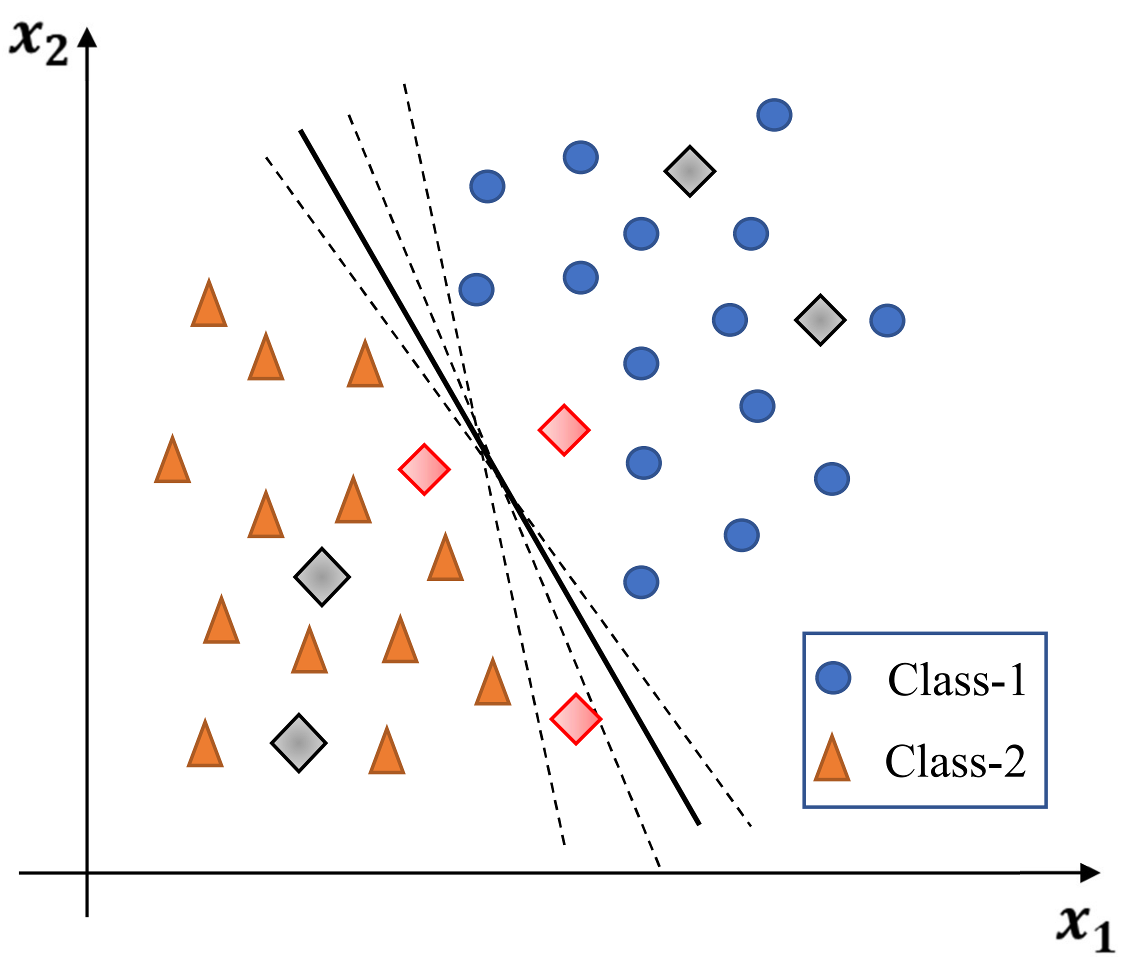
SUPPORT VECTOR MACHINE (SVM) CLASSIFIER

Sourav Karmakar

souravkarmakar29@gmail.com

SUPPORT VECTORS: INTUITION



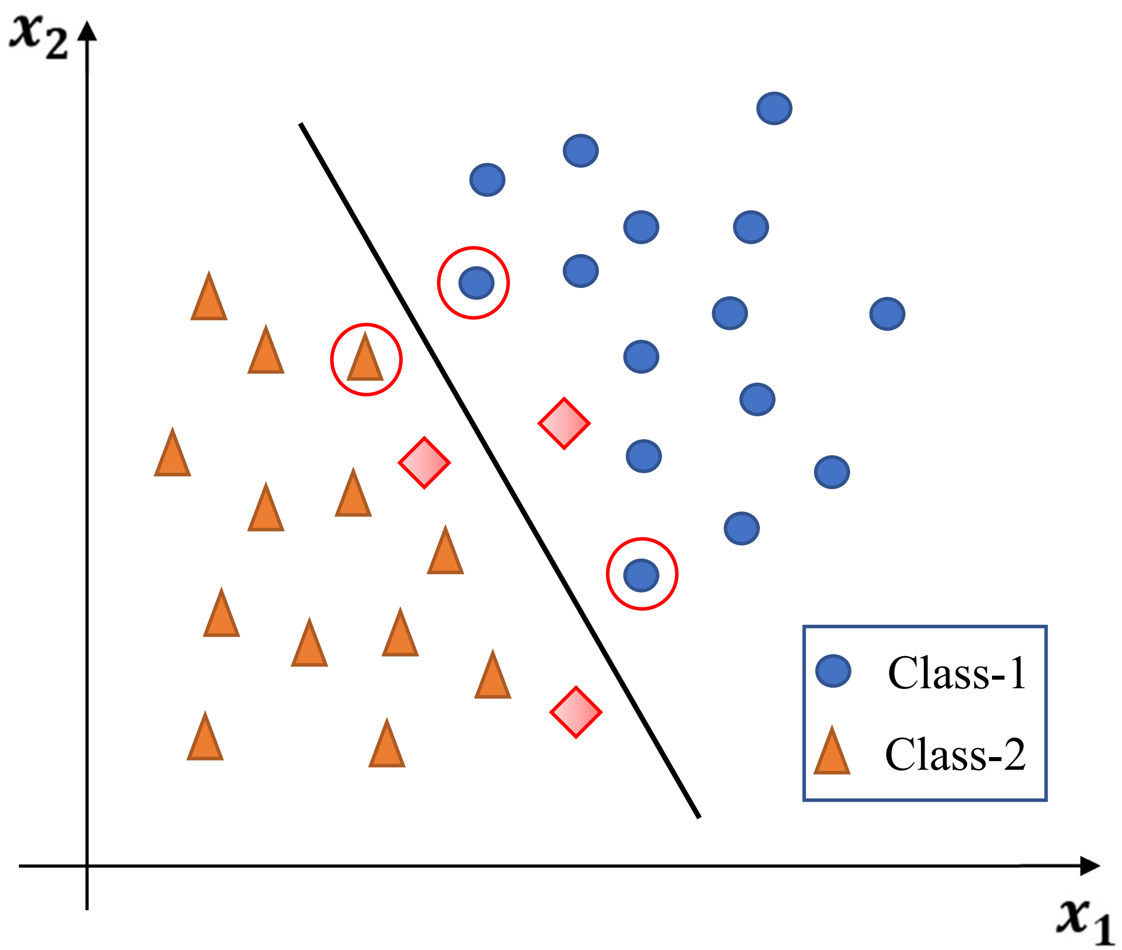
- In a binary classification problem we usually want to find a Line (Usually a hyperplane) that separates the two classes of the points.

- To choose a "good" line:
  - We have to optimize some objective function (For ex. In Logistic regression we minimized the cross-entropy loss function)
  - Usually the objective function depends on all the points.

- There can be many such lines. Hence, finding a "good" line is a difficult task. Moreover, changing the position of the training points can affect the decision plane.

- Primarily we want **least number of misclassification** of the test points. Now consider a decision plane. Which points are more likely to be misclassified?

