

# SVM

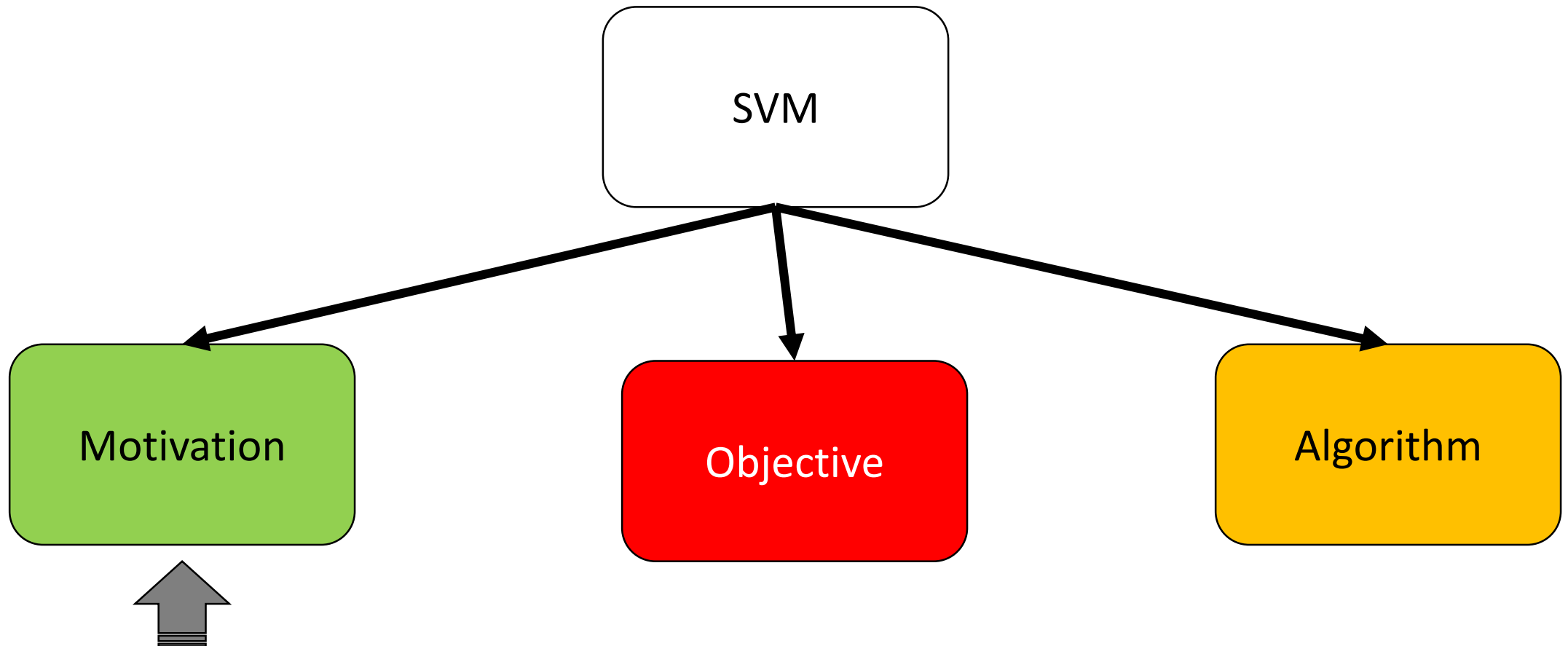
# Machine Learning

Dr. Amani RAAD

[amani.raad@ul.edu.lb](mailto:amani.raad@ul.edu.lb)

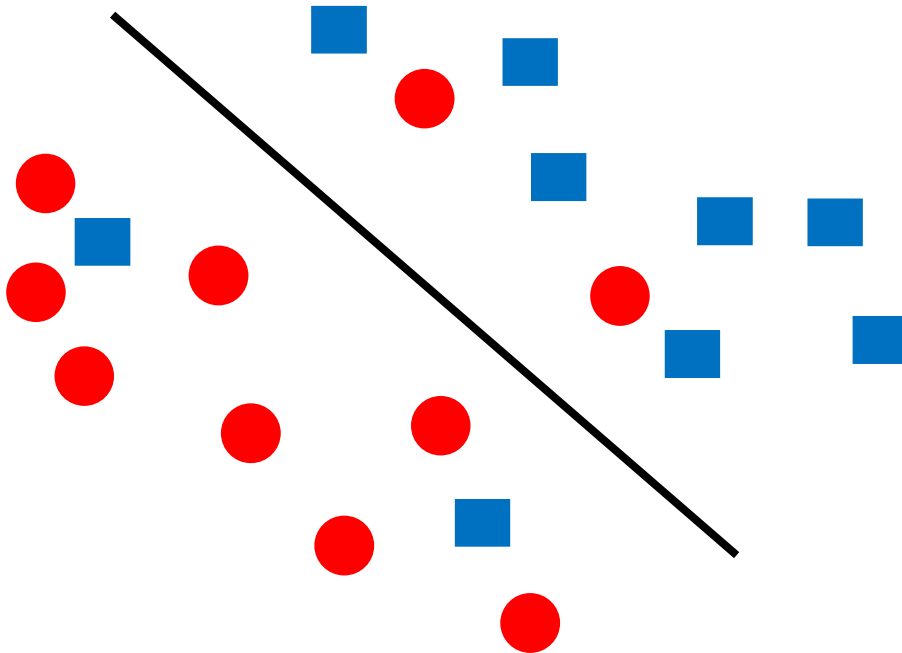
2022-2023

# Outline



# Problems of Perceptron

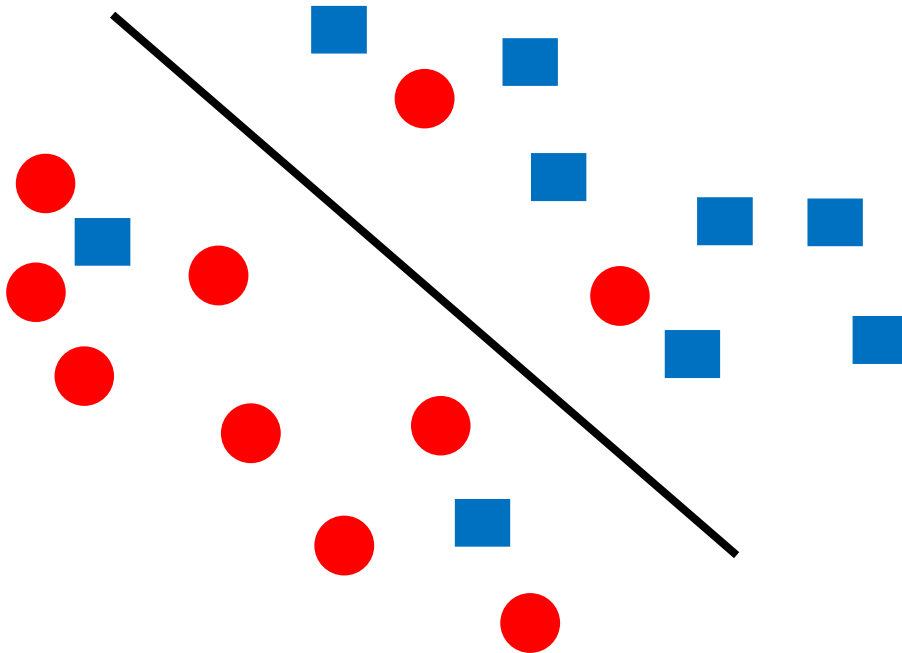
- Perceptrons exhibit various limitations in their ability to classify data
  - The biggest problem is when data is not linearly separable (*problem 1*)



**In this case, any line through the examples will yield examples of both classes on at least one of the sides!**

# Problems of Perceptron

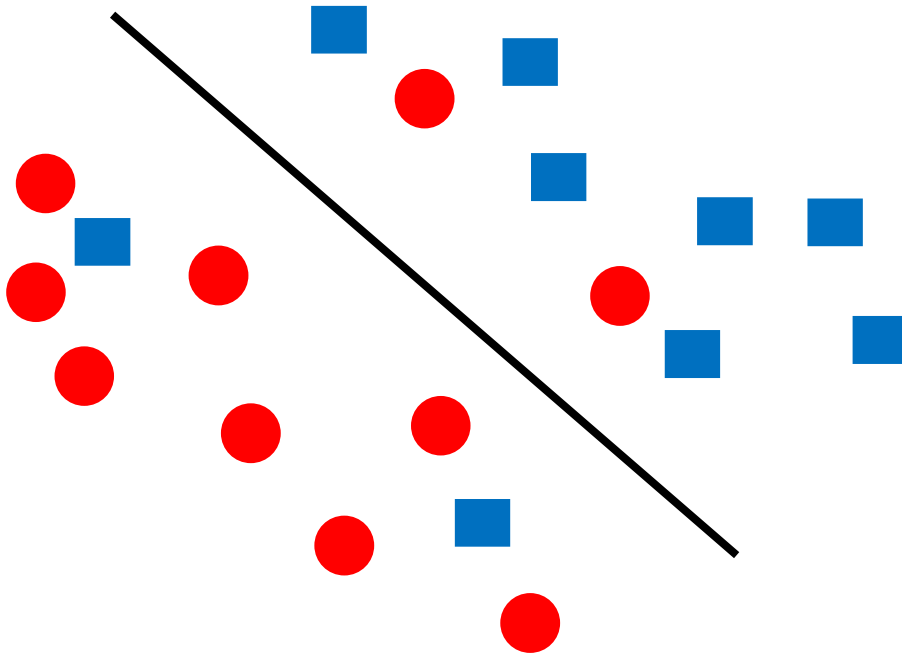
- Perceptrons exhibit various limitations in their ability to classify data
  - The biggest problem is when data is not linearly separable (*problem 1*)



**In principle, it is possible to transform the examples into another space, which makes them linearly separable**

# Problems of Perceptron

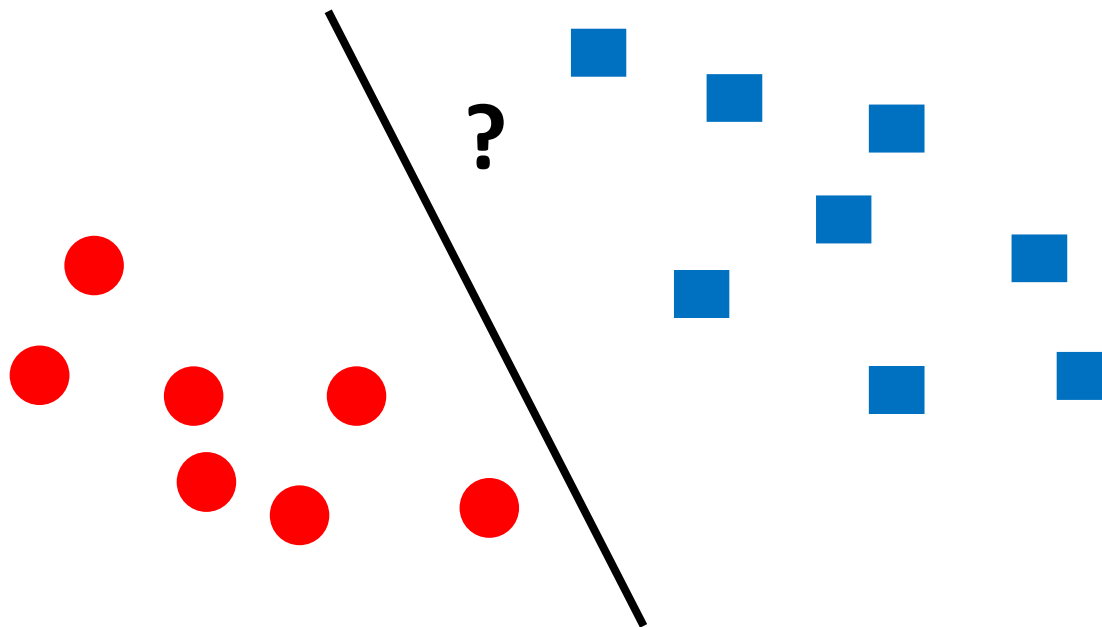
- Perceptrons exhibit various limitations in their ability to classify data
  - The biggest problem is when data is not linearly separable (*problem 1*)



However, doing so could lead to **overfitting** (*the situation where the classifier works well on training data but not on new data*)

# Problems of Perceptron

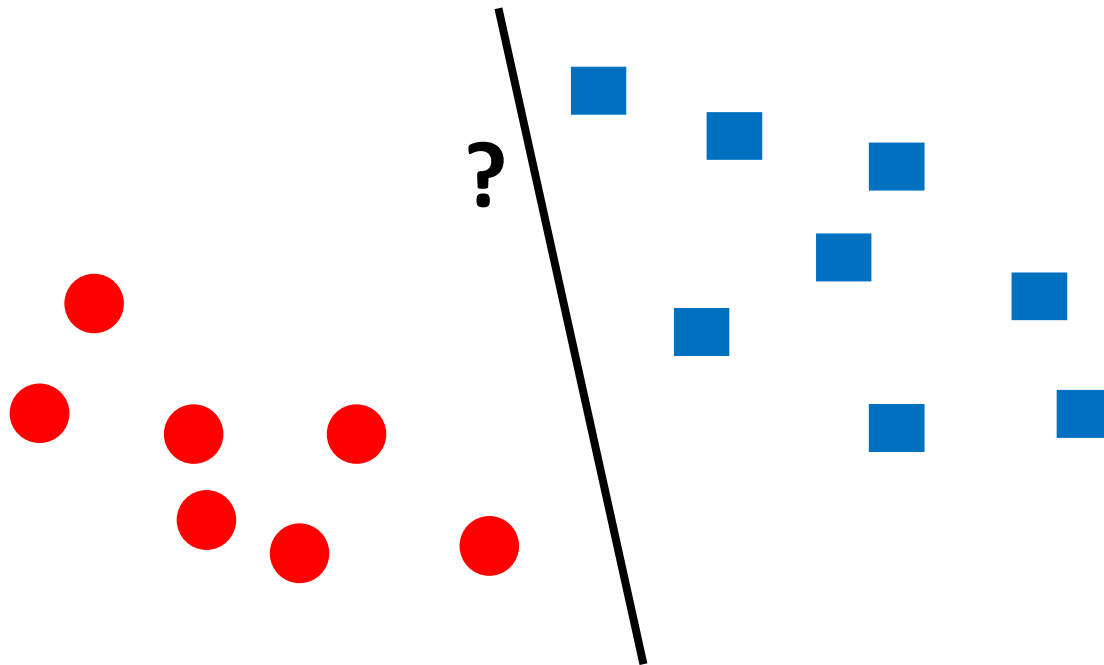
- Perceptrons exhibit various limitations in their ability to classify data
  - And even if the space is (or is made) linearly separable, there could be many hyperplanes, which are not all equally good (*problem 2*)



**An acceptable hyperplane and the new example indicated by “?” will be classified as a **square****

# Problems of Perceptron

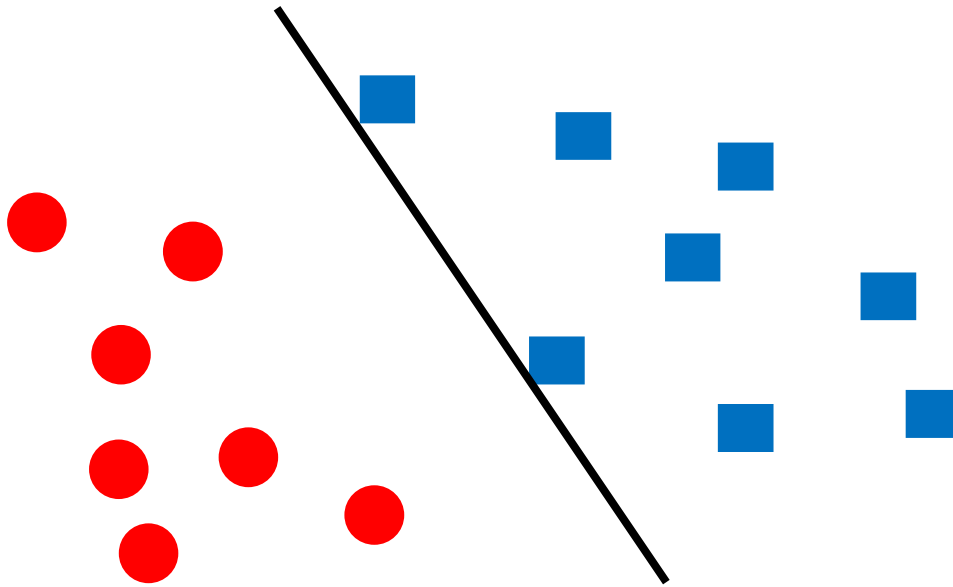
- Perceptrons exhibit various limitations in their ability to classify data
  - And even if the space is (or is made) linearly separable, there could be many hyperplanes, which are not all equally good (*problem 2*)



**Another acceptable hyperplane,**  
but the new example will now be  
classified as a **circle** (*although it  
seems closer to squares!*)

# Problems of Perceptron

- Perceptrons exhibit various limitations in their ability to classify data
  - Yet, another problem is that perceptrons usually stop as soon as there are no misclassified examples (*problem 3*)

