points that lie **closest to the decision surface** (or hyperplane)
- They are the data points most difficult to classify
- They have direct bearing on the optimum location of the decision surface
- Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

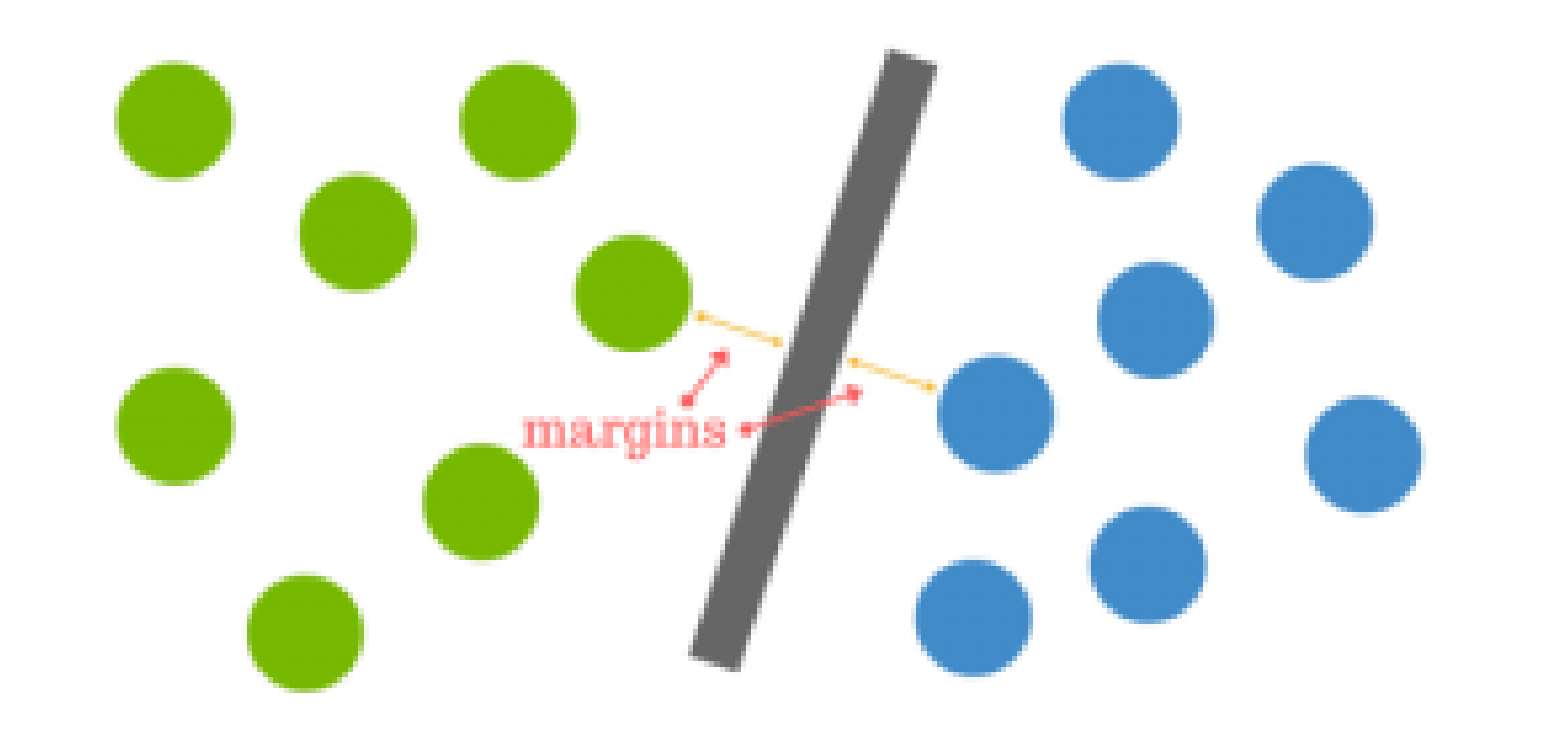
What is a hyperplane?

- As a simple example, for a classification task with only two features, you can think of a hyperplane as a line that linearly separates and classifies a set of data.
- Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.
- So when new testing data are added, whatever side of the hyperplane it lands will decide the class that we assign to it.

How do we find the right hyperplane?

- How do we best segregate the two classes within the data?
- The distance between the hyperplane and the nearest data point from either set is known as the **margin**.
- The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.
- **There will never be any data point inside the margin.**

How do we find the right hyperplane? ...



How do we find the right hyperplane? ...

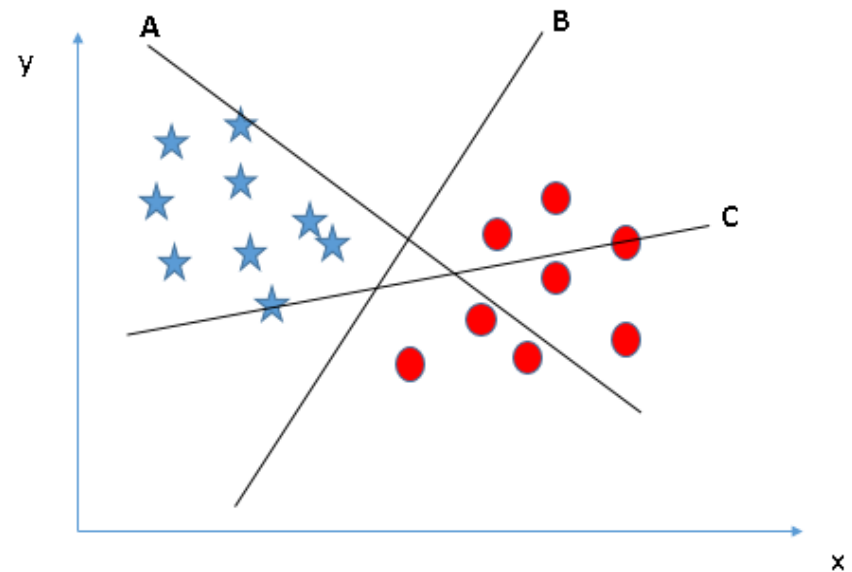


Remember a thumb rule to identify the right hyper-plane:

“Select the hyper-plane which segregates the two classes better”.