SUPPORT VECTORS: INTUITION

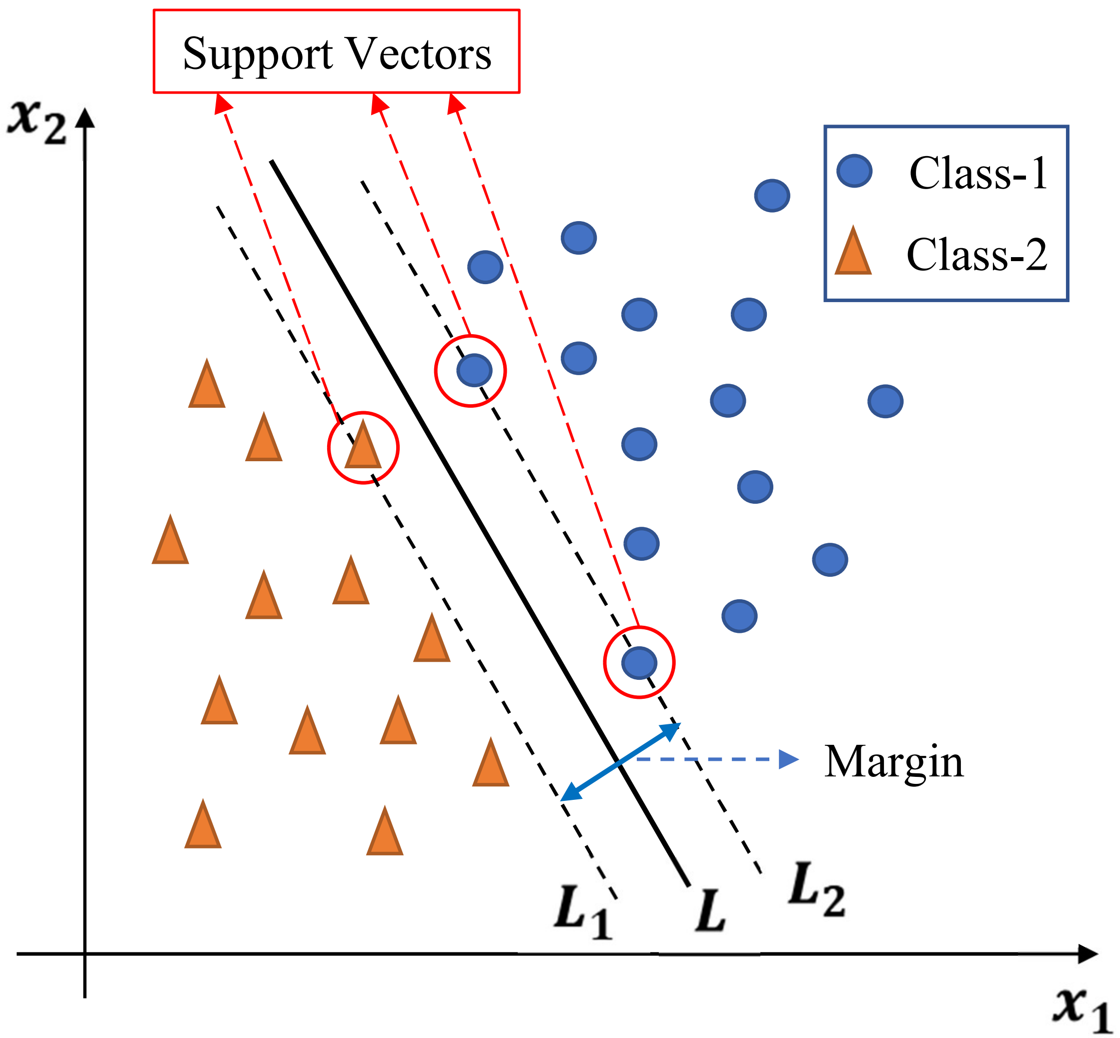


- **Answer:** The test points which are closer to the decision boundary are more likely to be misclassified.
- Hence, while designing the classifier, we need to give more emphasis on the training points which are closer to the border.
- The training points closer to the border which are most crucial to design the classifier are known as ***Support Vectors***.

- **Some Mathematical Pre-requisite:**
  - <u>Norm of a vector:</u> Let a vector $\vec{\boldsymbol{v}} = [v_1, v_2, \ v_3, \dots, v_n]^T$.
    Then its norm is defined as $\|\vec{\boldsymbol{v}}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$
  - <u>Equation of a n-dimensional Hyperplane:</u>

$$w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = a \Rightarrow \vec{\boldsymbol{w}}^T\vec{\boldsymbol{x}} = a \quad where\ \vec{\boldsymbol{w}} = [w_1, w_2, \dots, w_n]^T\ is\ the\ weight\ vector$$