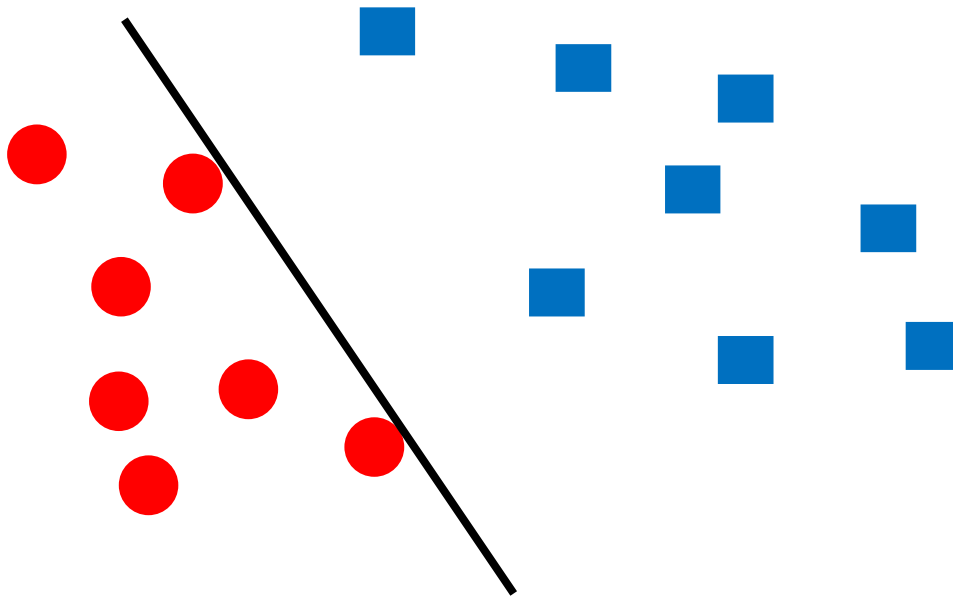


This hyperplane *just* managed to accommodate the two **squares** it touches before stopped



# Problems of Perceptron

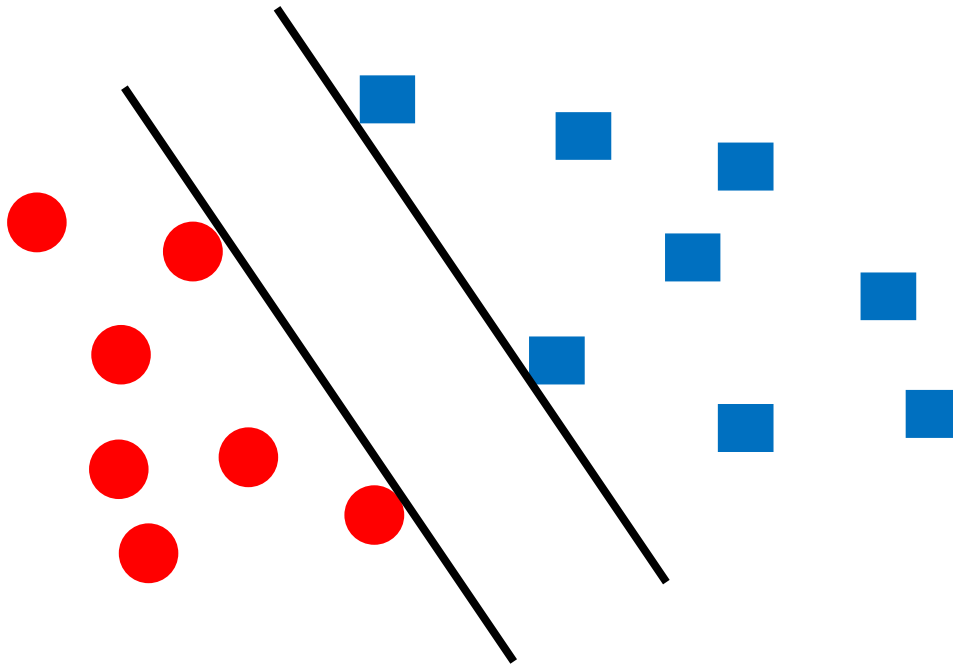
- Perceptrons exhibit various limitations in their ability to classify data
  - Yet, another problem is that perceptrons usually stop as soon as there are no misclassified examples (*problem 3*)



This hyperplane also *just* managed to accommodate the two **circles** it touches before stopped

# Problems of Perceptron

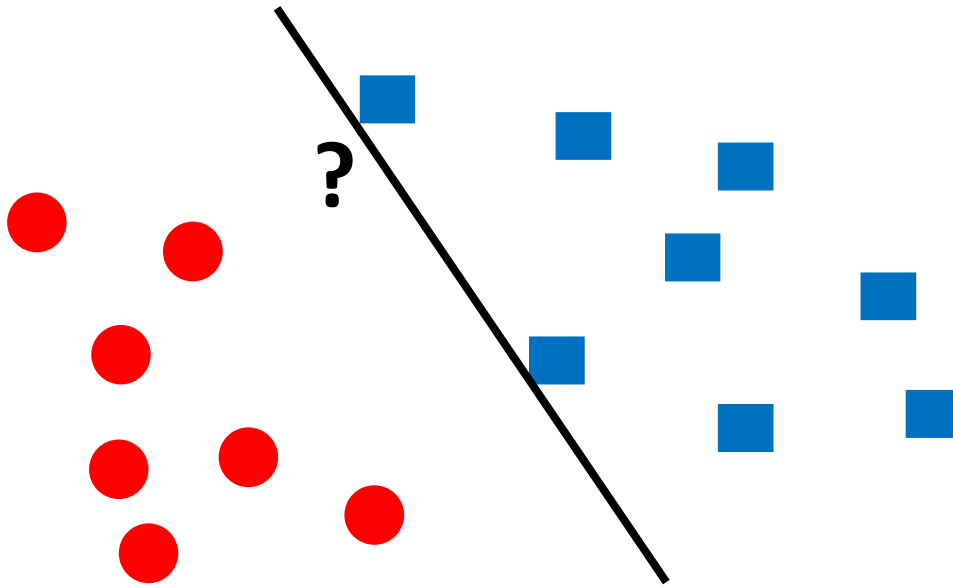
- Perceptrons exhibit various limitations in their ability to classify data
  - Yet, another problem is that perceptrons usually stop as soon as there are no misclassified examples (*problem 3*)



**If either of these hyperplanes represents the final weight vector, the weights will be biased toward one of the classes**

# Problems of Perceptron

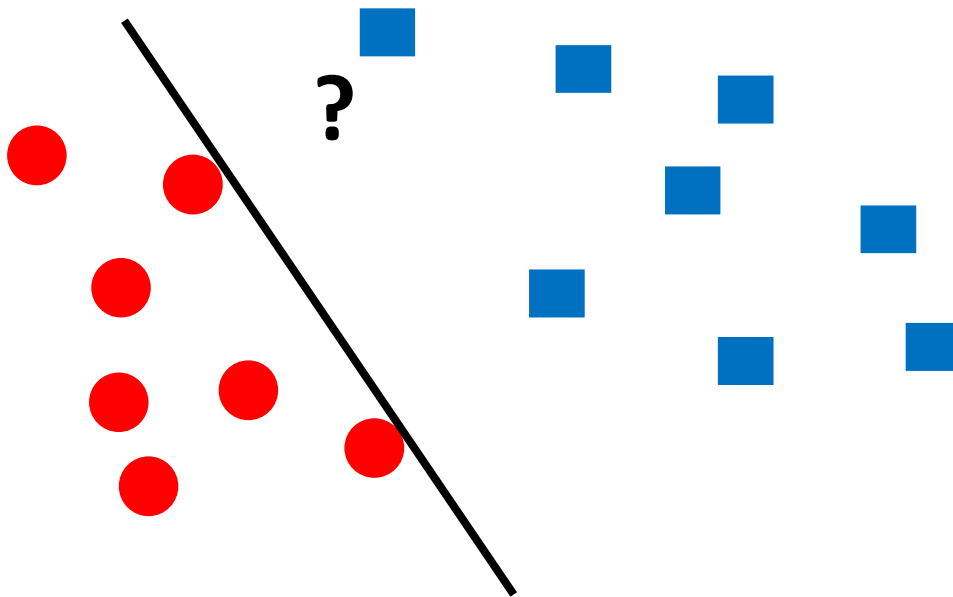
- Perceptrons exhibit various limitations in their ability to classify data
  - Yet, another problem is that perceptrons usually stop as soon as there are no misclassified examples (*problem 3*)



For example, if this hyperplane is the one that the perceptron chooses, the example indicated by “?” will be classified as a **circle**

# Problems of Perceptron

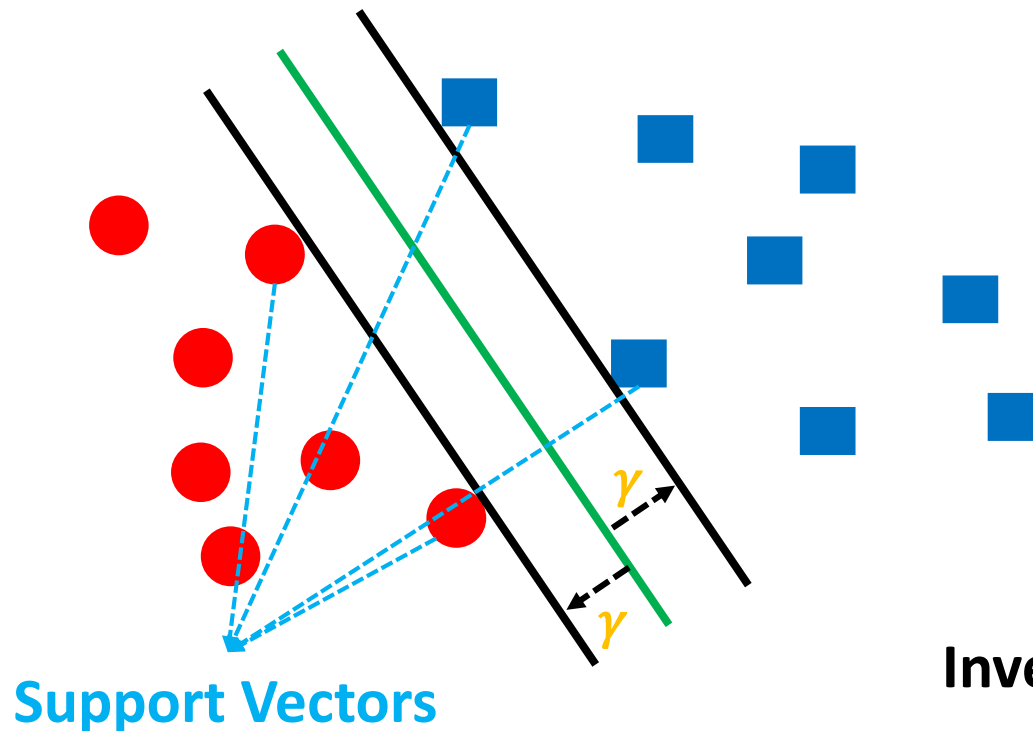
- Perceptrons exhibit various limitations in their ability to classify data
  - Yet, another problem is that perceptrons usually stop as soon as there are no misclassified examples (*problem 3*)



**However, if this hyperplane is the one that the perceptron selects, the example indicated by “?” will be classified as a **square**!**

# Support Vector Machines

- A **Support Vector Machine (SVM)** is an improvement over a perceptron, whereby it addresses the three aforementioned problems



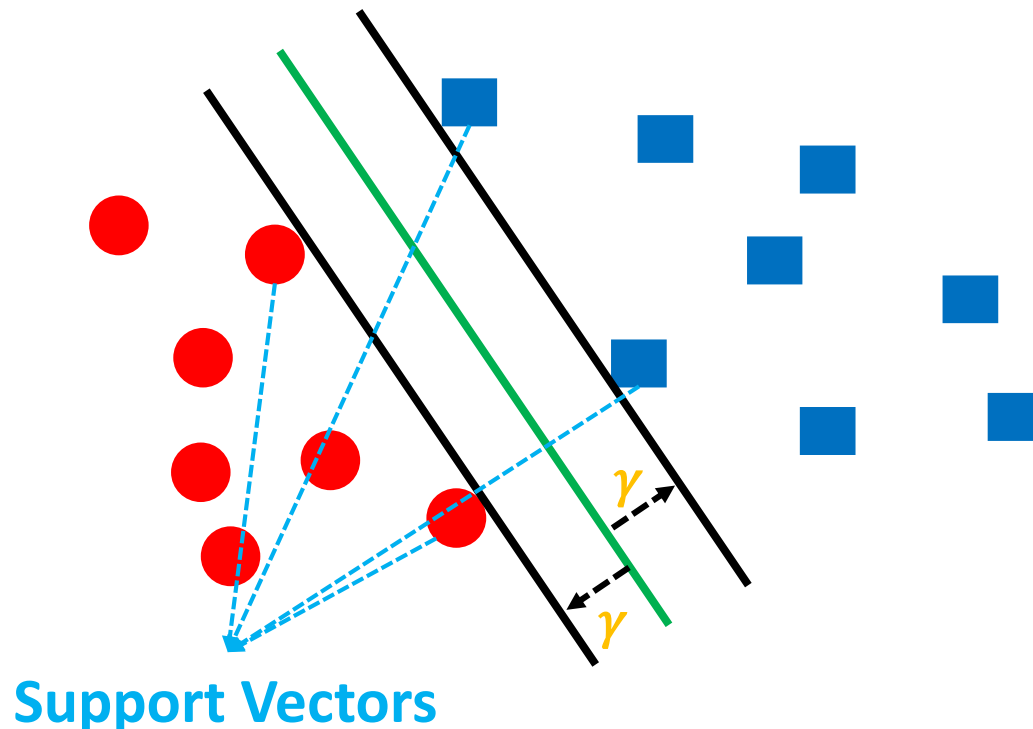
Invented by the professor **Vladimir Vapnik**

# Support Vector Machines

- SVM is related to statistical learning theory
- SVM was first introduced in 1992
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
- SVM is now regarded as an important example of “kernel methods”, one of the [key area in machine learning](#)

# Support Vector Machines

- A **Support Vector Machine (SVM)** is an improvement over a perceptron, whereby it addresses the three aforementioned problems



An SVM selects one particular hyperplane (**the green line in the figure**) that not only separates the examples into two classes, but does so in a way that maximizes the margin ( $\gamma$  in the figure), which is the distance between the hyperplane and the closest examples of the training set

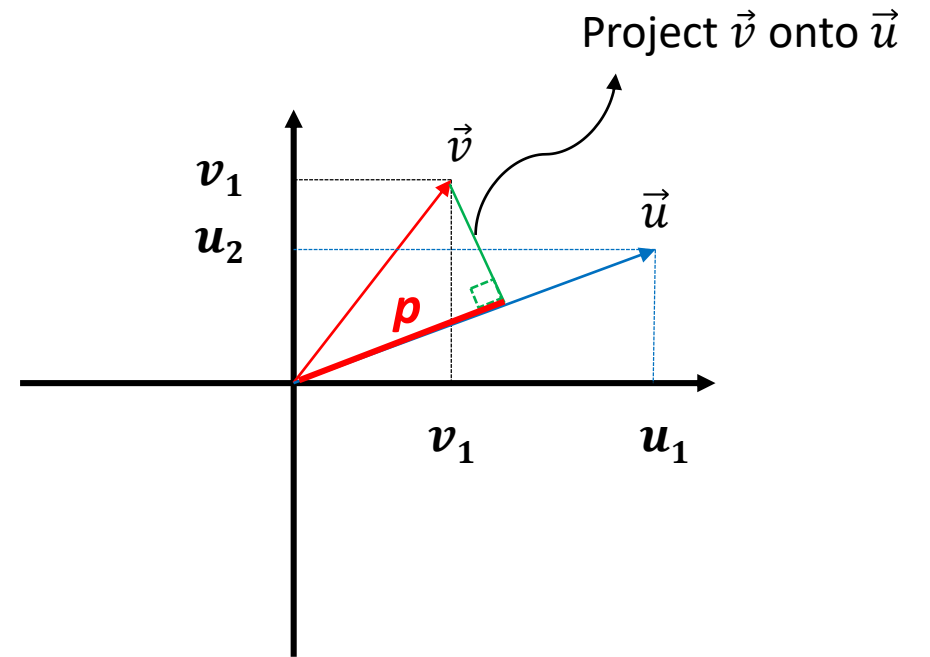
# Vector Inner Product

- Assume the following two vectors:

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- What is the inner product of  $\vec{u}$  and  $\vec{v}$ ?