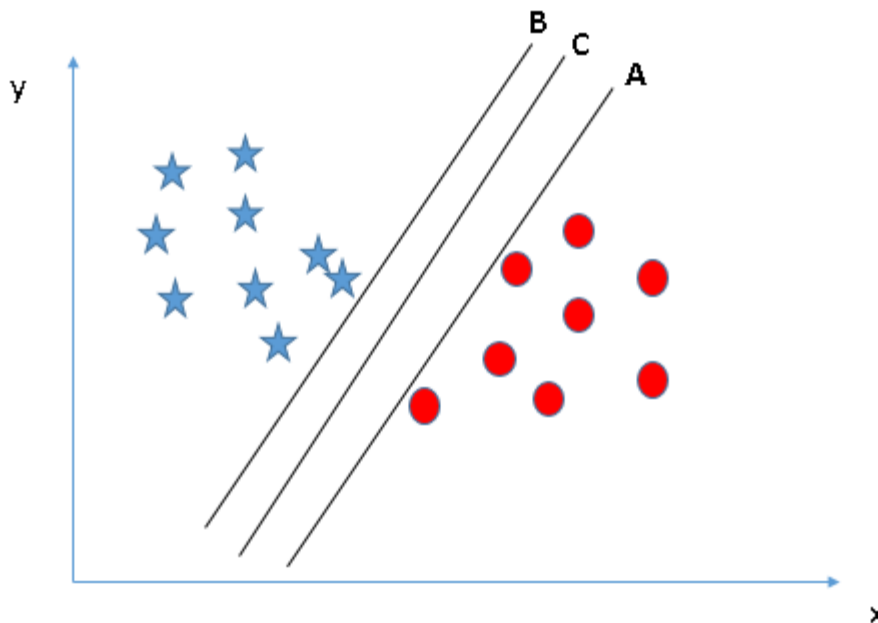
Identify the right hyperplane (Scenario-1)

- Here, we have three hyperplanes (A, B and C). Now, identify the right hyperplane to classify star and circle.
- Hyperplane “B” has excellently performed this job.



Identify the right hyperplane (Scenario-2)

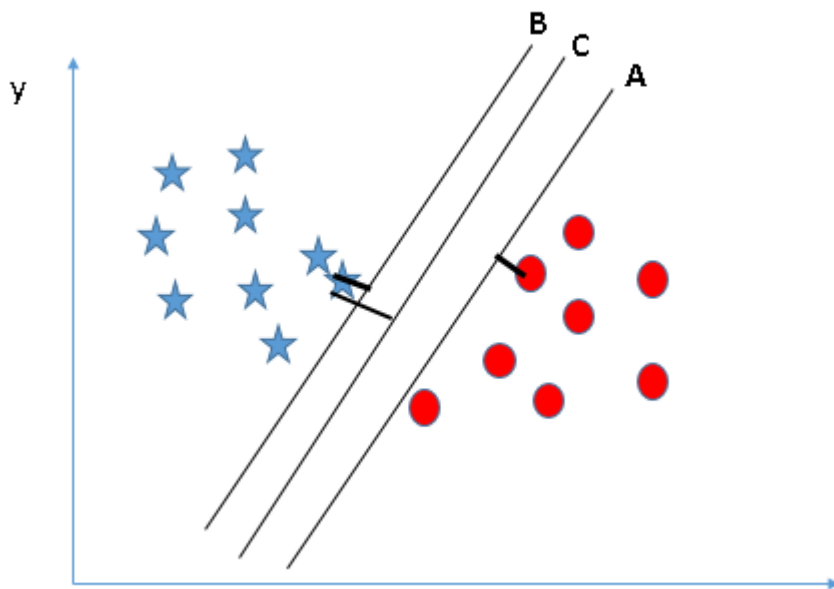
Here, we have three hyperplanes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyperplane?



Here, maximizing the distances between nearest data point (either class) and hyperplane will help us to decide the right hyperplane.

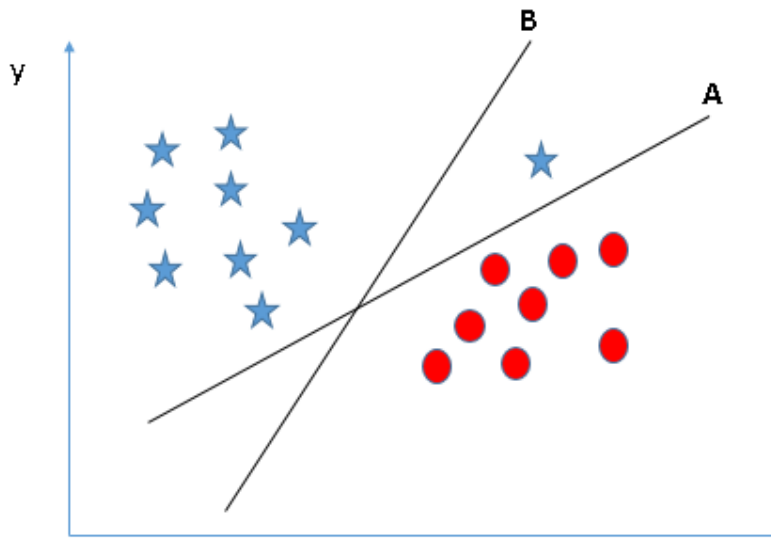
Identify the right hyperplane (Scenario-2)

This distance is called as **Margin**. Let's look at the below snapshot:



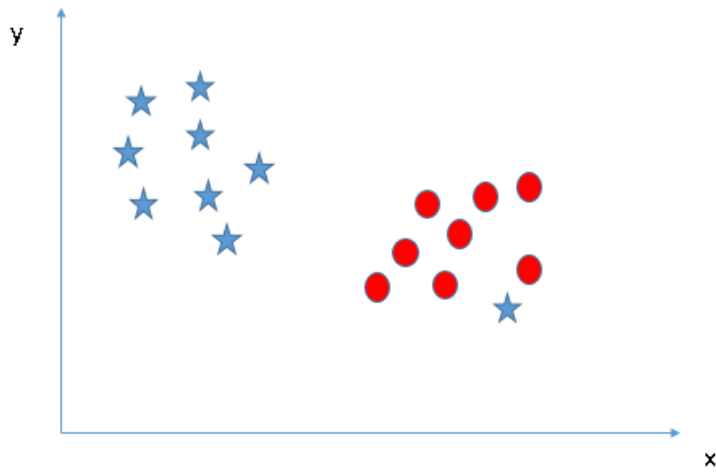
We can see that the margin for hyperplane C is high as compared to both A and B. Hence, we name the right hyperplane as C. Another lightning reason for selecting the hyperplane with higher margin is robustness. If we select a hyperplane having low margin then there is high chance of missclassification.