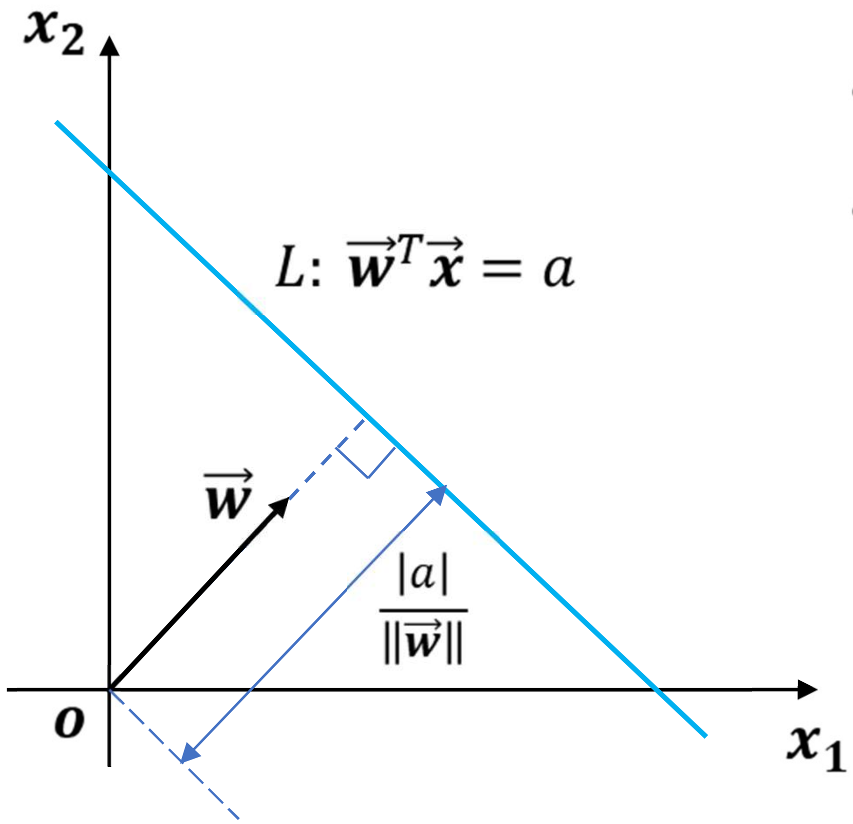
SUPPORT VECTOR MACHINE



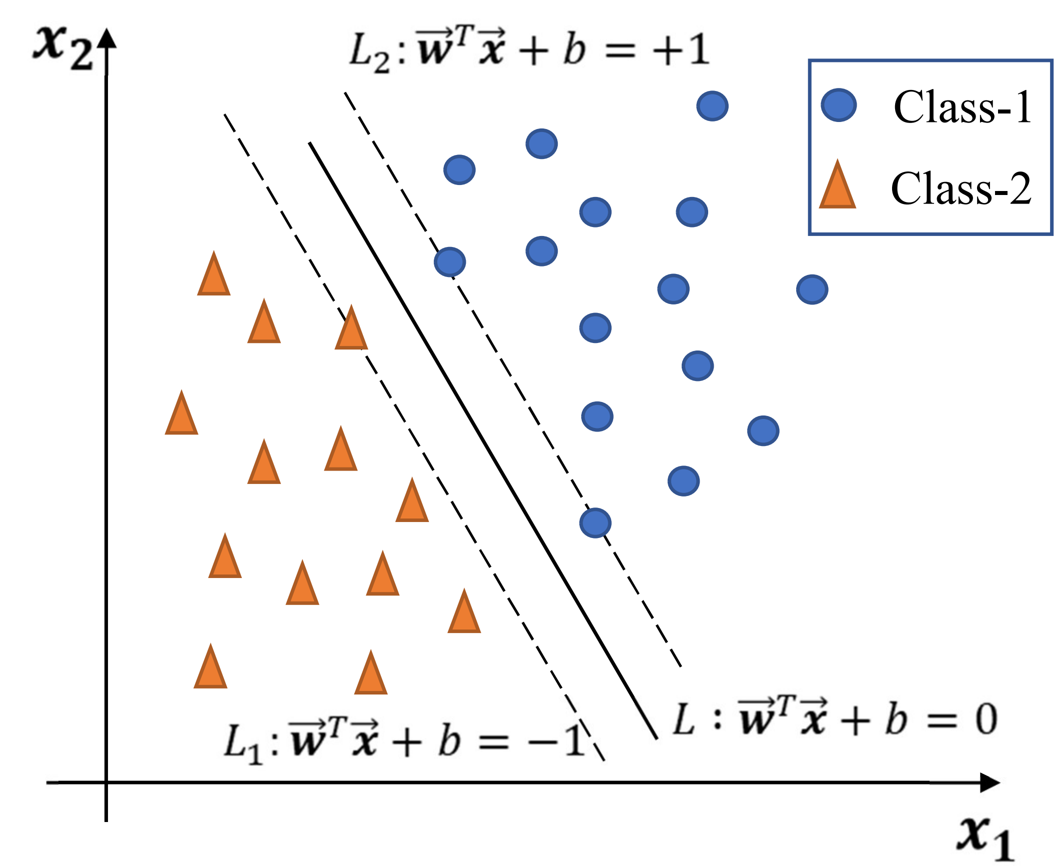
- Identify ***support vectors***, the training data points those are closer to the boundary and act as "support".
- $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are the lines (hyperplanes) defined by the support vectors.
- "Margin" is the separation (perpendicular distance) between the lines $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$.
- Our objective is to **maximize the margin**. Hence, Support Vector Machine is also called the *maximum margin classifier*.

- The decision boundary is the line (hyperplane) that pass through the middle of $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$.