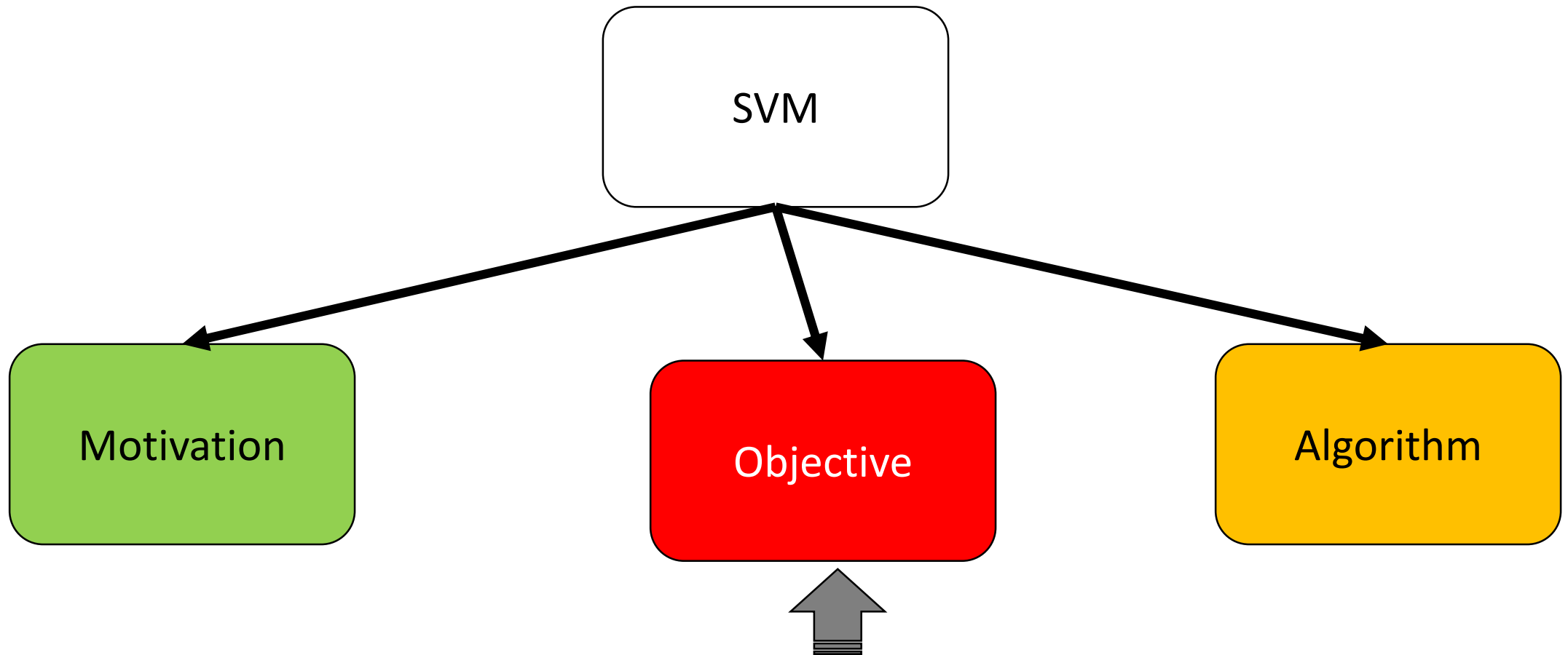
$$\vec{u}^T \vec{v} = ?$$



$p$  is the length of the projection of  $\vec{v}$  onto  $\vec{u}$

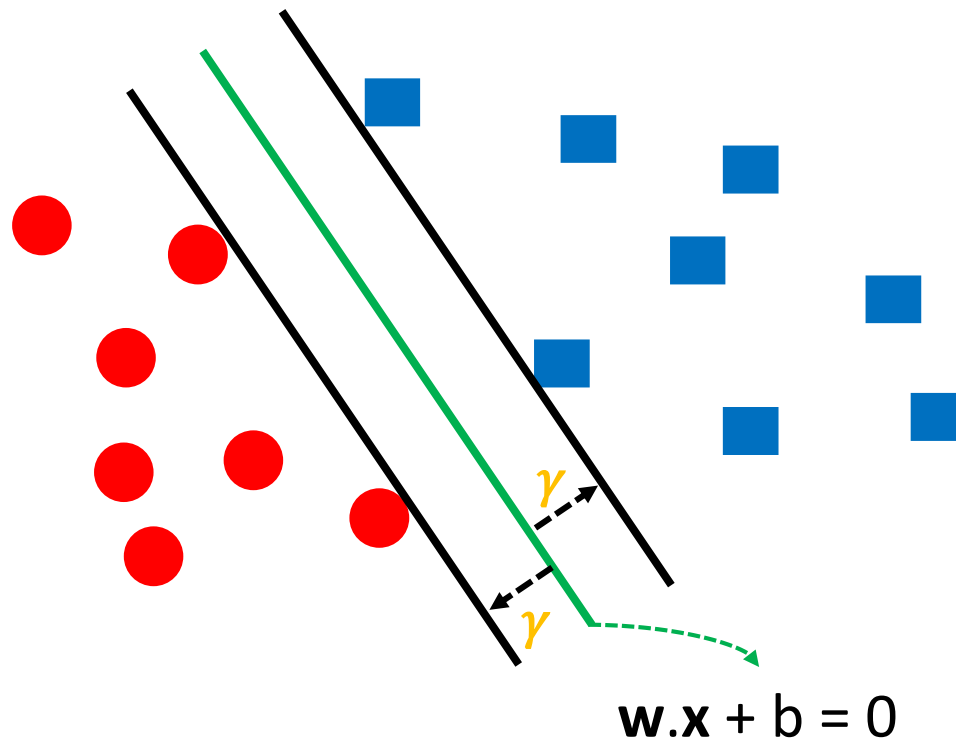


# Outline



# What is the Objective of SVM?

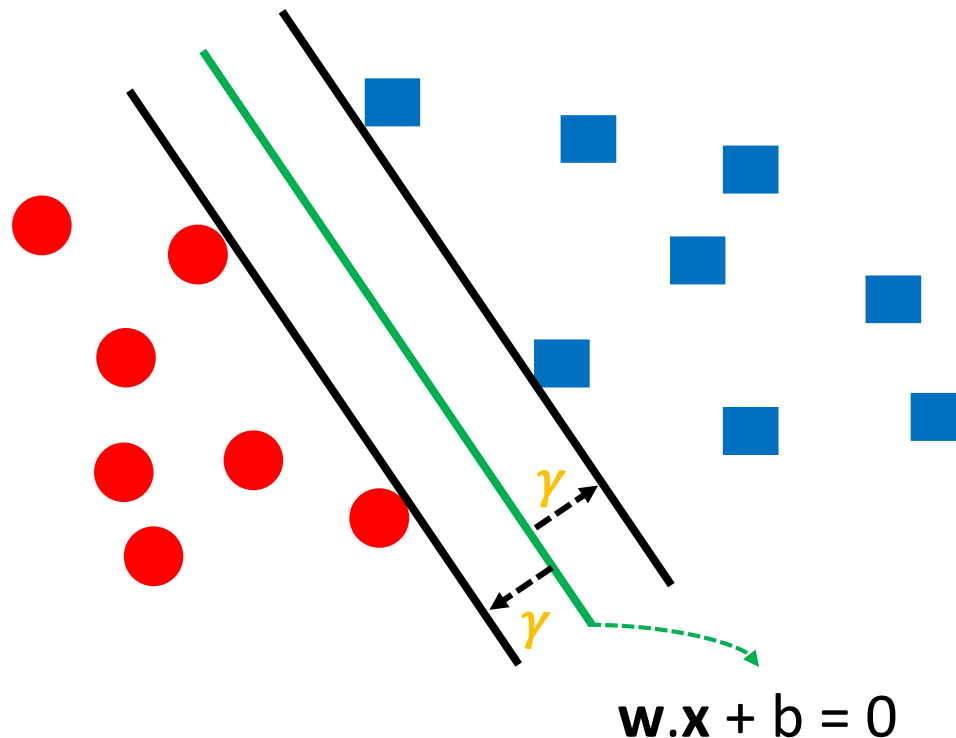
- The objective of an SVM is to select a hyperplane  $\mathbf{w} \cdot \mathbf{x} + b = 0$  that maximizes the distance,  $\gamma$ , between the hyperplane and any example in the training set



**Intuitively, we are more certain of the class of examples that are far from the separating hyperplane than we are of examples near to that hyperplane**

# What is the Objective of SVM?

- The objective of an SVM is to select a hyperplane  $\mathbf{w} \cdot \mathbf{x} + b = 0$  that maximizes the distance,  $\gamma$ , between the hyperplane and any example in the training set



**Thus, it is desirable that all the training examples be as far from the hyperplane as possible  
(*but on the correct side of that hyperplane, of course!*)**

# What is the Objective of SVM?

- More formally, the goal of any SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , maximize  $\gamma$  (by varying  $\mathbf{w}$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$y^i(\mathbf{w} \cdot \mathbf{x}^i + b) \geq \gamma$$

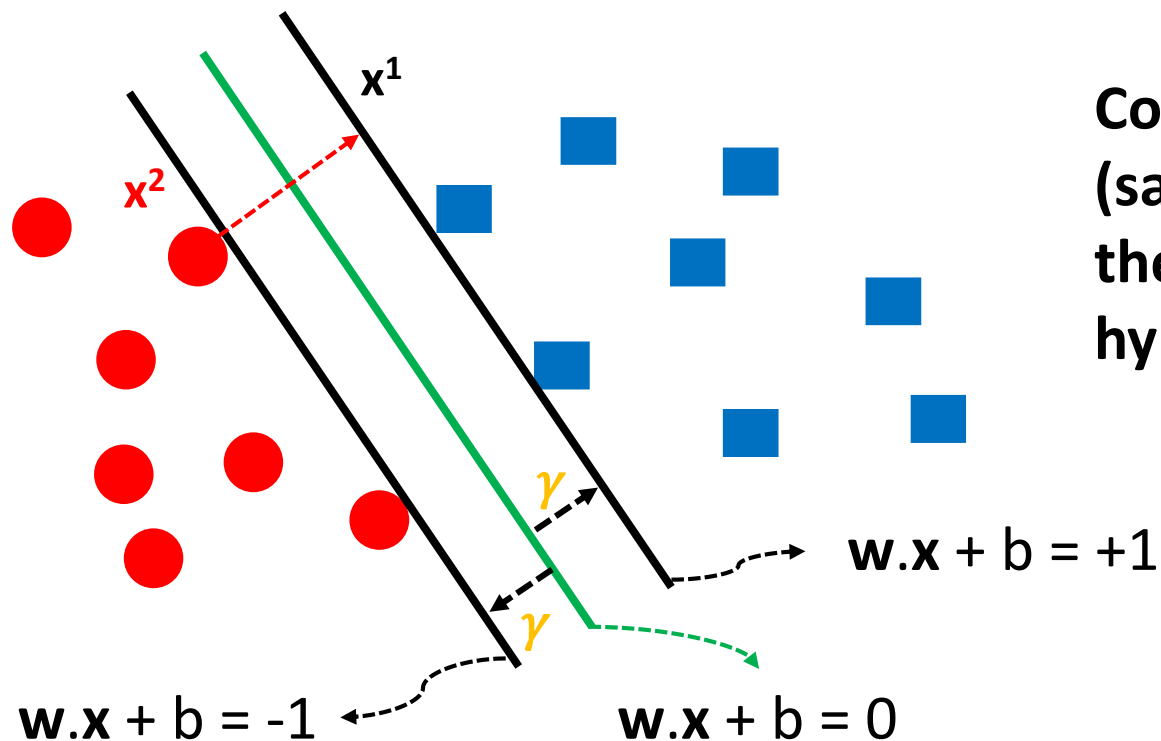
Notice that  $y^i$ , which must be +1 or -1, determines which side of the hyperplane the point  $x^i$  must be on, so the  $\geq$  relationship to  $\gamma$  is always correct!

# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector
  - Consequently, the parallel hyperplanes that just touch the support vectors can be described by the equations  $w \cdot x + b = +1$  and  $w \cdot x + b = -1$

# The Objective of SVM

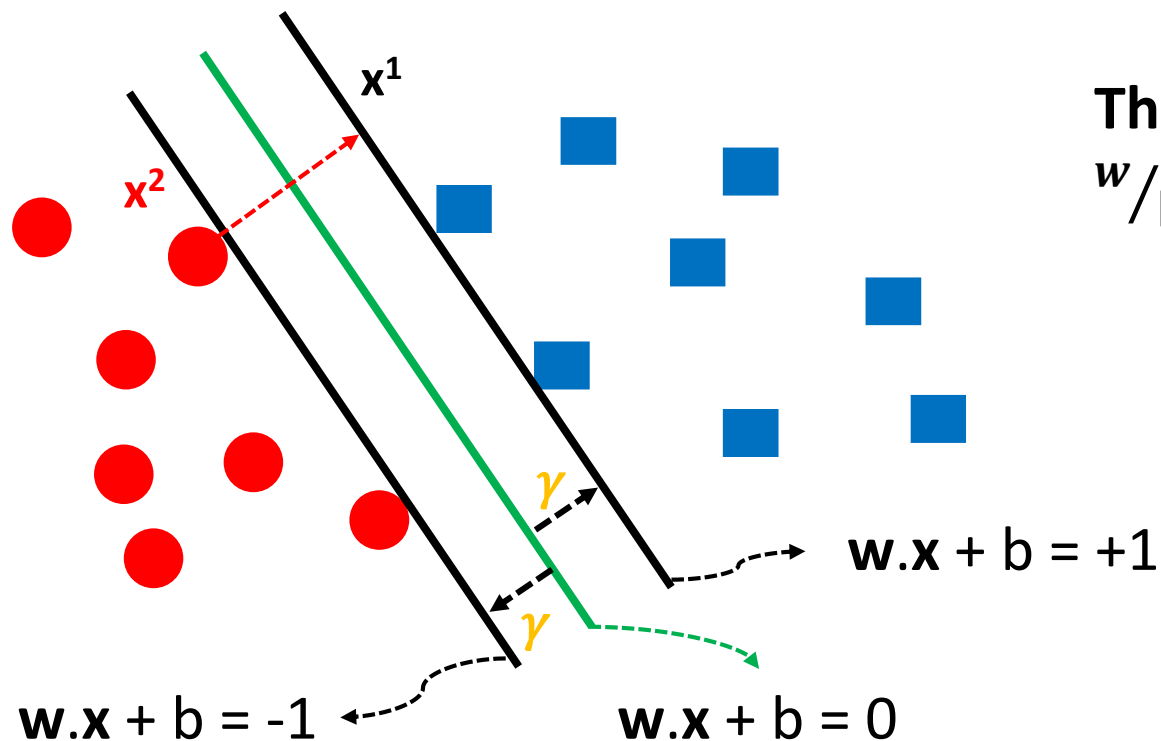
- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector



Consider one of the support vectors, (say,  $x^2$ , in the figure) and let  $x^1$  be the projection of  $x^2$  to the upper hyperplane

# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector

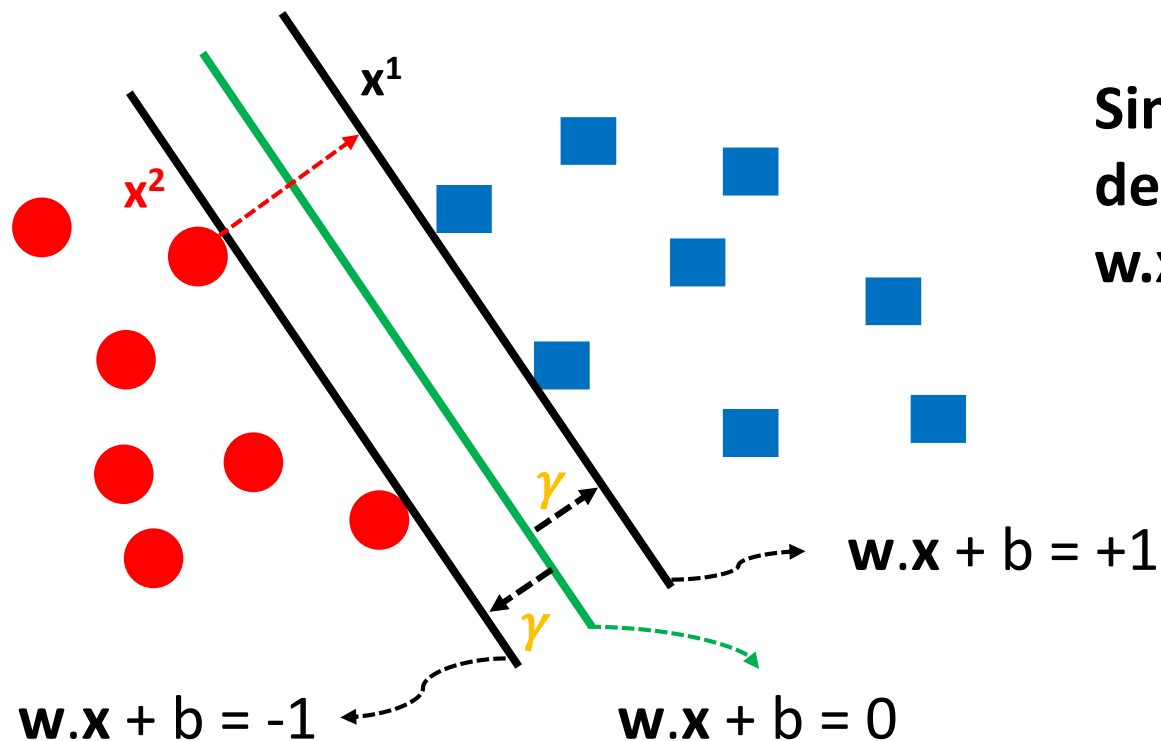


The distance from  $x^2$  to  $x^1$  in units of  $w/\|w\|$  is  $2\gamma$ . That is,