Identify the right hyperplane (Scenario-3)



Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyperplane which classifies the classes accurately prior to maximizing margin. Here, hyperplane B has a classification error and A has classified all correctly. Therefore, the right hyperplane is **A**.

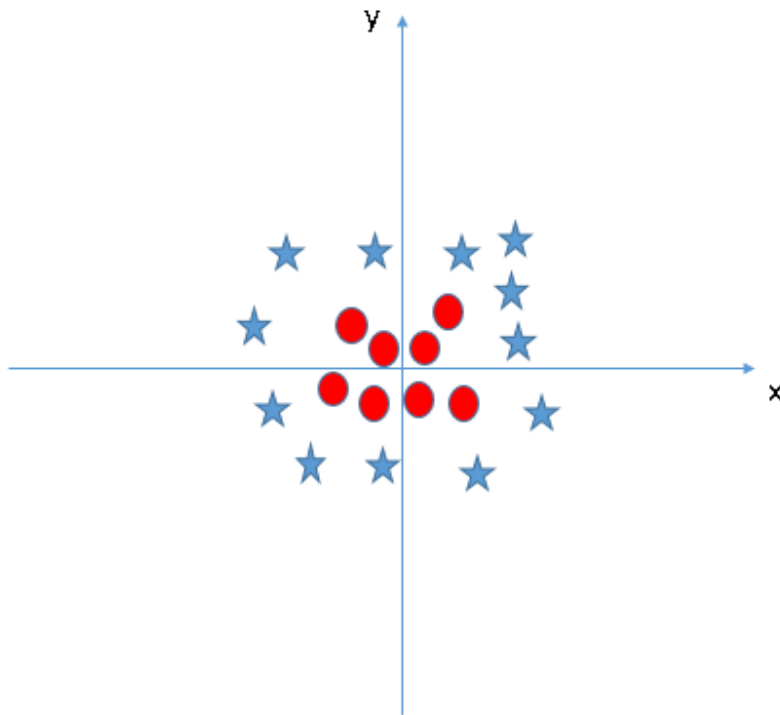
Identify the right hyperplane (Scenario-4)



- We are unable to segregate the two classes using a straight line, as one of star lies in the territory of other (circle) class as an outlier.
- One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyperplane that has maximum margin. Hence, we can say, SVM is robust to outliers.

Identify the right hyperplane (Scenario-5)

- In the scenario below, we can't have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyperplane.



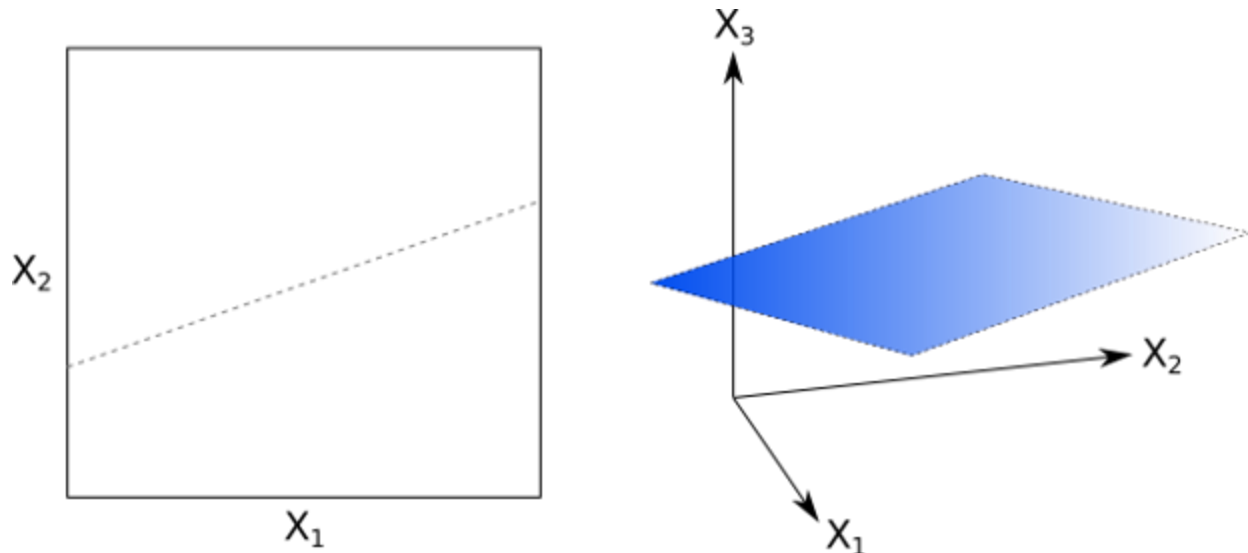
SVM has a technique called the **kernel trick**. These are functions which take low dimensional input space and transform it to a higher dimensional space i.e. it converts a non-separable problem to a separable problem, these functions are called **kernels**. It is mostly useful in non-linear separation problems.

Linear Separating Hyperplanes

- The linear separating hyperplane is the key geometric entity that is at the heart of the SVM. Informally, if we have a high-dimensional feature space, then the linear hyperplane is an object one dimension lower than this space that divides the feature space into two regions.
- If we consider a real-valued p -dimensional feature space, known mathematically as \mathbb{R}^p , then our linear separating hyperplane is an affine $p-1$ dimensional space embedded within it.

Linear Separating Hyperplanes ...

- For the case of $p=2$ this hyperplane is simply a one-dimensional straight line, which lives in the larger two-dimensional plane, whereas for $p=3$ the hyperplane is a two-dimensional plane that lives in the larger three-dimensional feature space.