SUPPORT VECTOR MACHINE



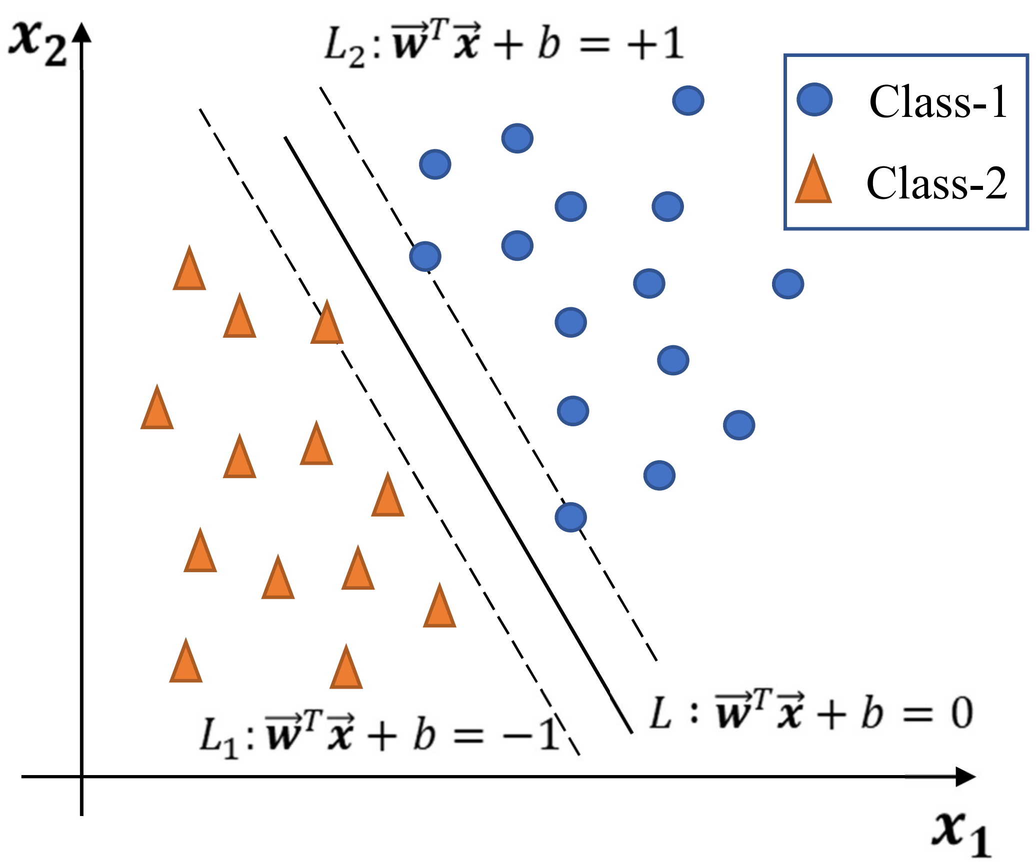
- $\vec{w}$ is the vector normal to the line (L) given by equation: $\vec{w}^T\vec{x} = a$
- The perpendicular distance of the line (L) from any point $\vec{u} = [u_1, u_2, u_3, \ldots, u_n]^T$ is given by $d(\vec{u}, L) = \frac{|\vec{w}^T\vec{u} - a|}{\|\vec{w}\|}$
- Hence, perpendicular distance of the line (L) from origin is given by: $d(0, L) = \frac{|a|}{\|\vec{w}\|}$



- Scale $\vec{w}$ and b, such that the lines are defined by these equations: $L_1: \vec{w}^T\vec{x} + b = -1,\ L_2: \vec{w}^T\vec{x} + b = +1 \quad and \quad L: \vec{w}^T\vec{x} + b = 0$
- Then $d(0, L_1) = \frac{|-1-b|}{\|\vec{w}\|}$ and $d(0, L_2) = \frac{|1-b|}{\|\vec{w}\|}$
- Hence, the margin (separation between L_1 and L_2) = $d(L_1, L_2) = \frac{2}{\|\vec{w}\|}$
- Hence, to maximize the margin, $d(L_1, L_2)$, we have to minimize: $\|\vec{w}\|$

SUPPORT VECTOR MACHINE

- We have to find $\vec{w}$ *and* b which will minimize $||\vec{w}||$. Minimizing $||\vec{w}||$ is same as minimizing $||\vec{w}||^2 = \vec{w}^T\vec{w}$



- Let C_1 is set of all points belongs to class-1 and C_2 is set of all points belongs to class-2

- Let y_i is the corresponding class label of i^{th} training point $\vec{x}_i$, such that

$$y_i = \begin{cases} +1\,, if\ \ \vec{x_i} \in C_1 \\ -1\,, if\ \ \vec{x_i} \in C_2 \end{cases}$$