$$x^1 = x^2 + 2\gamma \frac{w}{\|w\|}$$

# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector



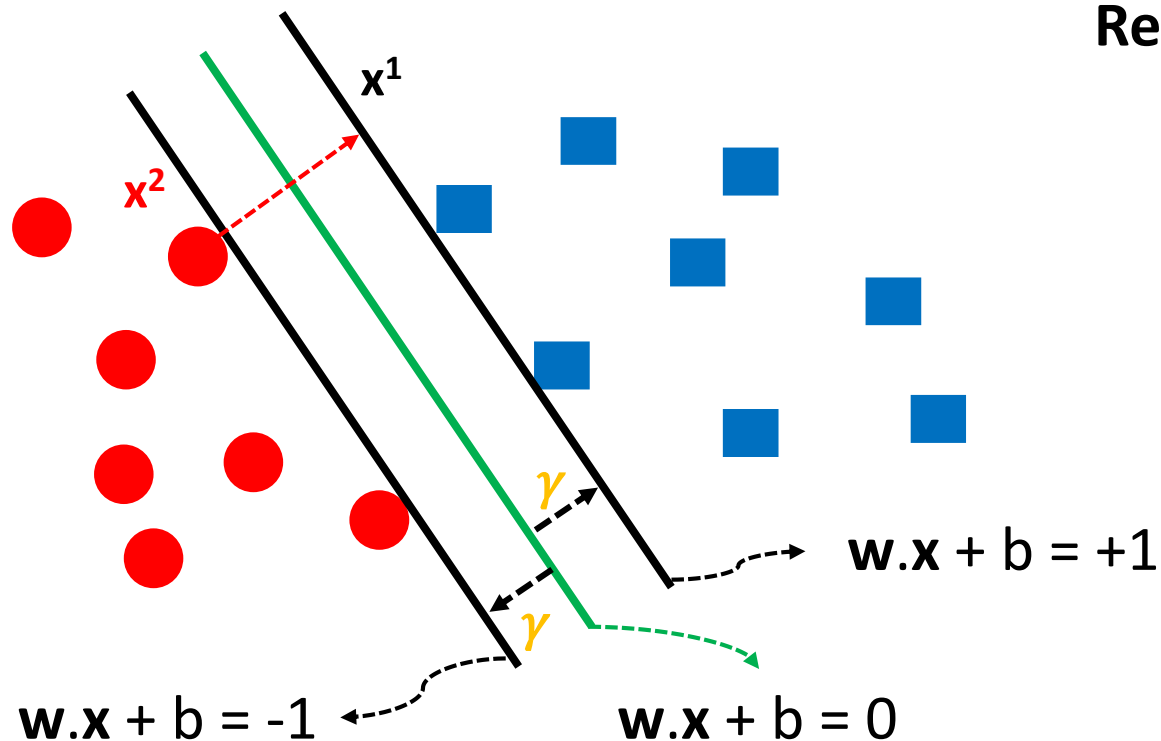
Since  $x^1$  is on the hyperplane defined by  $w.x + b = +1$ , we know that  $w.x^1 + b = 1$ . If we substitute for  $x^1$ :

$$\begin{aligned} x^1 &= x^2 + 2\gamma \frac{w}{\|w\|} \\ &\equiv \\ w.(x^2 + 2\gamma \frac{w}{\|w\|}) + b &= 1 \end{aligned}$$



# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector

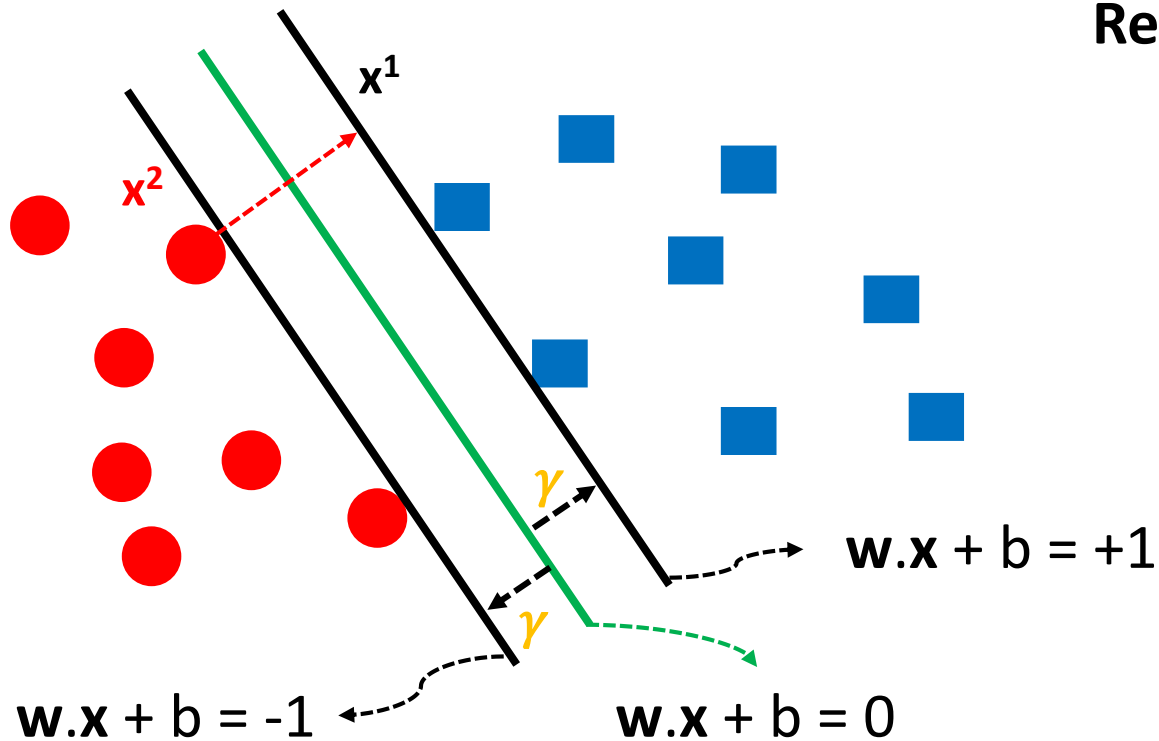


Regrouping terms, we see:

$$\begin{aligned}x^1 &= x^2 + 2\gamma \frac{w}{\|w\|} \\ &\equiv \\ w.(x^2 + 2\gamma \frac{w}{\|w\|}) + b &= 1 \\ &\equiv \\ \underbrace{w.x^2 + b}_{-1} + 2\gamma \frac{w.w}{\|w\|} &= 1 \\ -1 &\end{aligned}$$

# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector



Regrouping terms, we see:

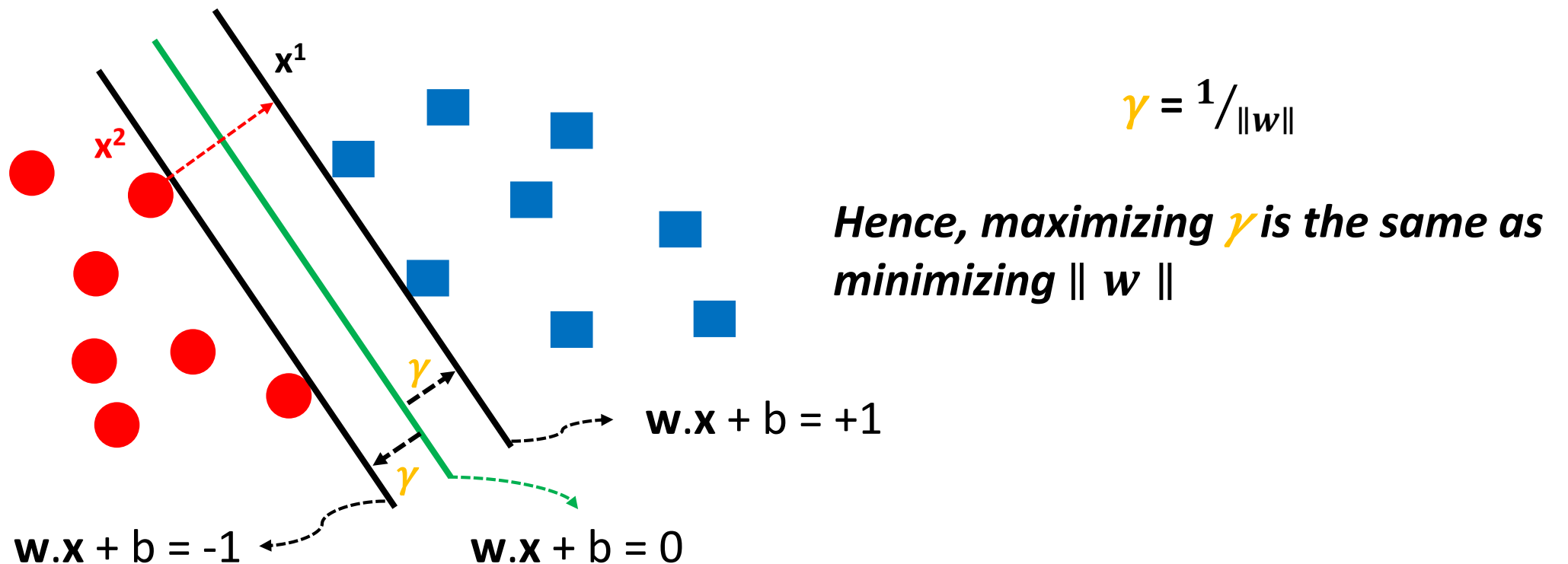
$$\begin{aligned} x^1 &= x^2 + 2\gamma w/\|w\| \\ &\equiv \\ w \cdot (x^2 + 2\gamma w/\|w\|) + b &= 1 \\ &\equiv \\ w \cdot x^2 + b + 2\gamma w \cdot w/\|w\| &= 1 \\ &\equiv \\ \gamma \|w\|^2/\|w\| &= 1 \\ &\equiv \\ \gamma &= 1/\|w\| \end{aligned}$$

# Recall

- Recall the distance from a point  $(x_0, y_0)$  to a line:
- $Ax + By + c = 0$  is  $|A x_0 + B y_0 + c| / \sqrt{A^2 + B^2}$
- The distance between  $H$  and  $H_1$  is:
- $|\mathbf{w} \bullet \mathbf{x} + b| / \|\mathbf{w}\| = 1 / \|\mathbf{w}\|$

# The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector  $w$  (i.e.,  $w/\|w\|$ ), thus making it a unit vector



# The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , maximize  $\gamma$  (by varying  $\mathbf{w}$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}^i + b \geq \gamma & \text{if } y^i = +1 \\ \mathbf{w} \cdot \mathbf{x}^i + b \leq -\gamma & \text{if } y^i = -1 \end{cases}$$

# The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} w \cdot x^i + b \geq 1 & \text{if } y^i = +1 \\ w \cdot x^i + b \leq -1 & \text{if } y^i = -1 \end{cases}$$

# Finding the Decision Boundary

- Let  $\{x_1, \dots, x_n\}$  be our data set and let  $y_i \in \{1, -1\}$  be the class label of  $x_i$   
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$$
- The decision boundary should classify all points correctly  $\Rightarrow$
- To see this: when  $y=-1$ , we wish  $(\mathbf{w}\mathbf{x}+b)<1$ , when  $y=1$ , we wish  $(\mathbf{w}\mathbf{x}+b)>1$ . For support vectors, we wish  $y(\mathbf{w}\mathbf{x}+b)=1$ .
- The decision boundary can be found by solving the following constrained optimization problem

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

# The Dual Problem (we ignore the derivation)

- The original problem is known as the primal problem
- Reformulate the optimization problem: A "trick" often used in optimization is to do a Lagrangian formulation of the problem. The constraints will be replaced by constraints on the Lagrangian multipliers and the training data will only occur as dot products.
- The new objective function is in terms of  $\alpha_i$  only
- It is known as the dual problem: if we know  $\mathbf{w}$ , we know all  $\alpha_i$ ; if we know all  $\alpha_i$ , we know  $\mathbf{w}$
- The objective function of the dual problem needs to be maximized!



# The Dual Problem (we ignore the derivation)

- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

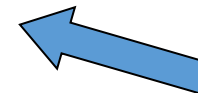
$$\max. \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to  $\alpha_i \geq 0$ ,



Properties of  $\alpha_i$  when we introduce the Lagrange multipliers

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



The result when we differentiate the original Lagrangian w.r.t.  $b$

# The Dual Problem

$$\max. \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

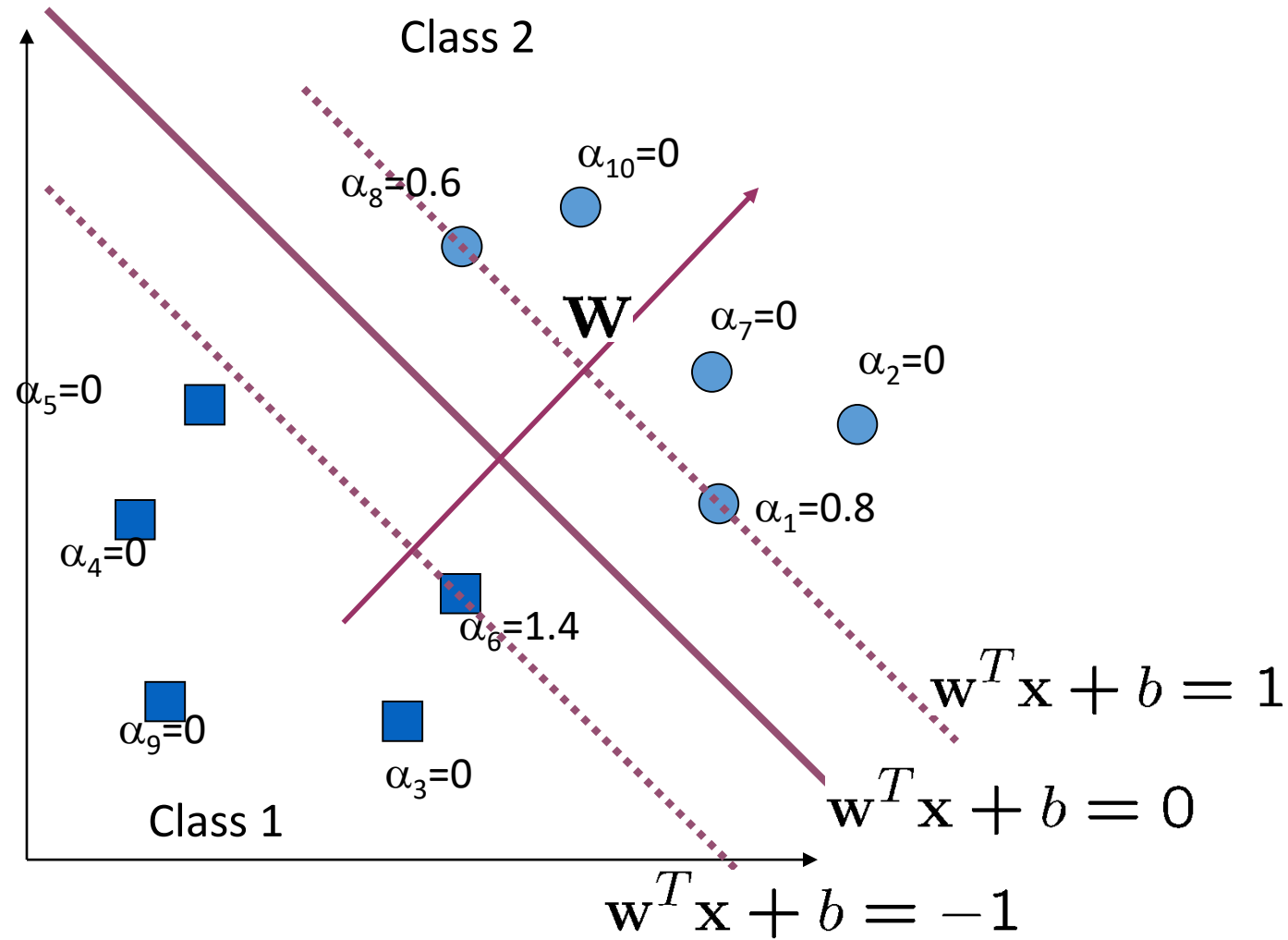
- This is a quadratic programming (QP) problem
  - A global maximum of  $\alpha_i$  can always be found

- $\mathbf{w}$  can be recovered by 
$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

# Characteristics of the Solution

- Many of the  $\alpha_i$  are zero (see next slide for example)
  - $\mathbf{w}$  is a linear combination of a small number of data points
  - This “sparse” representation can be viewed as data compression as in the construction of knn classifier
- $\mathbf{x}_i$  with non-zero  $\alpha_i$  are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let  $t_j$  ( $j=1, \dots, s$ ) be the indices of the  $s$  support vectors.  
We can write 
$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$
- For testing with a new data  $\mathbf{z}$ 
  - Compute  $\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$  and classify  $\mathbf{z}$  as class 1 if the sum is positive, and class 2 otherwise
  - Note:  $\mathbf{w}$  need not be formed explicitly

# A Geometrical Interpretation



# Non separable case- Soft margin

**Objective:** find a good separating hyper-plane for the non-separable case

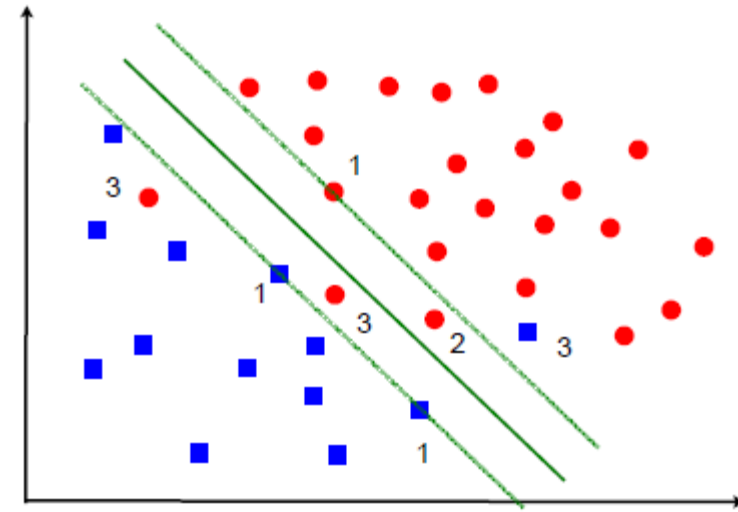
**Problem:** Cannot satisfy  $y_i[\mathbf{w}^\top \mathbf{x}_i + b] \geq 1$  for all  $i$