Support Vector Machine

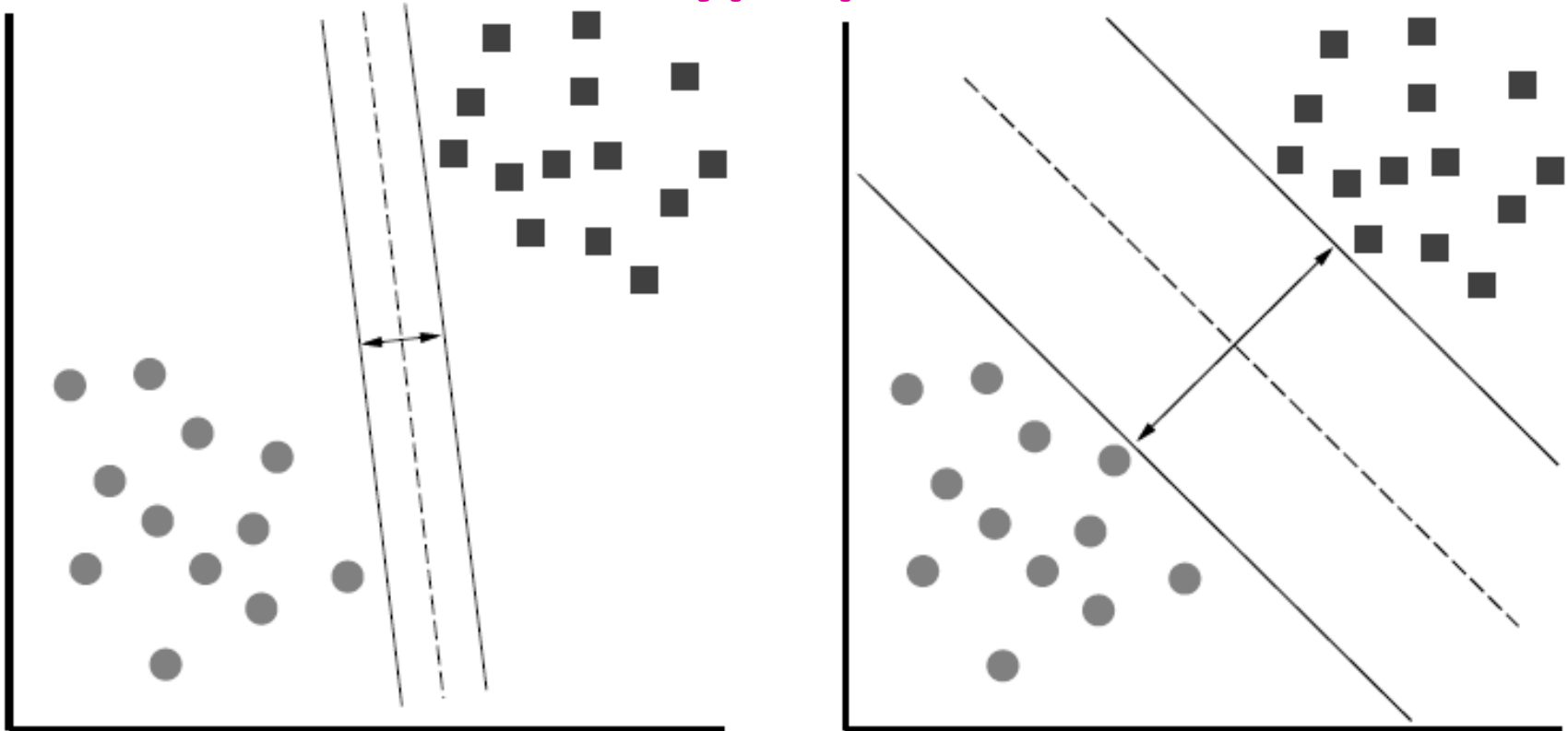


SVM is a training algorithm that maximize the margin between the training patterns and the decision boundary.

The solution is expressed as a linear combination of supporting patterns.

Largest Margin

- **Margin γ :** Distance of closest example from the decision line/hyperplane

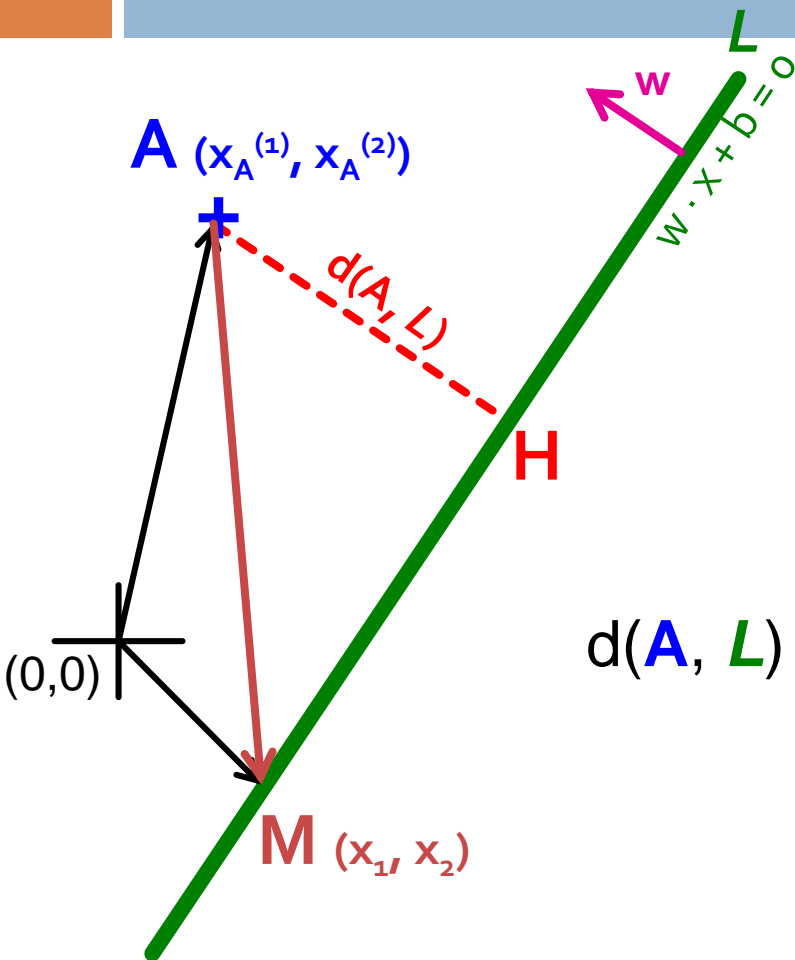


The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.