- All the training points on the left of the line L_1 belongs to class-2 where as all the training points on the right of the line L_2 belongs to class-1

- Hence, the **constraint** to our optimization problem is:

$$\vec{w}^T\vec{x_i} + b \leq -1\,, \forall\, \vec{x_i} \in C_2 \quad and \quad \vec{w}^T\vec{x_i} + b \geq 1\,, \forall\, \vec{x_i} \in C_1$$

Or we can simply say:

$y_i(\vec{w}^T\vec{x_i} + b) \geq 1\,, \forall\ i \in \{1,2, \dots, m\}$, here m is the no. of training samples

SUPPORT VECTOR MACHINE

- Therefore our overall optimization problem for SVM is:

$$\underset{\vec{\boldsymbol{w}}, b}{minimize} \quad \frac{1}{2}\vec{\boldsymbol{w}}^T\vec{\boldsymbol{w}}, \qquad subjected\ to\ the\ constraints:\ y_i(\vec{\boldsymbol{w}}^T\vec{\boldsymbol{x}_i} + b) \geq 1\ , \forall\ i \in \{1,2, \dots, m\}$$

- The $\vec{w}$ and b, thus obtained defines our classifier.

- For predicting the class of a new test point $\vec{\boldsymbol{x}}$ we can use following equation:

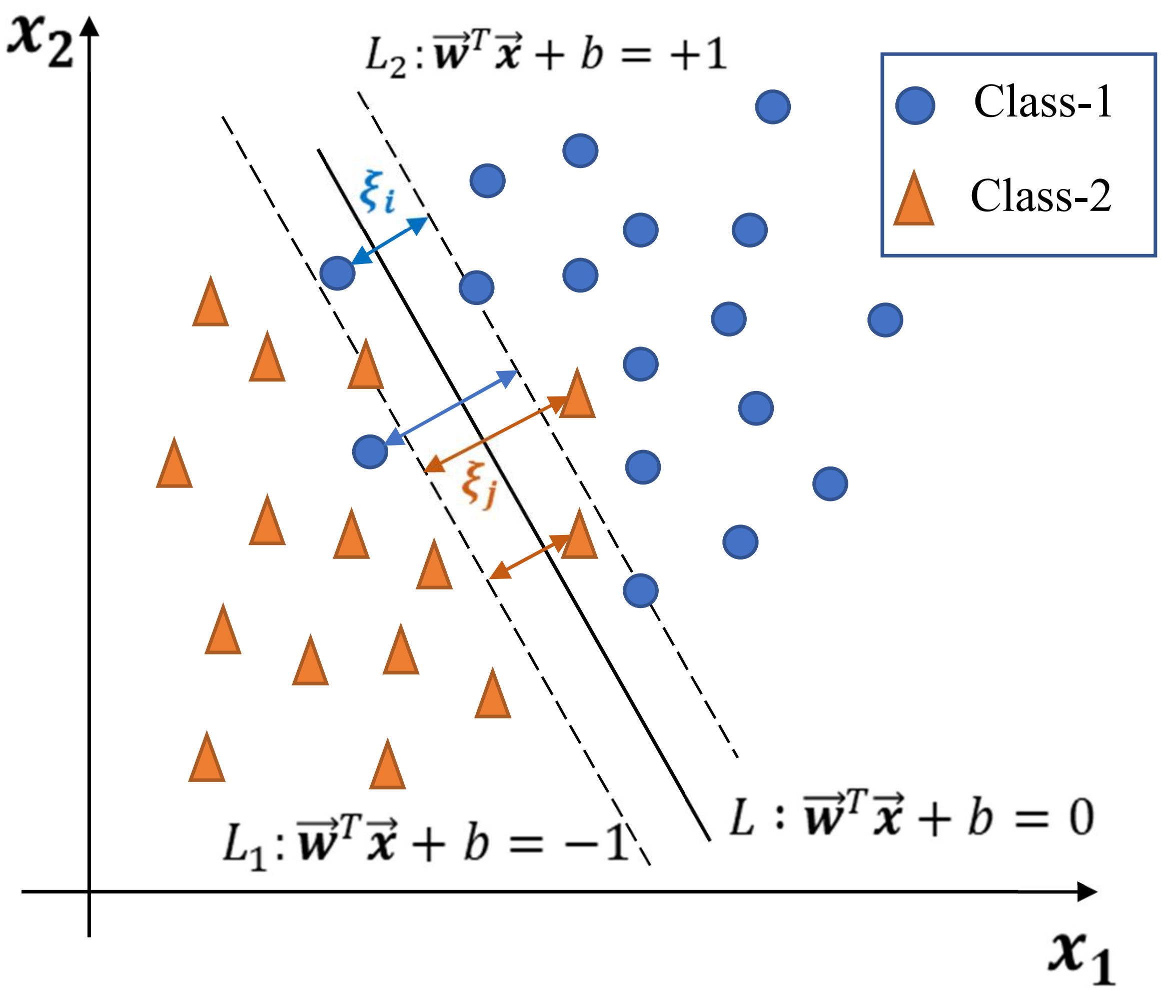
$$Class\ of\ \vec{\boldsymbol{x}} = \operatorname{sgn}(\vec{\boldsymbol{w}}^T\vec{\boldsymbol{x}} + b)$$