**Solution:** *Slack* variables

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq +1 - \xi_i && \text{for } y_i = +1, \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq -1 + \xi_i && \text{for } y_i = -1, \\ \xi_i &\geq 0 && k = 1, 2, \dots, n. \end{aligned}$$

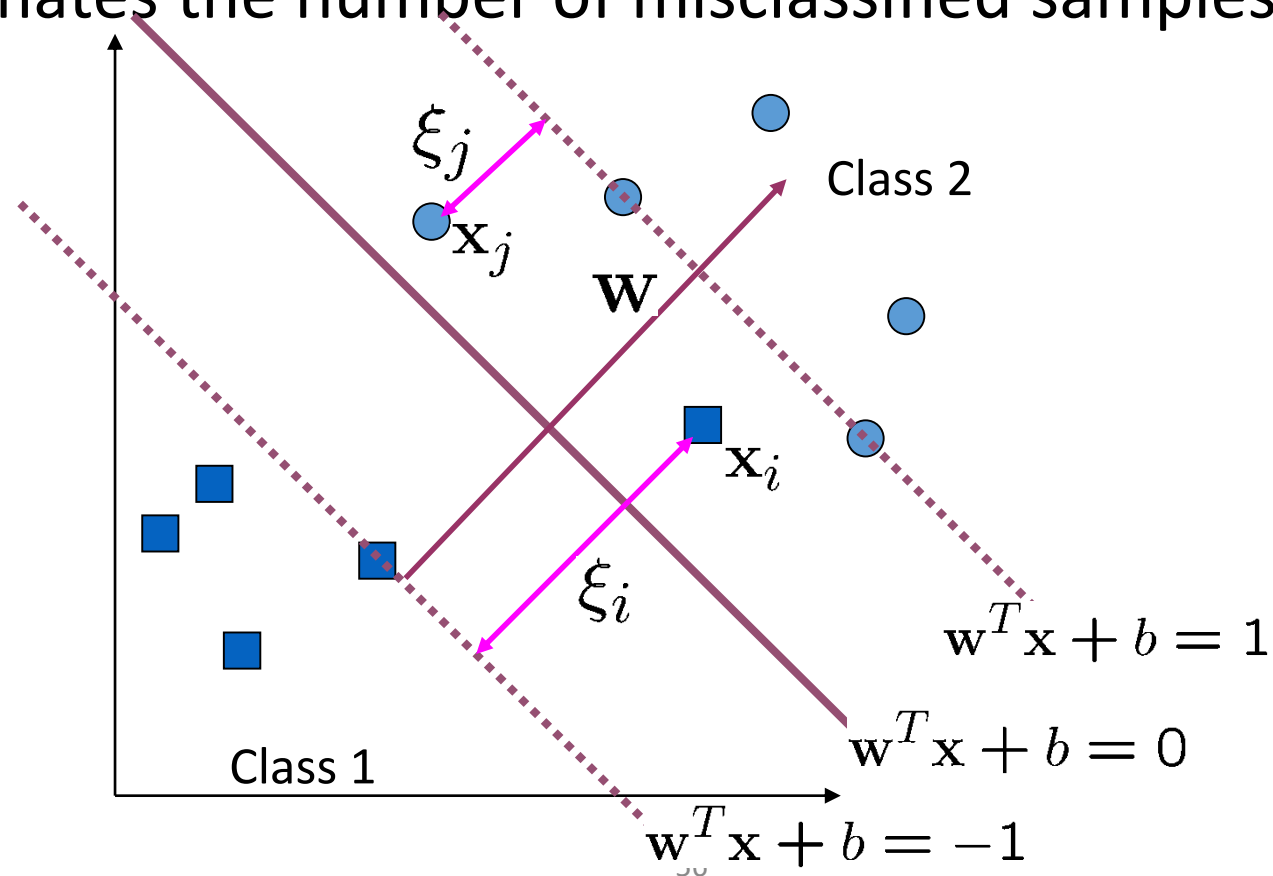
An error occurs if  $\xi_i > 1$ . Thus,

$$\sum_{i=1}^n I(\xi_i > 1) = \# \text{ errors}$$



# Allowing errors in our solutions

- We allow “error”  $\xi_i$  in classification; it is based on the output of the discriminant function  $\mathbf{w}^T \mathbf{x} + b$
- $\xi_i$  approximates the number of misclassified samples



**Proposed solution:** Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^n I(\xi_i > 1) \quad (\text{non - convex!})$$

**Suggestion:** Replace  $I(\xi_i > 1)$  by  $\xi_i$  (upper bound)

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & L_P(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

**Tradeoff:** Large  $C$  - penalize errors, Small  $C$  penalize complexity

**Dual Problem:** Same as in separable case, except that  $0 \leq \alpha_i \leq C$

**Support vectors:**  $\alpha_i > 0$  - but lose geometric interpretation!

# Soft Margin Hyperplane

- If we minimize  $\sum_i \xi_i$ ,  $\xi_i$  can be computed by

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- $\xi_i$  are “slack variables” in optimization
- Note that  $\xi_i=0$  if there is no error for  $\mathbf{x}_i$
- $\xi_i$  is an upper bound of the number of errors
- We want to minimize  $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$ 
  - $C$  : tradeoff parameter between error and margin
- The optimization problem becomes Minimize  $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$   
subject to  $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$



# Choice of C

- C is referred to as a *regularization parameter*.
  - Pick a large C if you really do not want to misclassify points, but you would accept a narrow margin
  - Pick a small C if you are OK with some misclassified points, but want most of the points to be far away from the boundary (i.e., the margin is large)

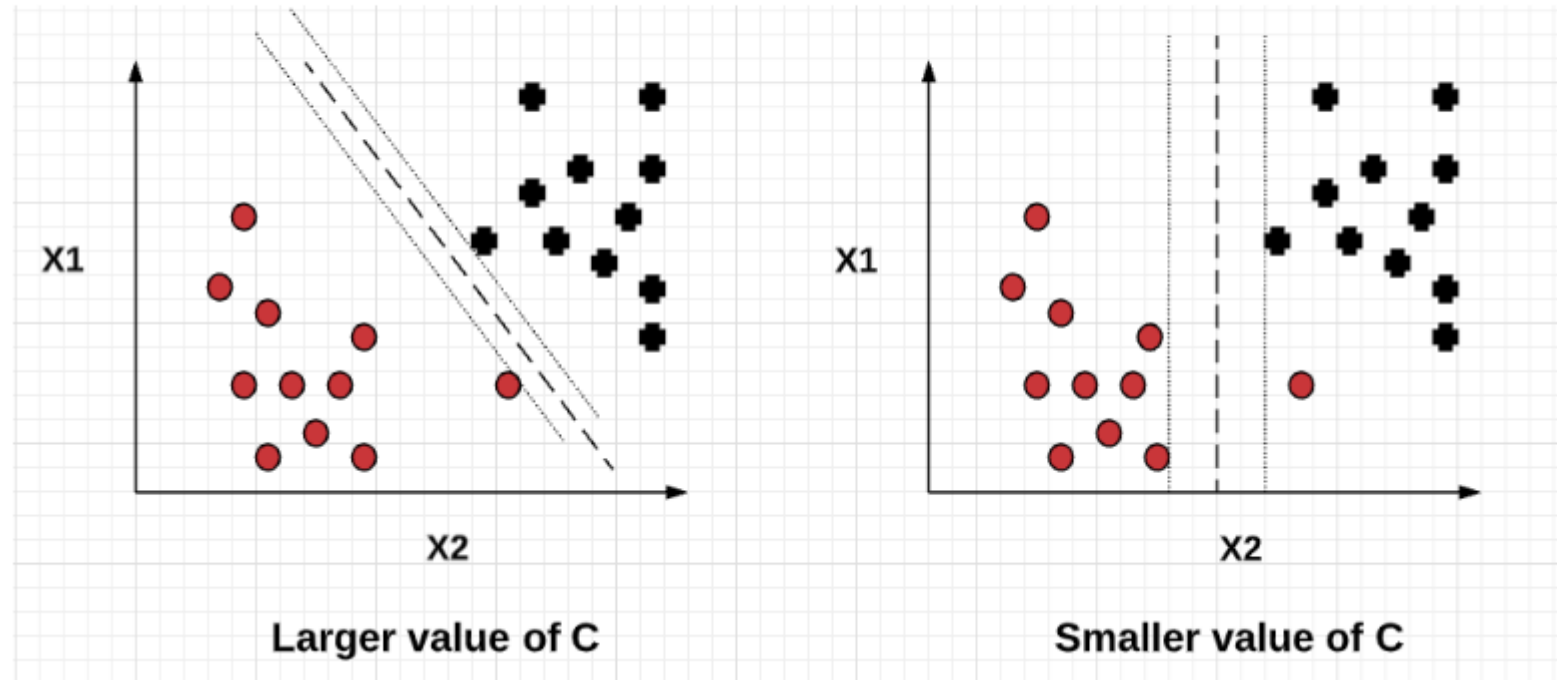
# Choice of $C$

- The choice of  $C$  depends on the problem at hand and the characteristics of the dataset.
- In general, a smaller value of  $C$  is preferred when the dataset is noisy, or when the goal is to have a more generalizable model that can perform well on new, unseen data.
- A larger value of  $C$  is preferred when the dataset is clean and the goal is to maximize classification accuracy on the training data.

# Choice of C

- As a general guideline, it's a good idea to start with a smaller value of C and gradually increase it until the desired level of classification accuracy is achieved. However, it's important to keep in mind that a larger value of C may lead to overfitting, which means that the model performs well on the training data but poorly on new, unseen data. Therefore, it's important to strike a balance between maximizing the margin and minimizing the classification error, and choose an appropriate value of C based on the specific problem and dataset.

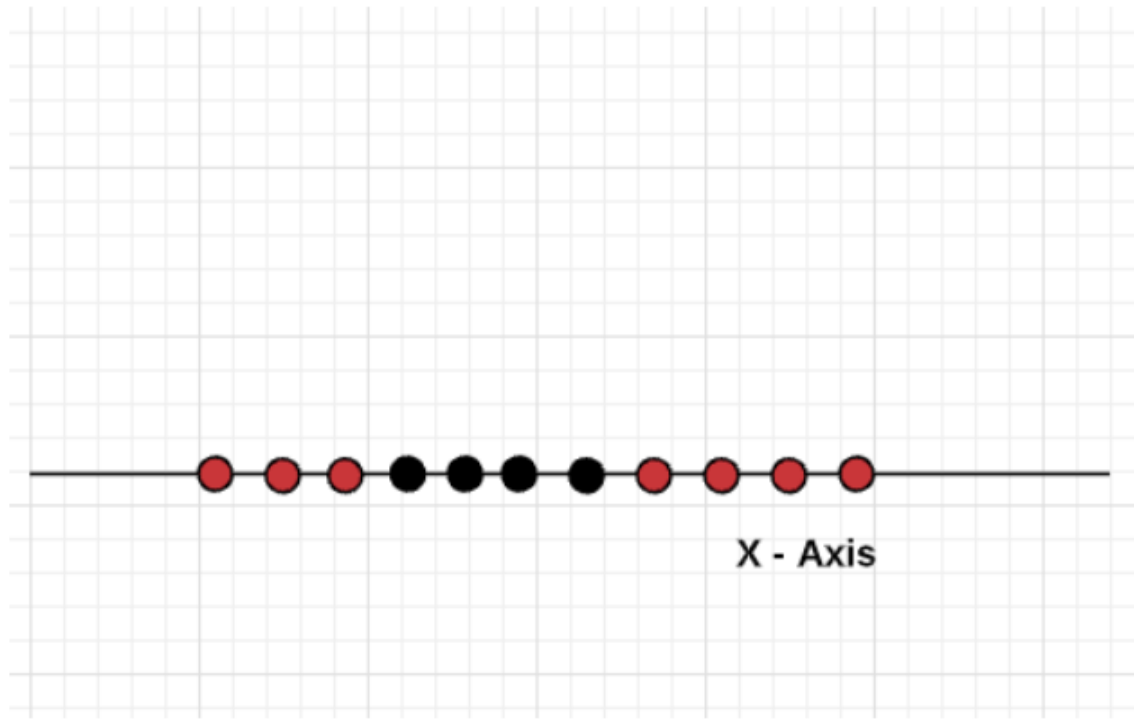
The diagram given below represents the models with different value of  $C$ .



# Extension to Non-linear Decision Boundary

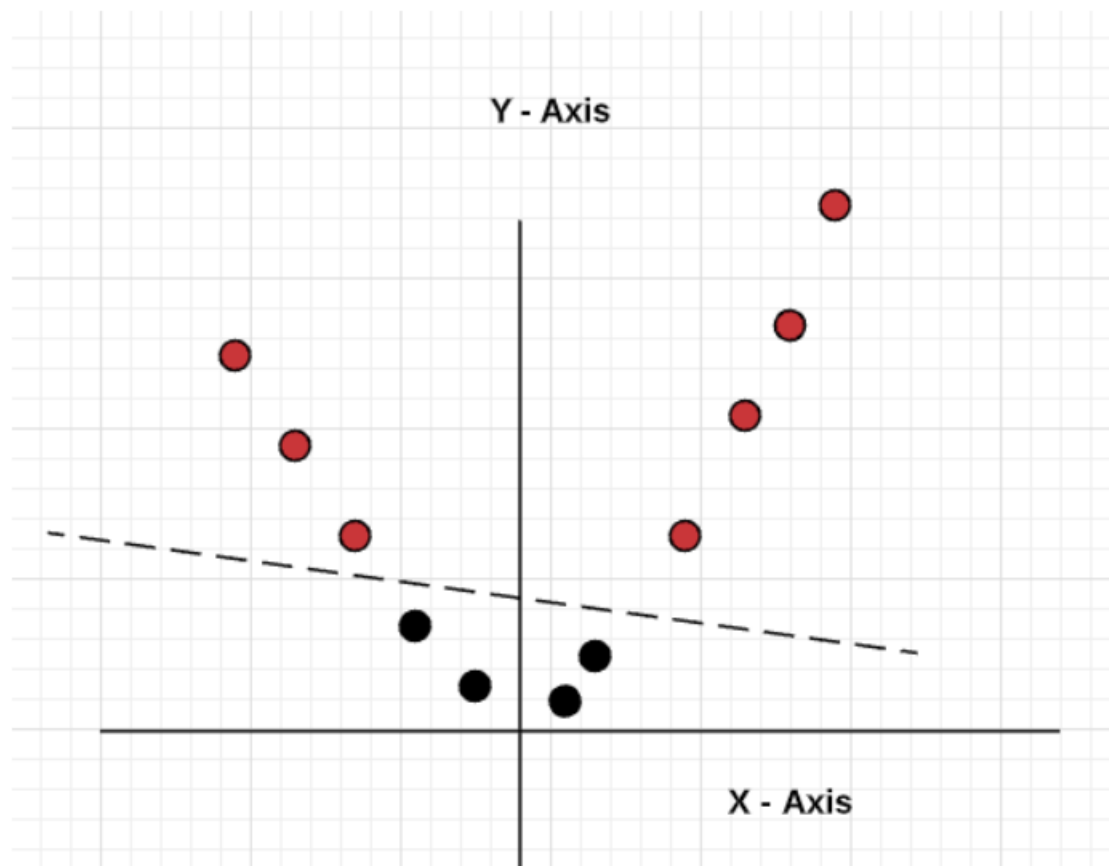
- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform  $\mathbf{x}_i$  to a higher dimensional space
  - Input space: the space the point  $\mathbf{x}_i$  are located
  - Feature space: the space of  $\phi(\mathbf{x}_i)$  after transformation

Let's take a look at another simple example of data in 1 dimension which is not easy to separate and how adding another dimension makes it easy.