What is the margin?

Distance from a point to a line

Note we assume
 $\|w\|_2 = 1$



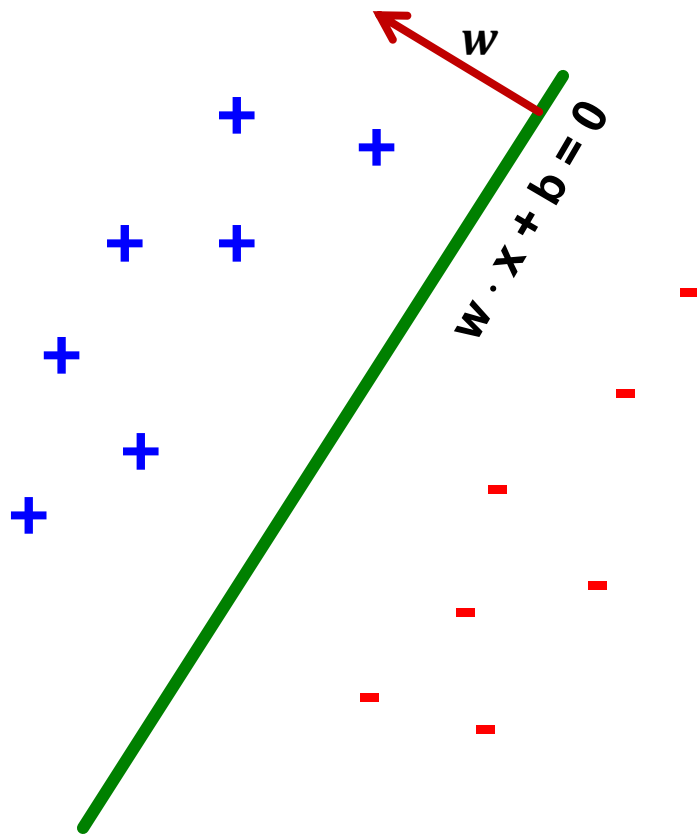
■ Let:

- **Line L:** $w \cdot x + b = 0$
 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + b = 0$
- $w = (w^{(1)}, w^{(2)})$
- **Point A** = $(x_A^{(1)}, x_A^{(2)})$
- **Point M** on a line = $(x_M^{(1)}, x_M^{(2)})$

$$\begin{aligned} d(A, L) &= |AH| \\ &= |(A - M) \cdot w| \\ &= |(x_A^{(1)} - x_M^{(1)}) w^{(1)} + (x_A^{(2)} - x_M^{(2)}) w^{(2)}| \\ &= x_A^{(1)} w^{(1)} + x_A^{(2)} w^{(2)} + b \\ &= w \cdot A + b \end{aligned}$$

Remember $x_M^{(1)} w^{(1)} + x_M^{(2)} w^{(2)} = -b$
since **M** belongs to line **L**

Largest Margin



- Prediction = $\text{sign}(w \cdot x + b)$
- “**Confidence**” = $(w \cdot x + b) y$
- For i -th datapoint:
$$\gamma_i = (w \cdot x_i + b) y_i$$
- Want to solve:
$$\max_w \min_i \gamma_i$$
- Can rewrite as

$$\max_{w, \gamma} \gamma$$

$$s.t. \forall i, y_i (w \cdot x_i + b) \geq \gamma$$

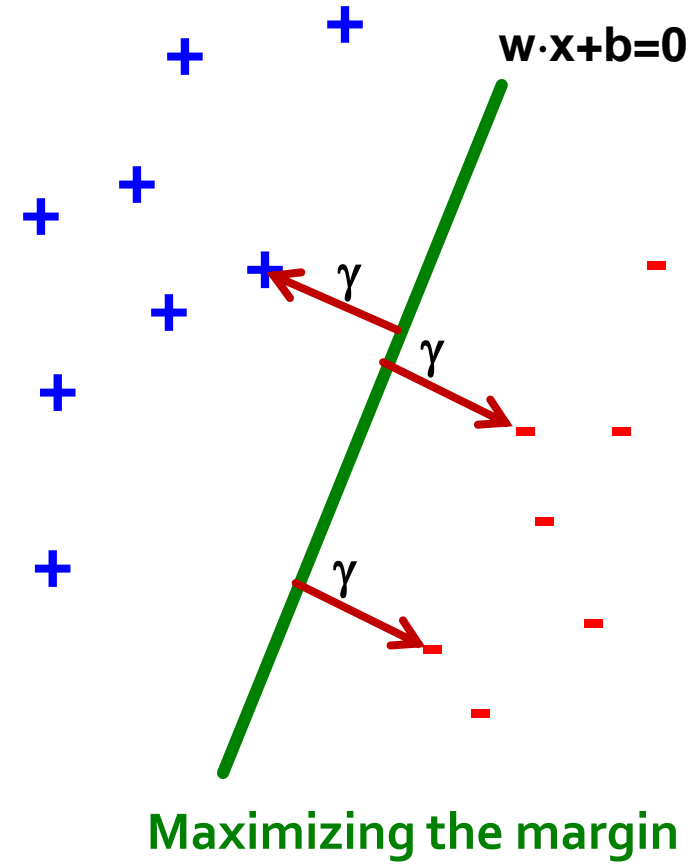
Support Vector Machine

- **Maximize the margin:**
 - Good according to intuition, theory (VC dimension) & practice