- Note that this is for binary classification. For multiclass classification we can again use decomposition techniques like *One-vs-All (OVA) or One-vs-Rest (OVR)*.

- This is called *Hard Margin Support Vector Machine (SVM) Classifier*, as opposed to *Soft Margin SVM Classifier*, which we shall introduce shortly.

SOFT MARGIN SVM

- Soft Margin SVM is suitable for the *non-ideal cases* where the data points are not completely separable.



- Some data points are in the wrong side of the margin. The standard approach is to allow the decision margin to make a few mistakes.
- We then pay a cost for each misclassified example, which depends on how far it is from meeting the margin requirement.
- To facilitate this, we introducing Slack variable (ξ_i) for each data points $\vec{x}_i$.
- A nonzero value of ξ_i allows $\vec{x}_i$ to not meet the margin requirement at a cost proportional to the value of ξ_i.

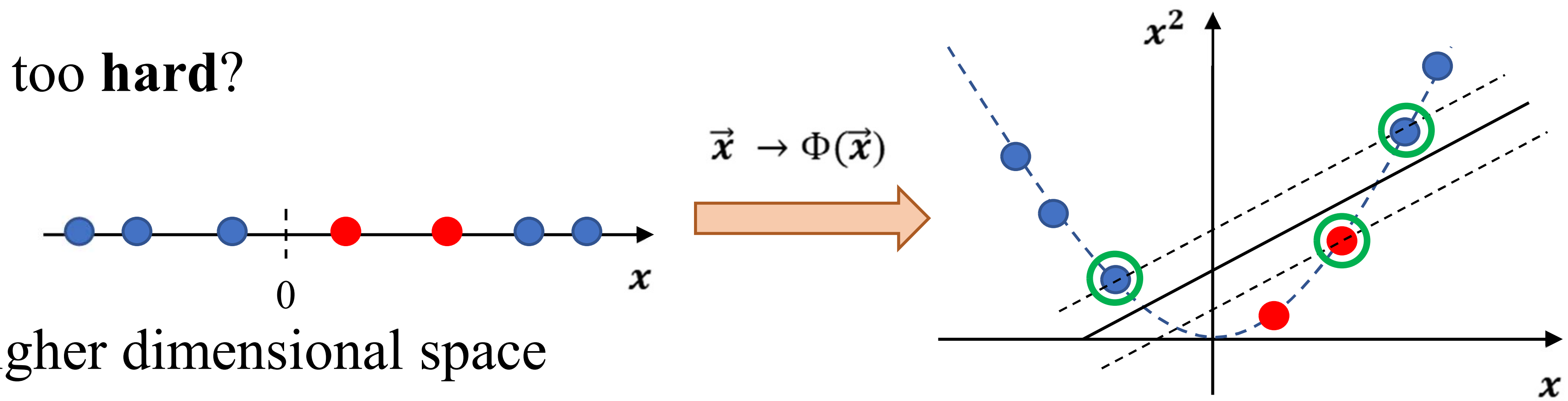
- Hence, our modified objective is:

  Find the value of $\vec{w}, b\ and\ \xi_i \geq 0$ such that:
  - $\frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{m} \xi_i$ is minimized &
  - $y_i(\vec{w}^T\vec{x_i} + b) \geq 1 - \xi_i\ , \forall i$