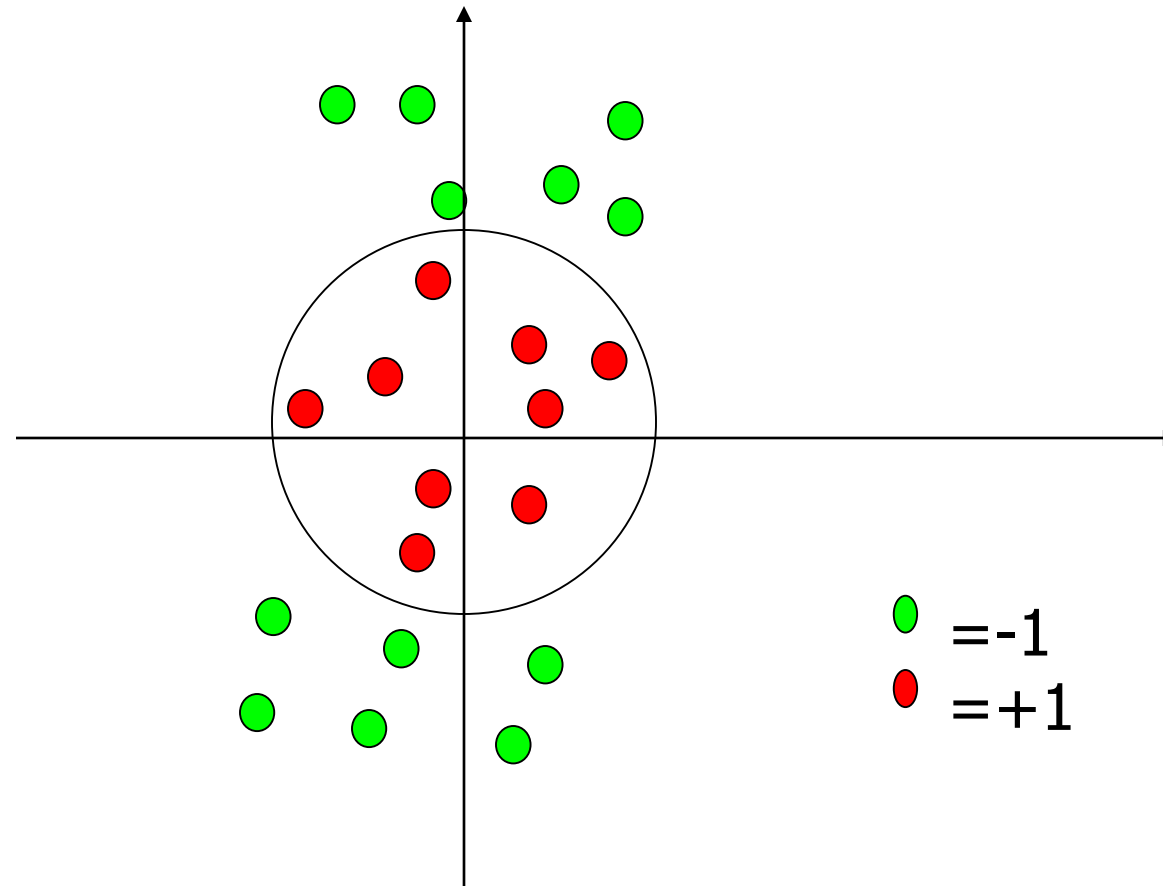


**Fig 3. Linearly inseparable data in one-dimension**

Let's apply the method of adding another dimension to the data by using the function  $Y = X^2$  (X-squared). Thus, the data looks like the following after applying the kernel function ( $Y = X^2$ ) and becomes linearly separable.



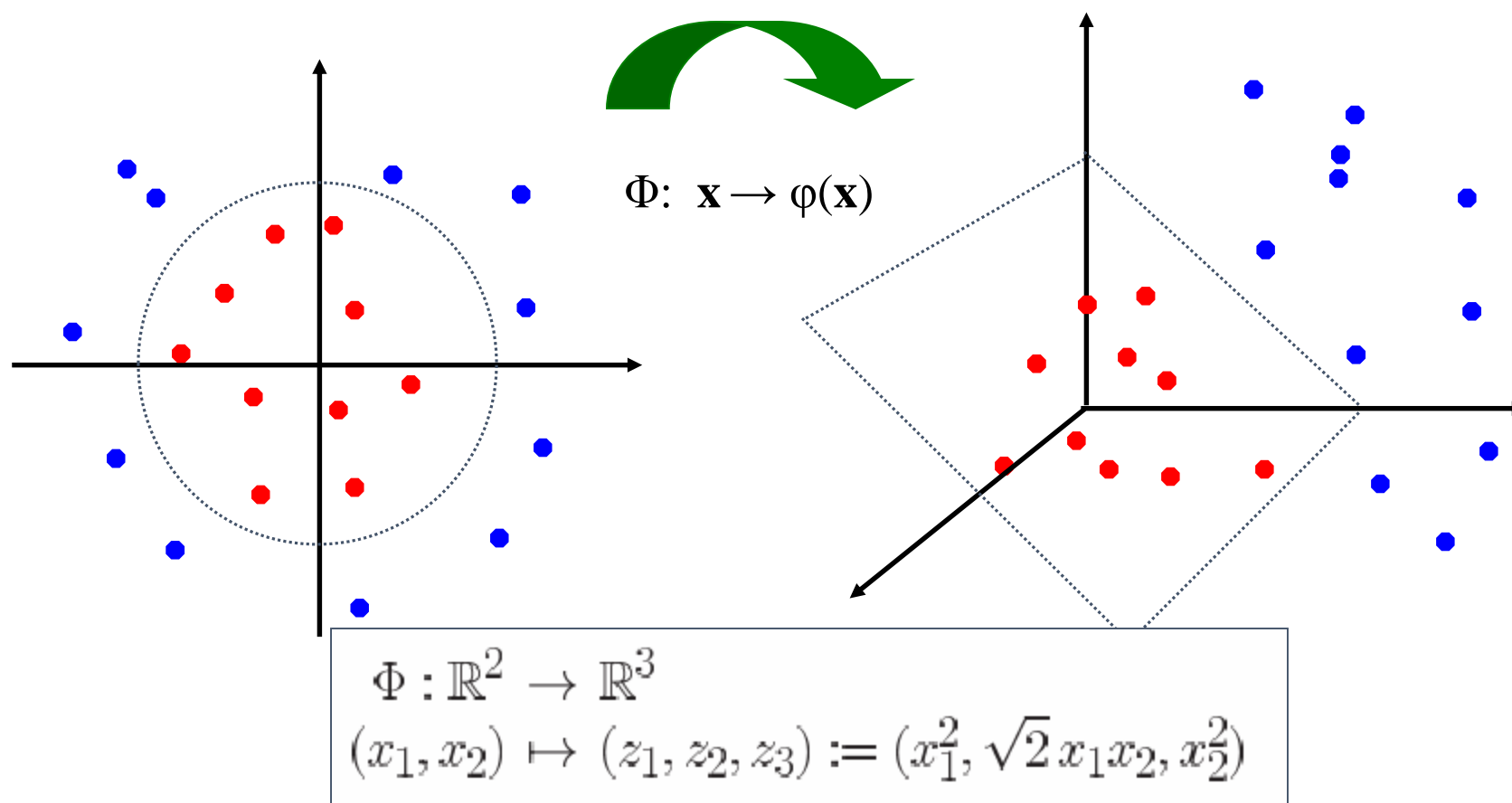
# Extension to Non-linear decision boundary



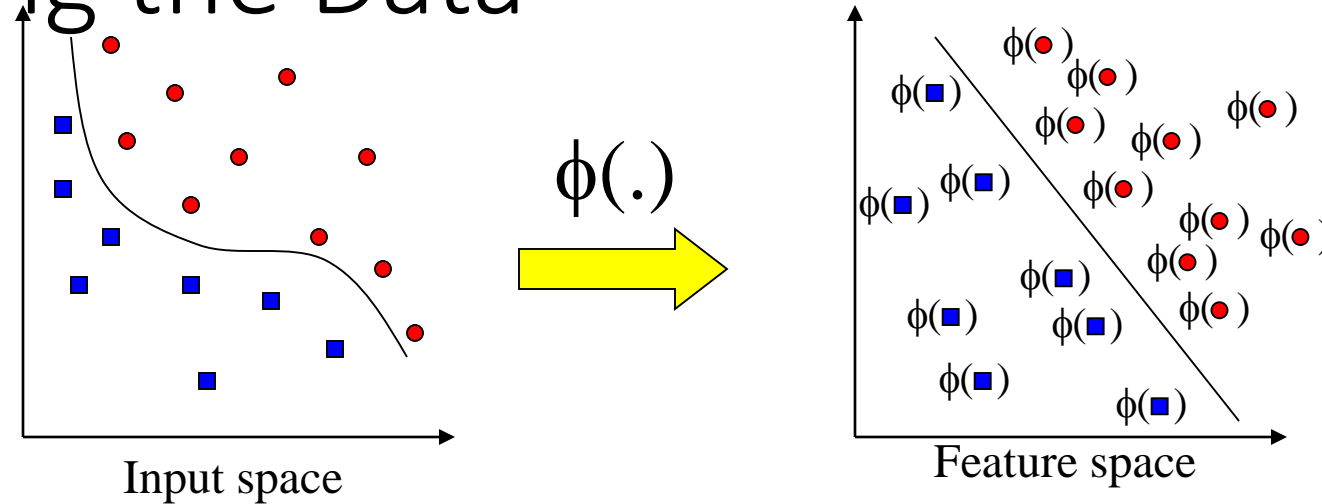
What if the decision function is not a linear?



# Non-linear SVMs



# Transforming the Data

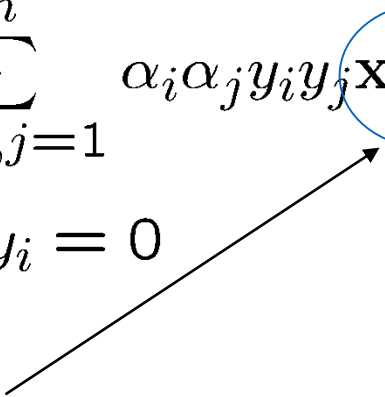


Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

# The Kernel Trick

- Recall the SVM optimization problem

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


- The data points only appear as **dot product**
- As long as we can calculate the dot product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function  $K$  by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

# An Example for $\phi(\cdot)$ and $K(\cdot, \cdot)$

- Suppose  $\phi(\cdot)$  is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

$$\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out  $\phi(\cdot)$  explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out  $\phi(\cdot)$  explicitly is known as the **kernel trick**

**Linear Separability:** More likely in high dimensions

**Mapping:** Map input into high-dimensional *feature space*  $\Phi$

**Classifier:** Construct *linear* classifier in  $\Phi$

**Motivation:** Appropriate choice of  $\Phi$  leads to linear separability.

**Non-linearity** and **high dimension** are essential (Cover '65)!

$$\Phi : \mathbb{R}^d \mapsto \mathbb{R}^D \quad (D \gg d)$$

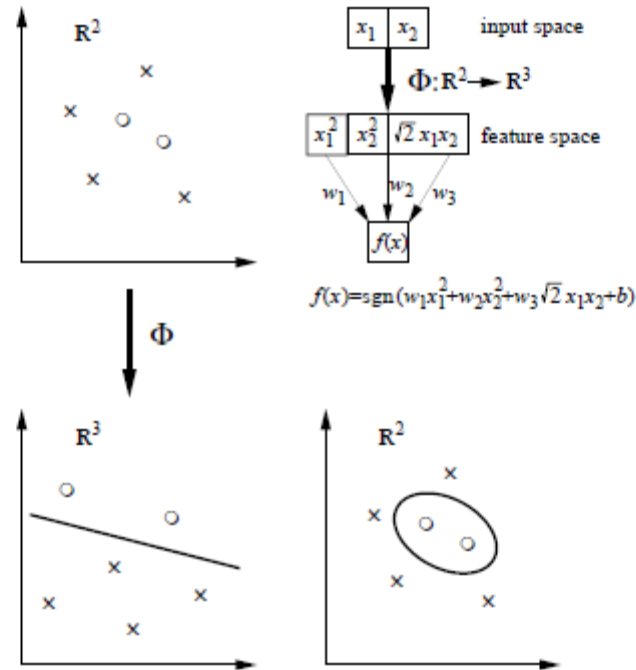
$$\mathbf{x} \mapsto \Phi(\mathbf{x})$$

**Hyper-plane condition:**  $\mathbf{w}^\top \Phi(\mathbf{x}) + b = 0$

**Inner products:**  $\mathbf{x}^\top \mathbf{x} \mapsto \Phi^\top(\mathbf{x}) \Phi(\mathbf{x})$

### Decisions in Input and Feature Space:

Problems becomes linearly separable in feature space (Fig. from Schölkopf & Smola 2002)



## Recall for linear SVM

$$f(\mathbf{x}) = \sum_{i=1}^d \mathbf{w}_i x_i + b \quad ; \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

Obtained

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b$$

In feature space

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b$$

**Kernel:** A symmetric function  $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$

**Inner product kernels:** In addition

$$K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x})^\top \Phi(\mathbf{z})$$

**Motivation:**  $\Phi \in \mathbb{R}^D$ , where  $D$  may be very large - inner products expensive

Classifier:

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn} \left( \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b \right) \\ &= \text{sgn} \left( \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \end{aligned}$$

The gain: Implement **infinite-dimensional** mapping, but do all calculations in **finite dimension**

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \quad ; \quad 0 \leq \alpha_i \leq C \end{aligned}$$

**Observe:** Only difference from linear case is in the kernel

Optimization task is unchanged!



# A Few Good Kernels

- Dot product kernel
  - $K(x_1, x_2) = \langle x_1, x_2 \rangle$
- Polynomial kernel
  - $K(x_1, x_2) = \langle x_1, x_2 \rangle^d$  (Monomials of degree  $d$ )
  - $K(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^d$  (All monomials of degree  $1, 2, \dots, d$ )
- Gaussian kernel or RBF kernel
  - $K(x_1, x_2) = \exp(-||x_1 - x_2||^2 / 2\sigma^2)$
  - Radial basis functions
- Sigmoid kernel
  - $K(x_1, x_2) = \tanh(\langle x_1, x_2 \rangle + \vartheta)$
  - Neural networks

# Gamma parameter

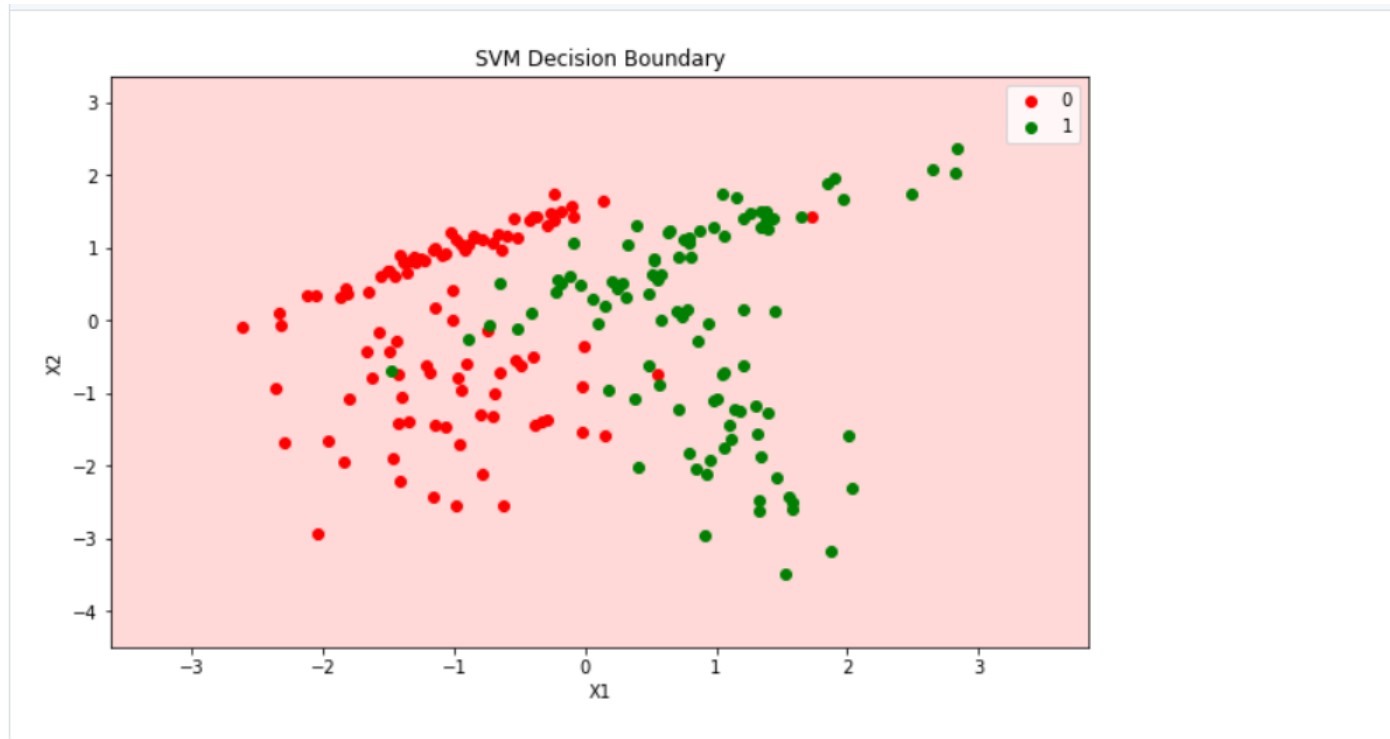
When introducing a new parameter  $\gamma = 1 / 2\sigma^2$ , the equation will be

$$K(X_1, X_2) = \exp(-\gamma \|X_1 - X_2\|^2)$$

# Gamma parameter

- The Gamma Hyperparameter comes in handy in case of a RBF or Poly kernel. It doesn't have any role when your kernel is linear.
- Gamma essentially controls the distance of influence of a single training point.
- Low values of Gamma indicate a large similarity radius which results in more points being grouped together.
- For High values of Gamma, the points need to be very close to each other in order for it to be considered in the same group.