$$\max_{w, \gamma} \gamma$$

$$s.t. \forall i, y_i (w \cdot x_i + b) \geq \gamma$$

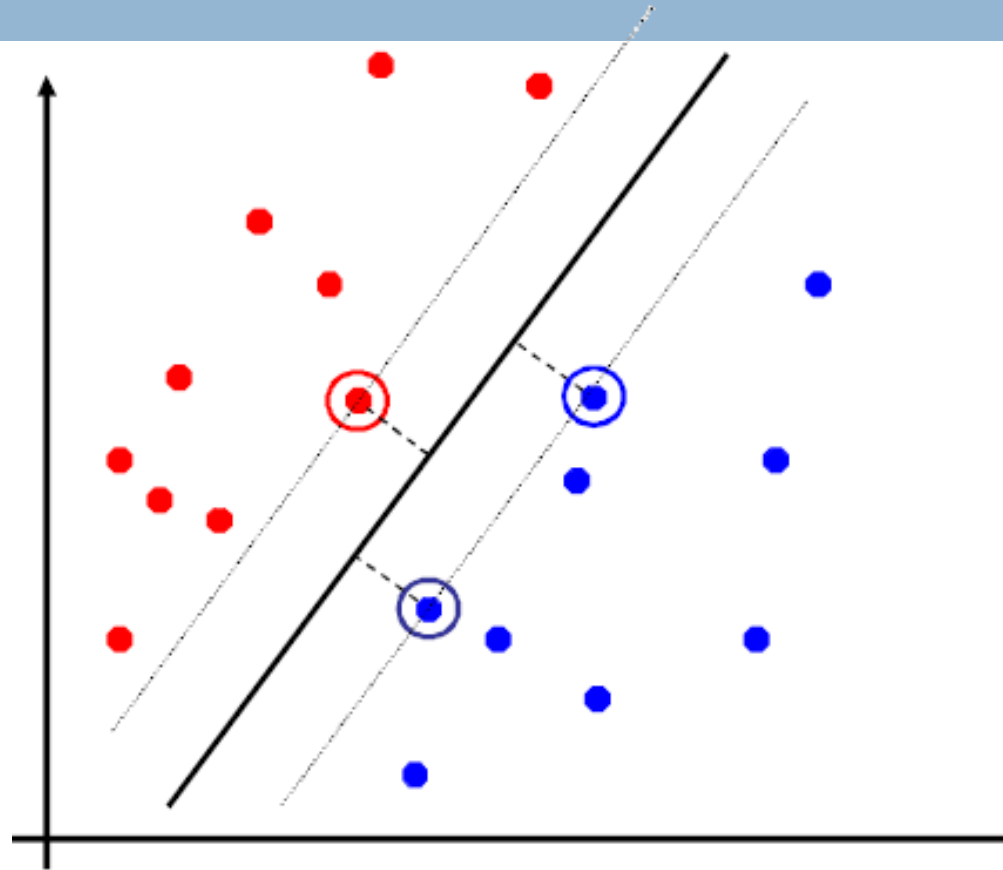
- γ is margin ... distance from the separating hyperplane



Support Vector Machine

- **Separating hyperplane**
is defined by the
support vectors

- Points on \pm planes from the solution
- If you knew these points, you could ignore the rest
- Generally,
 $d+1$ support vectors (for d dim. data)



Canonical Hyperplane: Problem

■ Problem:

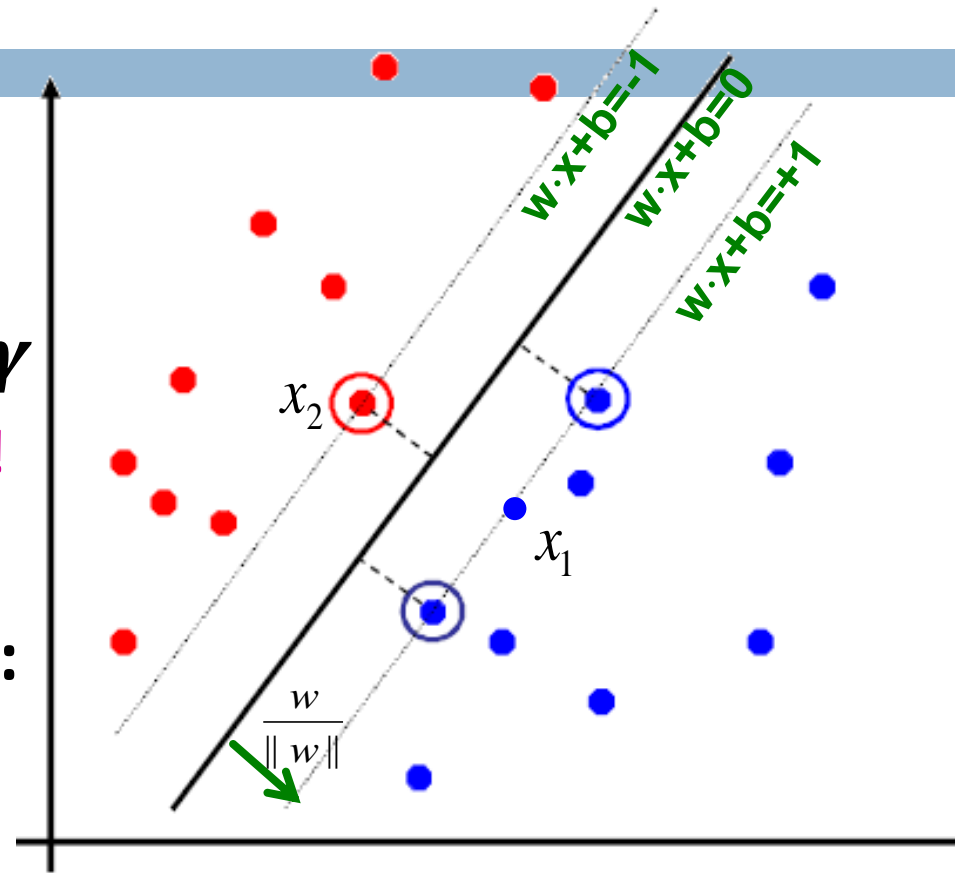
- Let $(w \cdot x + b)y = \gamma$
then $(2w \cdot x + 2b)y = 2\gamma$
 - Scaling w increases margin!

■ Solution:

- Work with normalized w :

$$\gamma = \left(\frac{w}{\|w\|} \cdot x + b \right) y$$

- Let's also require **support vectors** x_j
to be on the plane defined by: $w \cdot$
 $x_j + b = \pm 1$



$$\|w\| = \sqrt{\sum_{j=1}^d (w^{(j)})^2}$$

Canonical Hyperplane: Solution

■ Want to maximize margin γ !