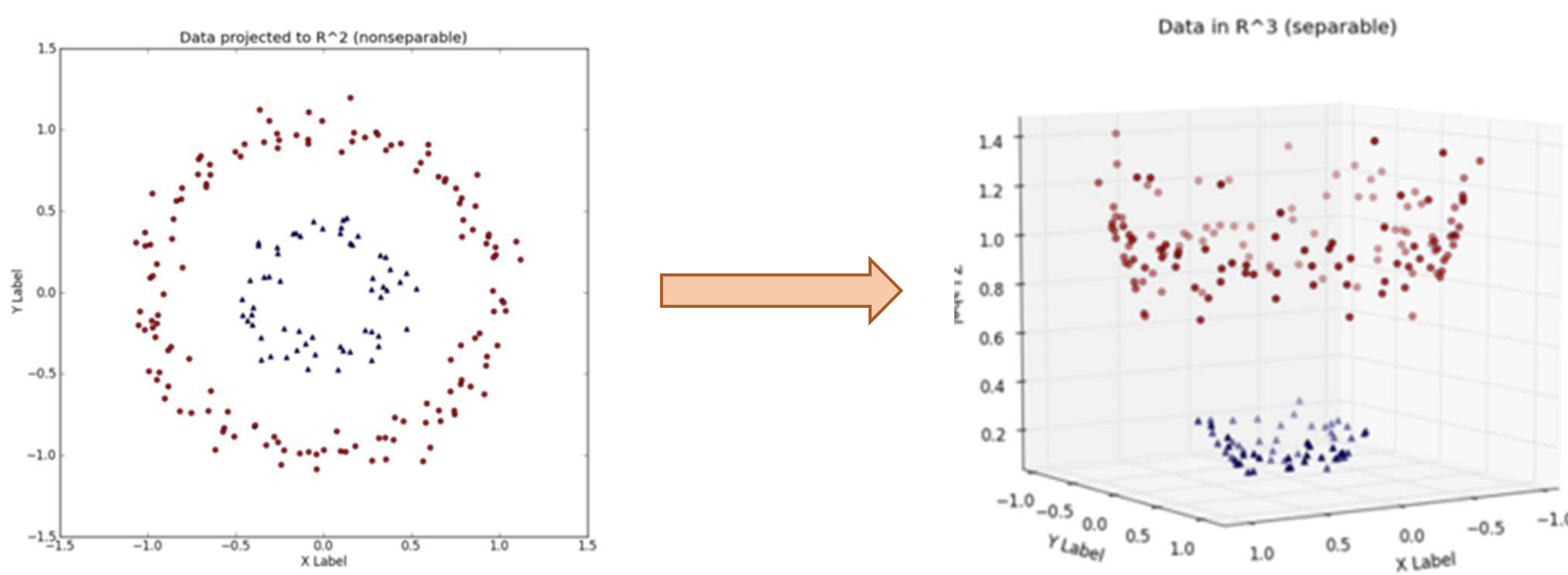
- C here is controlling parameter (user defined).
- Small $C \rightarrow$ allows large ξ_i's, thereby allowing more $\vec{x}_i$'s to slip through the margin.
- Large $C \rightarrow$ forces small ξ_i's.

NON-LINEAR SVM

- The SVM we have learned so far works great for linearly separable datasets. Hence, this is also called linear SVM.
- But what we are going to do when the dataset is too **hard**?
- How about mapping the data ($\vec{x}$) to a higher dimensional space $\Phi(\vec{x})$



- The dataset may become linearly separable in higher dimensional space and can be classified using linear SVM.
- **General idea:** the original feature space can always be mapped to some higher-dimensional feature space where the training set is almost linearly separable.



NON-LINEAR SVM: KERNEL TRICK

- Kernel function transforms the datapoints from lower dimensional space to higher dimensional space.

- There are different kinds of kernels. Few of them are mentioned below:

  - Linear Kernel: $K(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2) = \vec{\boldsymbol{x}}_1^T \, \vec{\boldsymbol{x}}_2$
  - Polynomial of degree d Kernel: $K_d(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2) = [\vec{\boldsymbol{x}}_1^T \, \vec{\boldsymbol{x}}_2 + 1]^d$
  - Radial Basis (or Gaussian) Kernel: $K_\gamma(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2) = \exp\left(-\gamma \, \left|\left|\vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_2\right|\right|^2\right)$
  - Sigmoid Kernel: $K_{a,b}(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2) = \tanh(a \, \vec{\boldsymbol{x}}_1^T \vec{\boldsymbol{x}}_2 + b)$

- There are different other kernels. User can define a novel kernel based on the requirement. However the above mentioned kernels are suitable for almost all kinds of problems and hence mostly used.

SVM: Merits & Demerits

Merits:

- SVM is a strong classifier which out performs other classifiers in many classification problems.
- Mathematically sound : A nice optimization problem which is guaranteed to converge to a single global optima.
- Can work on very high dimensional feature space as complexity doesn't depend on the dimensionality of the feature space.
- Can work on non-linearly separable cases using suitable kernel function.

Demerits:

- Tuning SVMs remains less intuitive: selecting a specific kernel function and parameters is usually done in a try-and-see manner.

Thank You