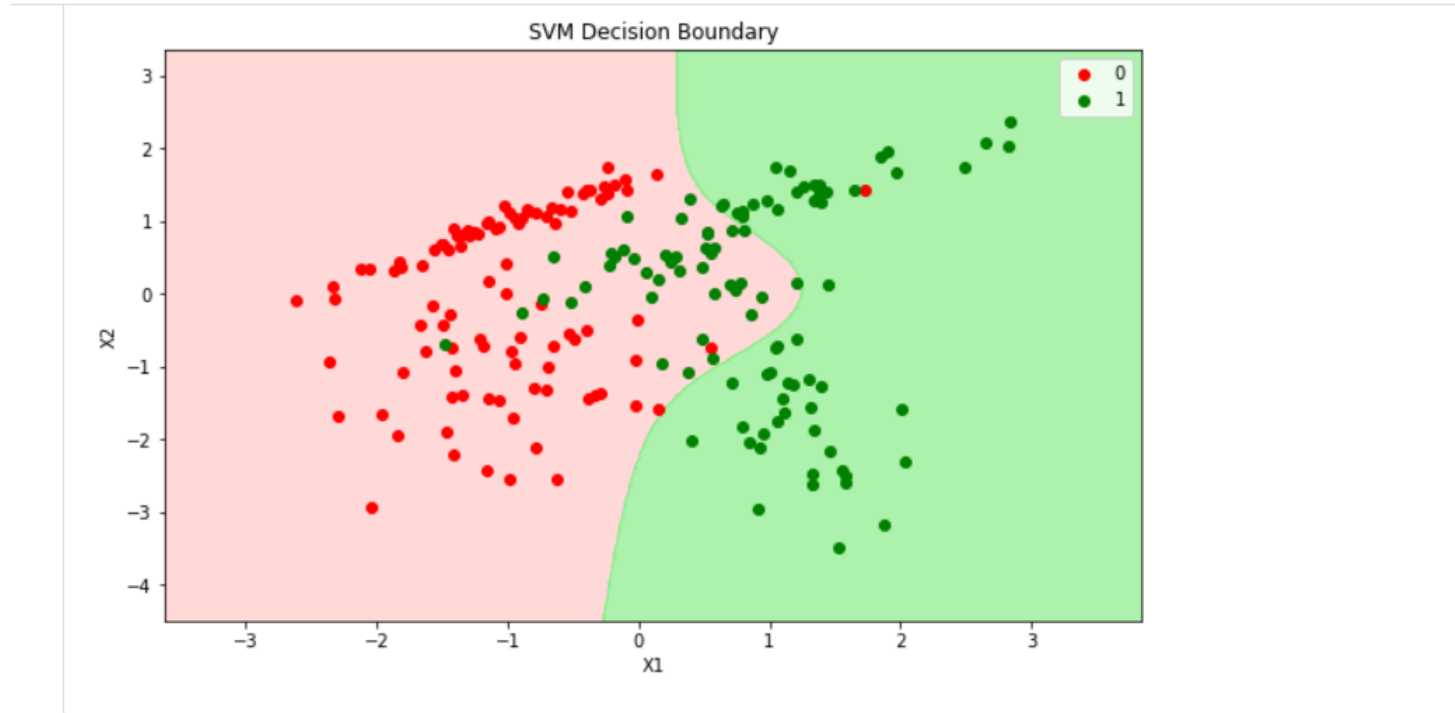
# Gamma=0.01



## Some Observations

- The model is not very accurate as it classifies all points into Class 0
- The accuracy of the model is 50%
- Gamma value is really low indicating that there is a large similarity radius which results in more points being clubbed together.

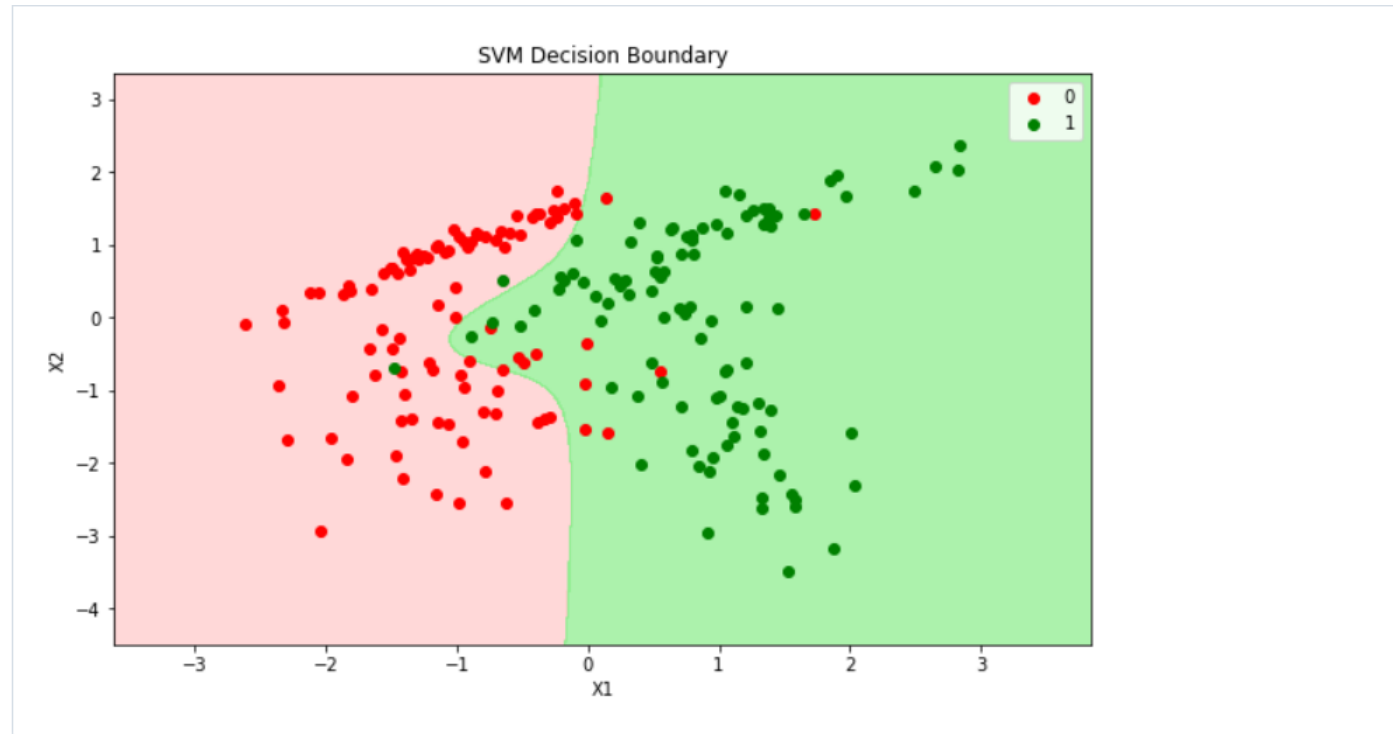
# Gamma=0.1



## Some Observations

- After increasing Gamma from 0.001 to 0.1, the model accuracy has increased from 50% to 80%.
- Increasing the Gamma value indicated that the similarity radius was made small which made the accuracy go up.

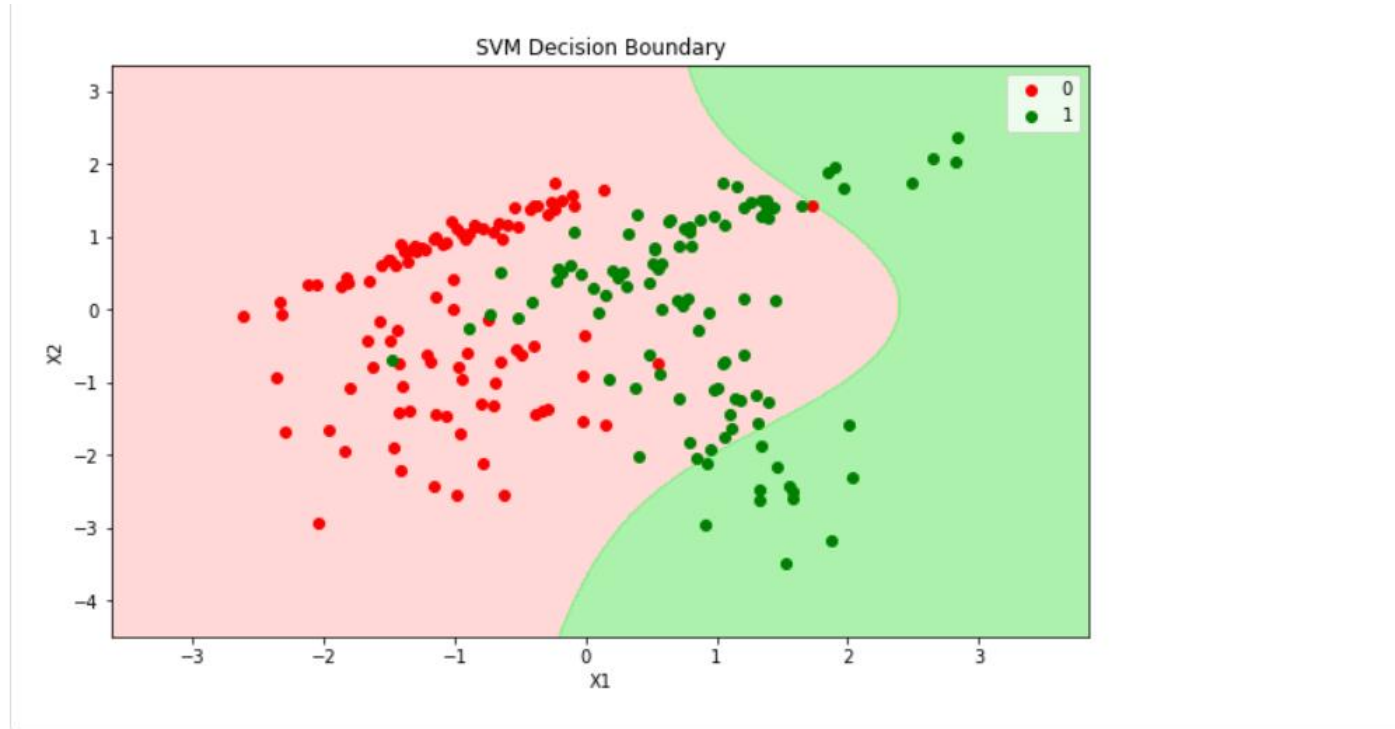
# Gamma=1



## Some Observations

- The model accuracy has increased to 93%
- As we keep increasing the Gamma value, the similarity radius would keep decreasing the distance of influence of a single point

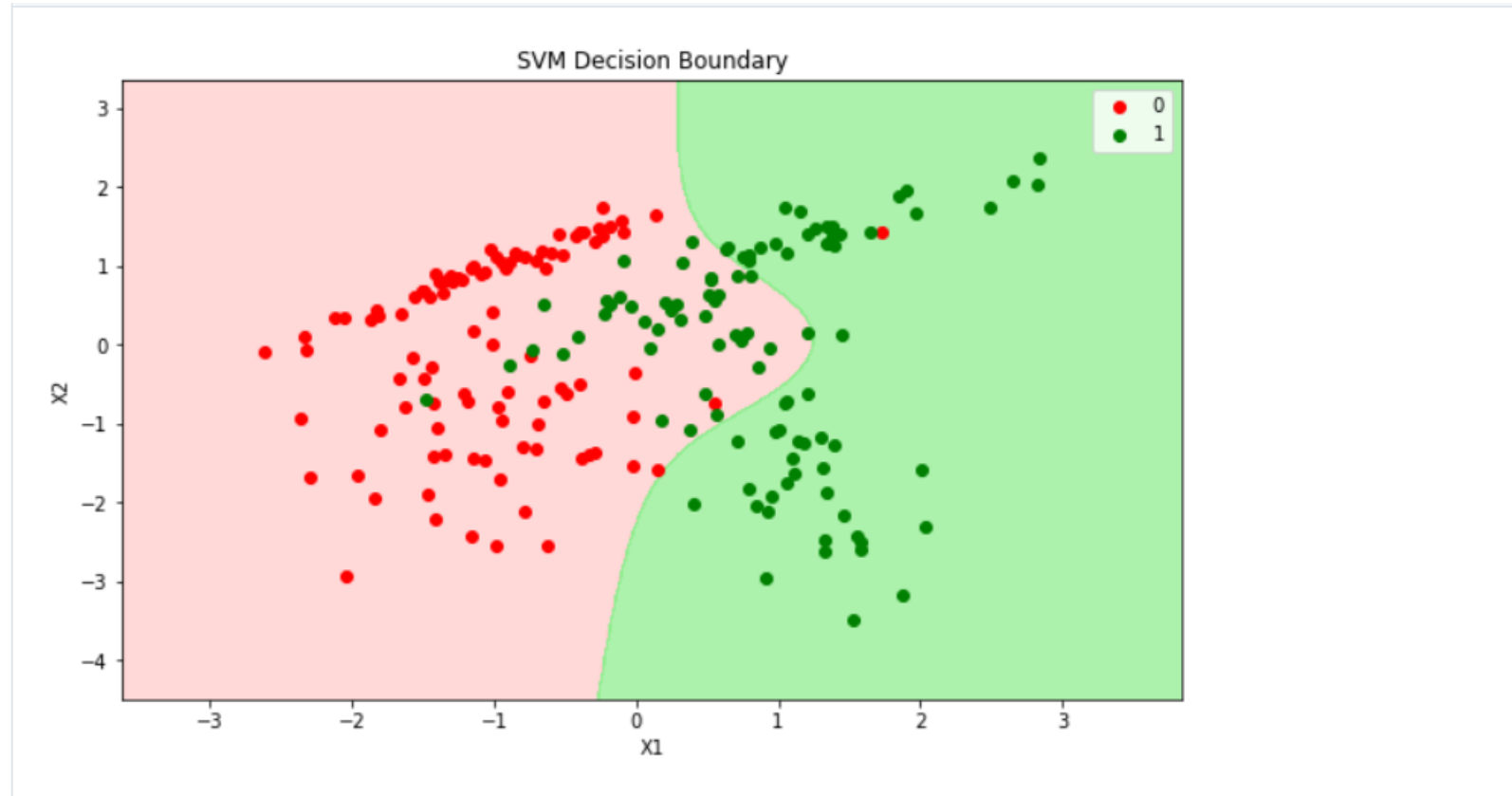
# C=0.1 Gamma constant=0.1



## Some Observations

- We will keep Gamma a constant to monitor the effect of C
- C is small and Gamma is set to 0.1
- The model is 60% accurate.
- The reason is simple, lower the C value, lower importance to misclassification thus less accuracy.

# Gamma same C=1

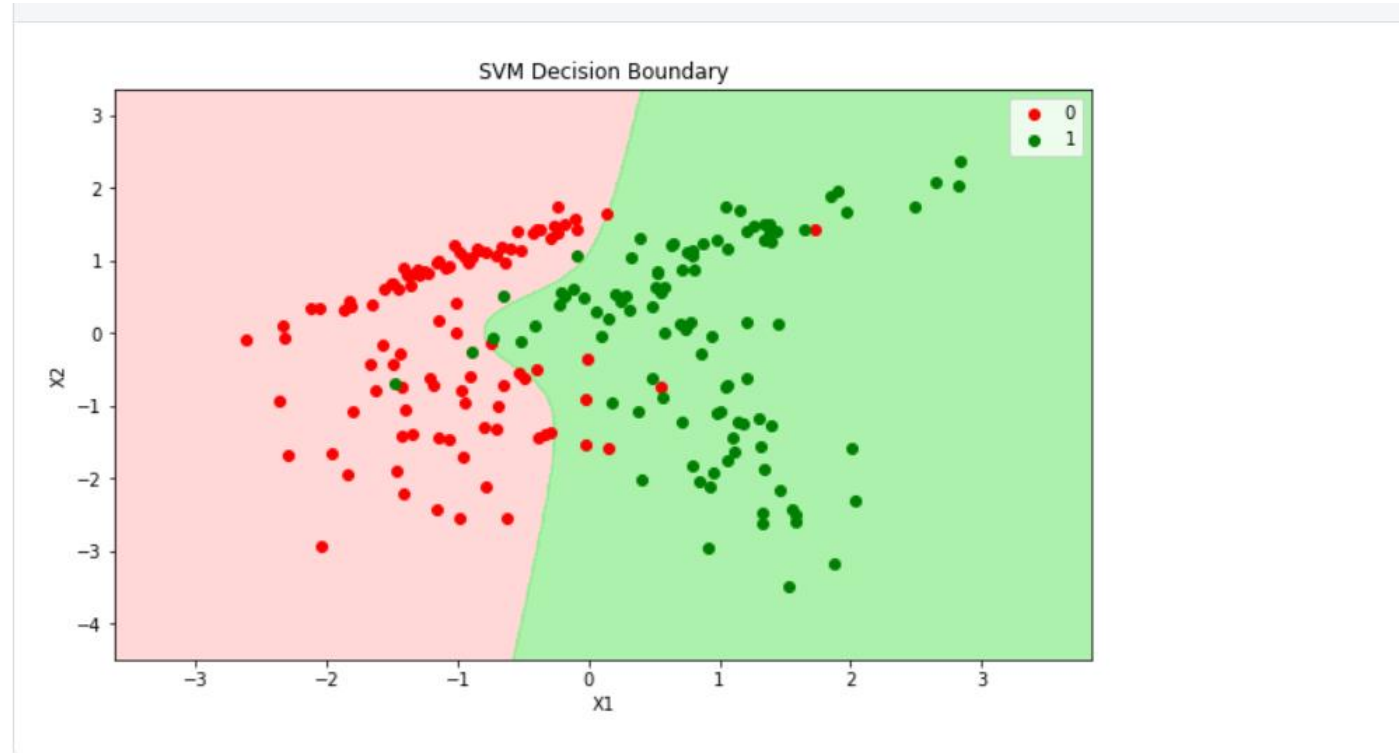


## Some Observations

- As we increased  $C$  to 1, more weightage was given to misclassified points thus the algorithm tried correcting it which resulted in a model with 80% accuracy.