■ What is the relation between x_1 and x_2 ?

■ $x_1 = x_2 + 2\gamma \frac{w}{\|w\|}$

■ We also know:

■ $w \cdot x_1 + b = +1$

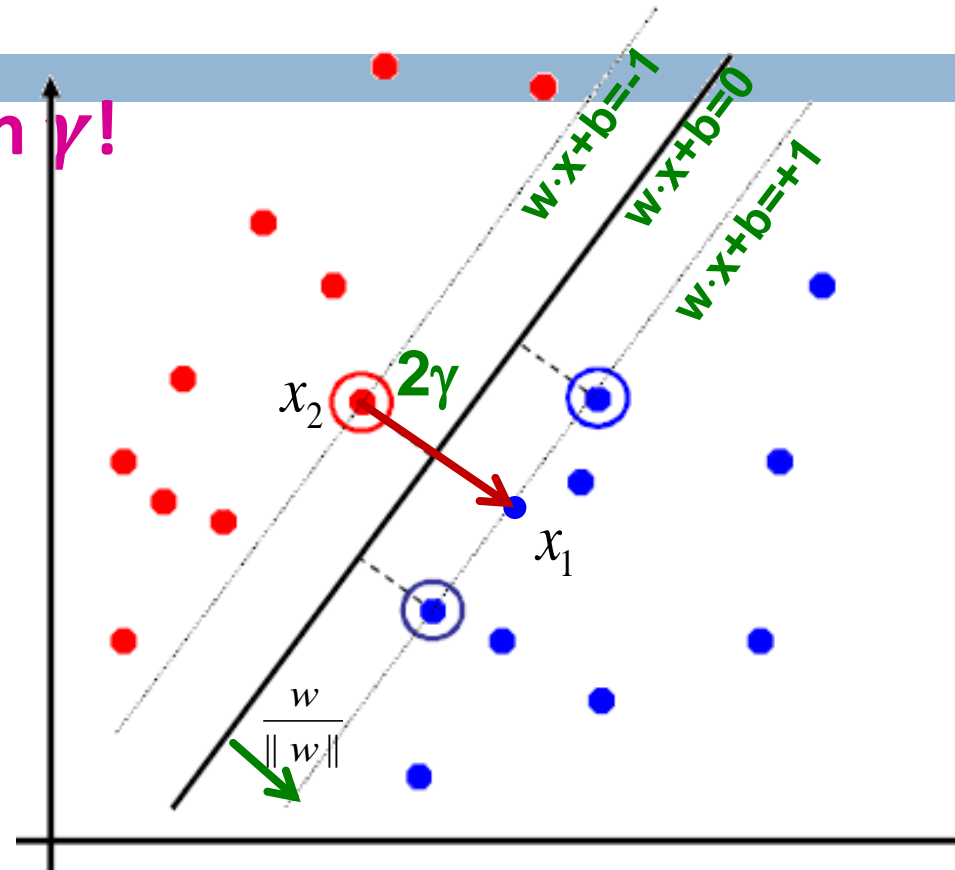
■ $w \cdot x_2 + b = -1$

■ So:

■ $w \cdot x_1 + b = +1$

■ $w \left(x_2 + 2\gamma \frac{w}{\|w\|} \right) + b = +1$

■ $\underbrace{w \cdot x_2 + b}_{-1} + 2\gamma \frac{w \cdot w}{\|w\|} = +1$



$$\Rightarrow \gamma = \frac{\|w\|}{w \cdot w} = \frac{1}{\|w\|}$$

Note:

$$w \cdot w = \|w\|^2$$

Maximizing the Margin

- We started with

$$\max_{w, \gamma} \gamma$$

$$s.t. \forall i, y_i (w \cdot x_i + b) \geq \gamma$$

But w can be arbitrarily large!

- We normalized and...

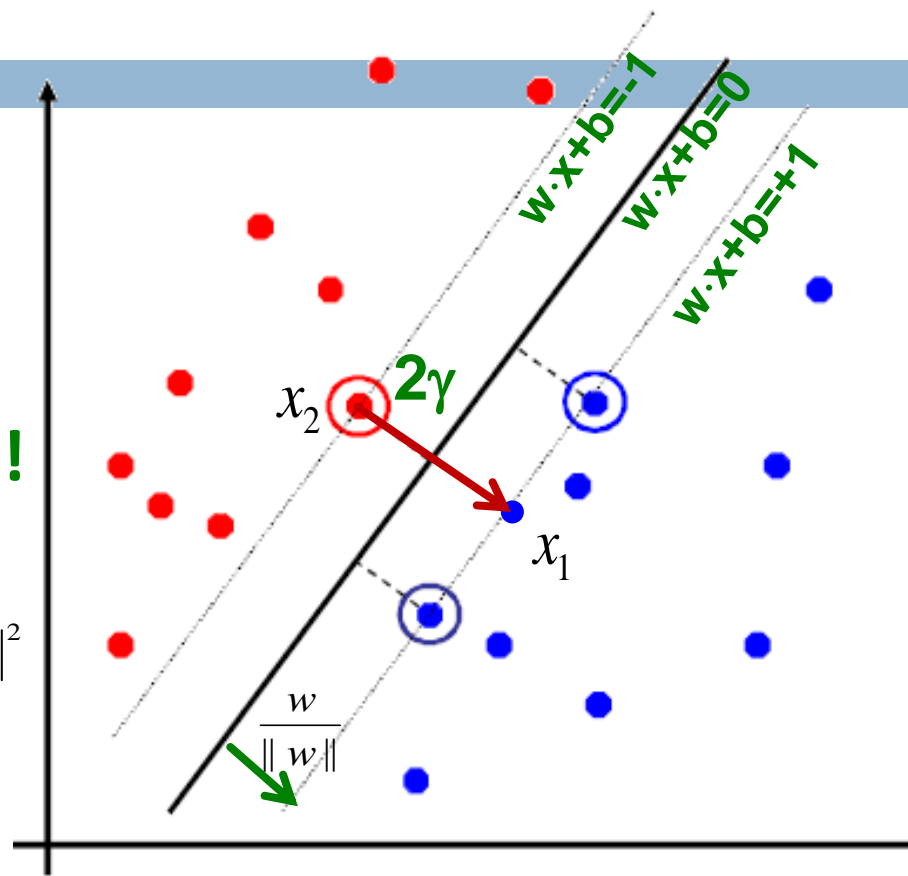
$$\arg \max \gamma = \arg \max \frac{1}{\|w\|} = \arg \min \|w\| = \arg \min \frac{1}{2} \|w\|^2$$