# Same Gamma C=10



## Some Observations

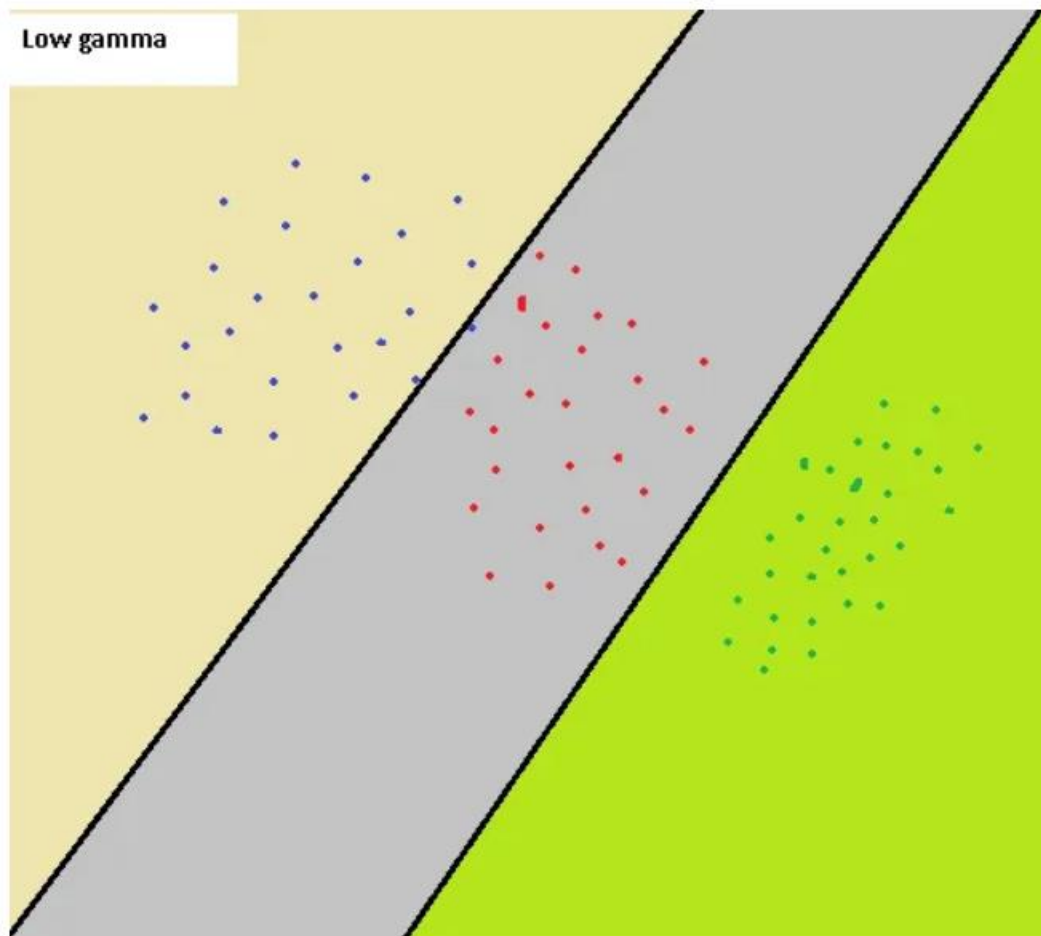
- Increasing  $C$  to 10, gave more emphasis on misclassification thus resulted in a more accurate model.

## **Gamma vs C parameter**

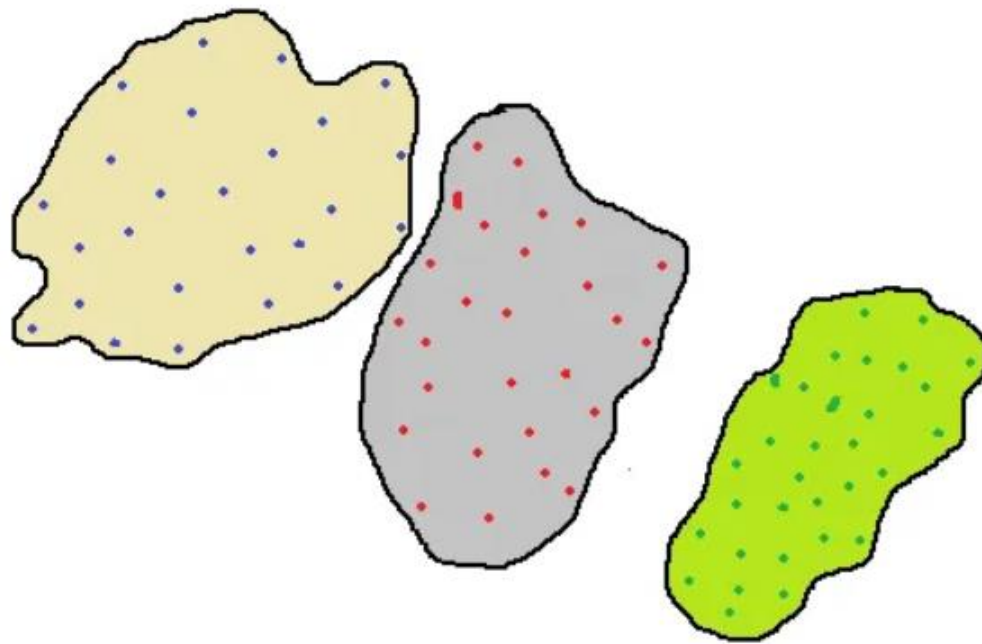
For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and gamma parameter need to be optimized simultaneously. If gamma is large, the effect of c becomes negligible. If gamma is small, c affects the model just like how it affects a linear model. Typical values for c and gamma are as follows. However, specific optimal values may exist depending on the application:

$$0.0001 < \text{gamma} < 10$$

$$0.1 < c < 100$$



Large gamma



The first image represents the case with a low gamma values. Similarity radius is large so all the points in the colored regions are considered to be in the same class. For instance, if we have a point the right bottom corner, it is classified as “green” class. On the other hand, the second image is the case with large gamma. For data points to be grouped in the same class, they must fall in the tight bounded area. Thus, a small noise may cause a data point to fall out of a class. Large gamma values are likely to end up in overfitting.

As the gamma decreases, the regions separating different classes get more generalized. Very large gamma values result in too specific class regions (overfitting).

# Example

- Suppose we have 5 one-dimensional data points
  - $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$ , with 1, 2, 6 as class 1 and 4, 5 as class 2  $\Rightarrow y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - $C$  is set to 100
- We first find  $\alpha_i$  ( $i=1, \dots, 5$ ) by

$$\max. \quad \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \quad \sum_{i=1}^5 \alpha_i y_i = 0$$

# Example

- By using a QP solver, we get
  - $\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are  $\{x_2=2, x_4=5, x_5=6\}$

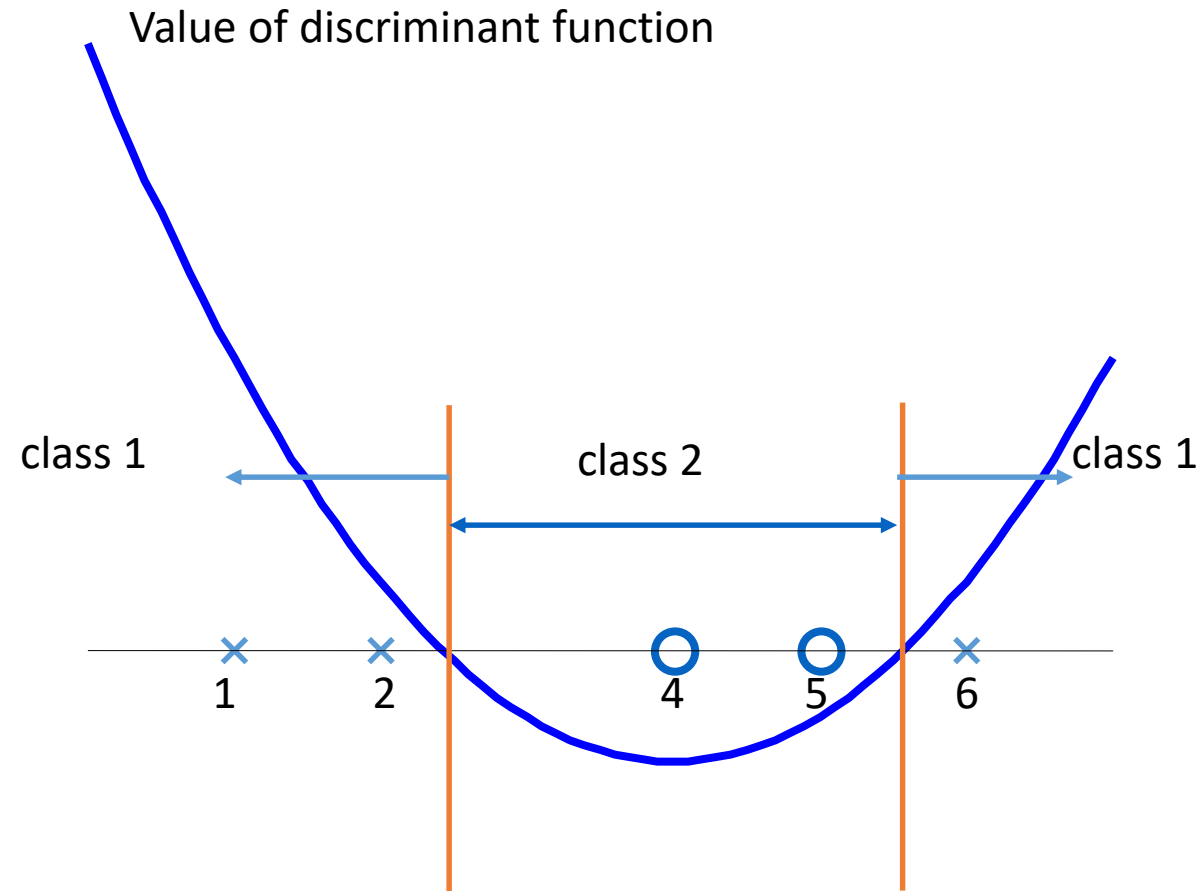
- The discriminant function is
 
$$f(z) = 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b$$

$$= 0.6667z^2 - 5.333z + b$$

- $b$  is recovered by solving  $f(2)=1$  or by  $f(5)=-1$  or by  $f(6)=1$ , as  $x_2$  and  $x_5$  lie on the line  $\phi(w)^T \phi(x) + b = 1$  and  $x_4$  lies on the line  $\phi(w)^T \phi(x) + b = -1$
- All three give  $b=9$

$$\longrightarrow f(z) = 0.6667z^2 - 5.333z + 9$$

# Example



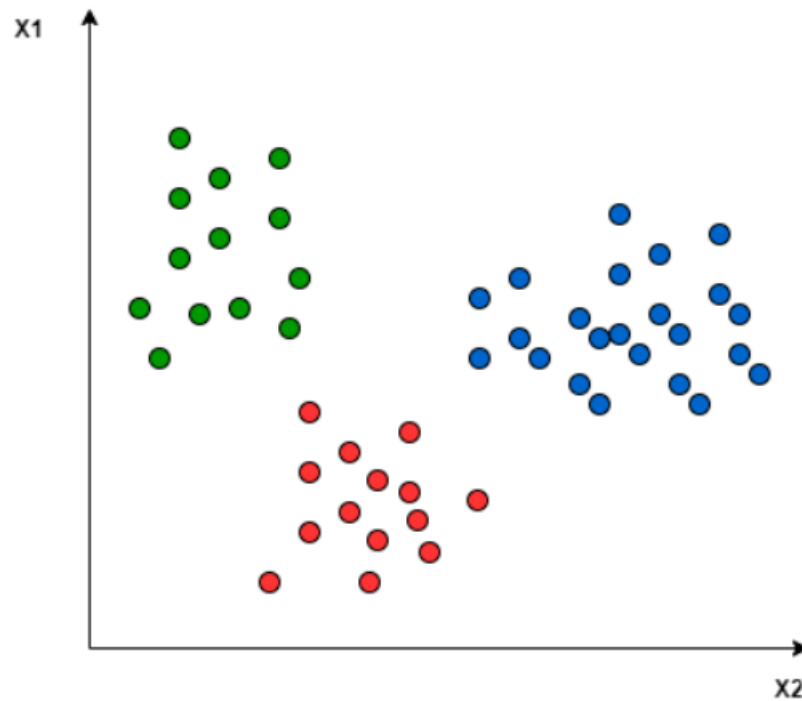


# Multiclass Classification Using SVM

- a *One-to-One* approach, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes.
- *One-to-rest* approach. In that approach, the breakdown is set to a binary classifier per each class

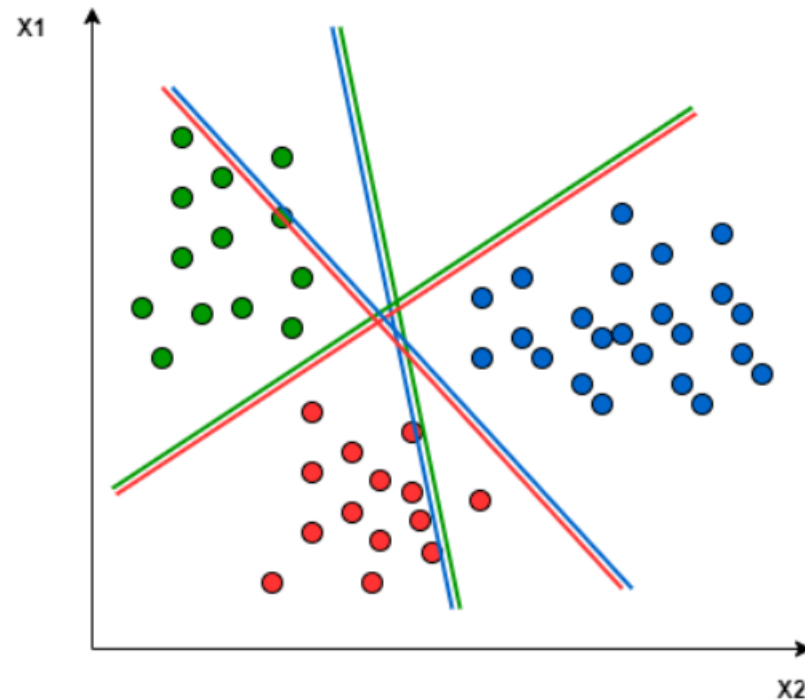
# Example

Let's take an example of 3 classes classification problem; green, red, and blue, as the following image:

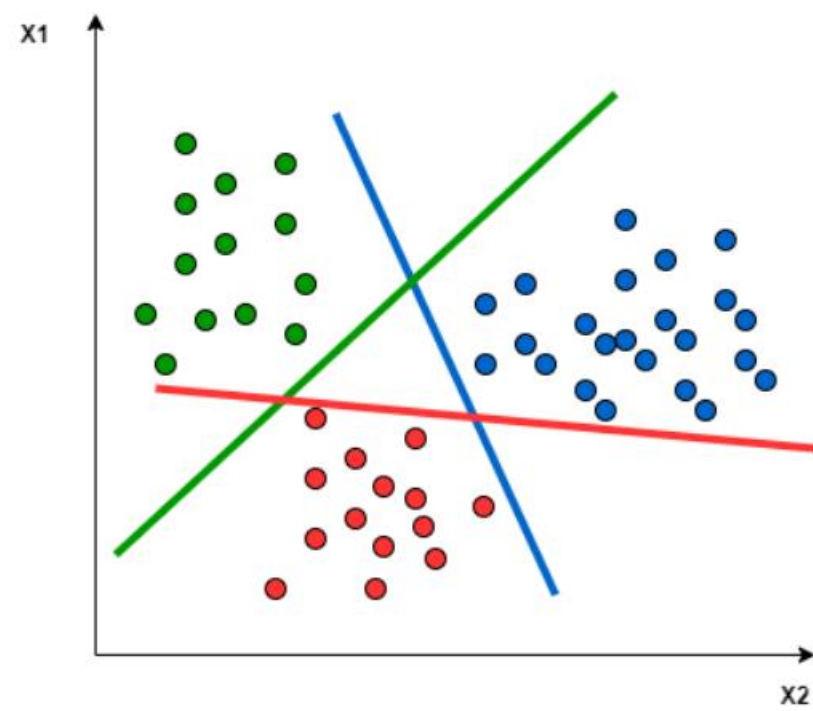


# One-to-One

In the *One-to-One* approach, we need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:



- In the *One-to-Rest* approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once:



# Strength or advantages of SVM

- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find globally best model
- Efficient algorithms
- kernel trick for non-linear boundary decision
- SVM is memory efficient since it relies on few support vectors for testing.

# Disadvantages of SVM

- Choosing a “good” kernel function is not easy.
- Long training time for large datasets.
- Difficult to understand and interpret the final model.
- The SVM hyper parameters are  $C$  and  $\gamma$ . It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.