- Then:

$$\min_w \frac{1}{2} \|w\|^2$$

$$s.t. \forall i, y_i (w \cdot x_i + b) \geq 1$$

This is called SVM with “hard” constraints



The Optimization Problem

- Then we can formulate the *quadratic optimization problem*:

Find \mathbf{w} and b such that

$$r = \frac{2}{\|\mathbf{w}\|} \text{ is maximized; and for all } \{(\mathbf{x}_i, y_i)\}$$
$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1; \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1$$

- A better formulation ($\min \|\mathbf{w}\| = \max 1 / \|\mathbf{w}\|$):

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Lagrangian of Original Problem

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to } 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0 \quad \text{for } i = 1, \dots, n \end{aligned}$$

The Lagrangian is

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

Lagrangian multipliers

– Note that $\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w}$

Setting the **gradient of \mathcal{L} w.r.t. \mathbf{w} and b to zero**, we have

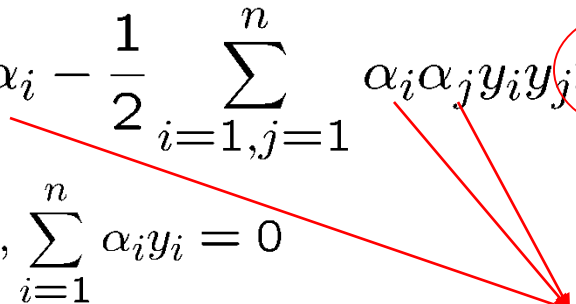
$$\mathbf{w} + \sum_{i=1}^n \alpha_i (-y_i) \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

The Dual Optimization Problem

We can transform the problem to its dual

$$\begin{aligned} \max. \quad W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to } \alpha_i &\geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$


Dot product of \mathbf{X}

α 's \rightarrow New variables
(Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

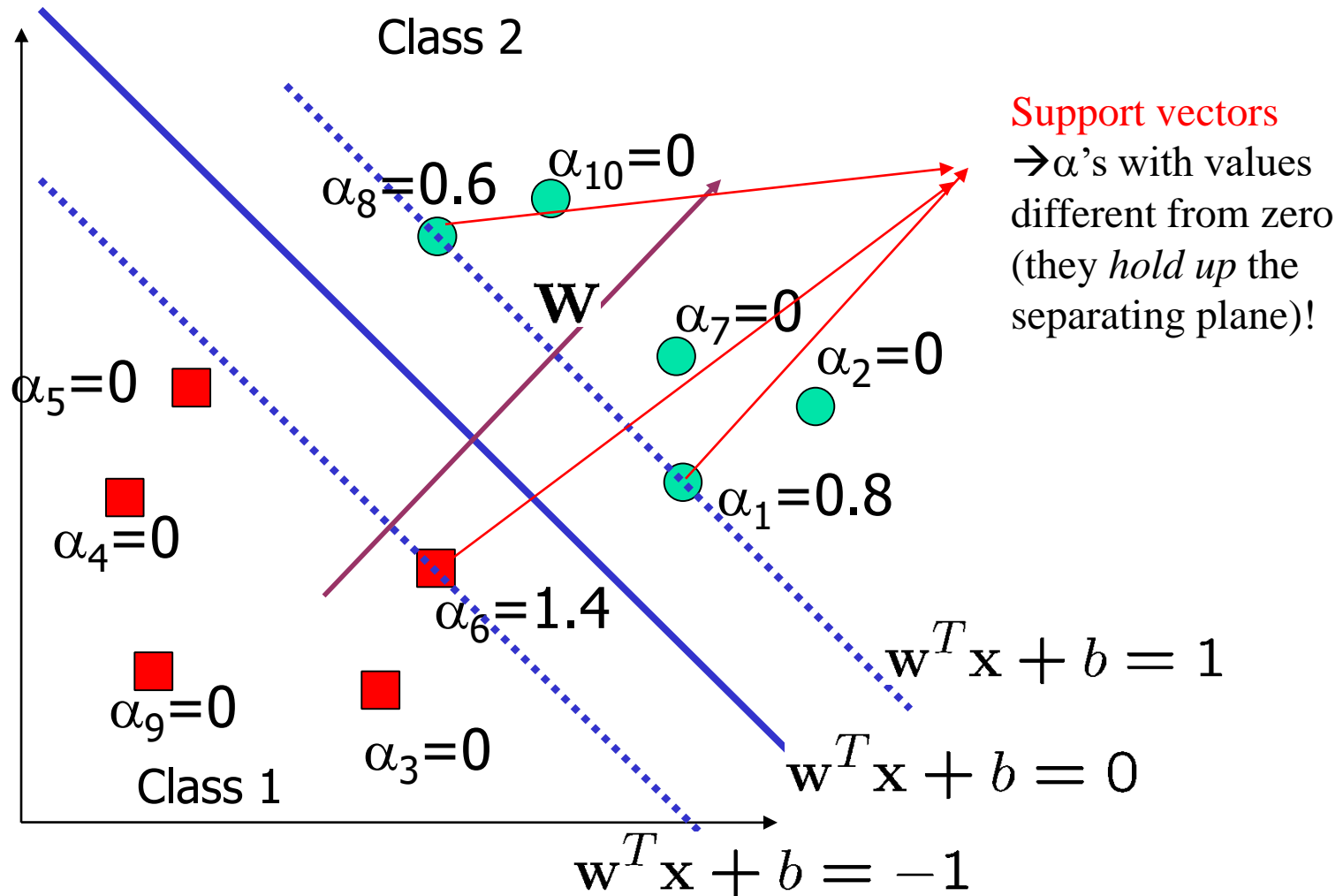
– Global maximum of α_i can always be found

\rightarrow well established tools for solving this optimization problem (e.g. cplex)

Note:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

A Geometrical Interpretation





Any Questions ?