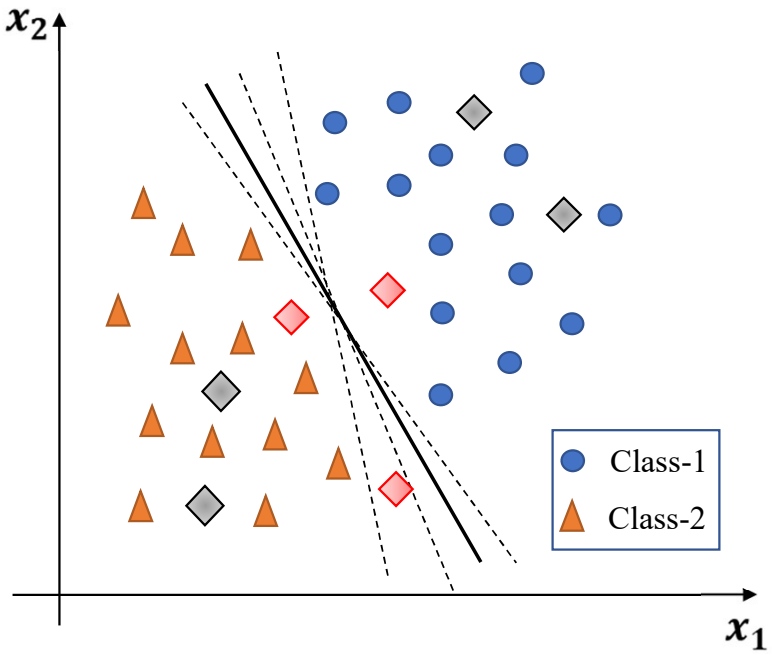
SUPPORT VECTOR MACHINE (SVM) CLASSIFIER

Sourav Karmakar

souravkarmakar29@gmail.com

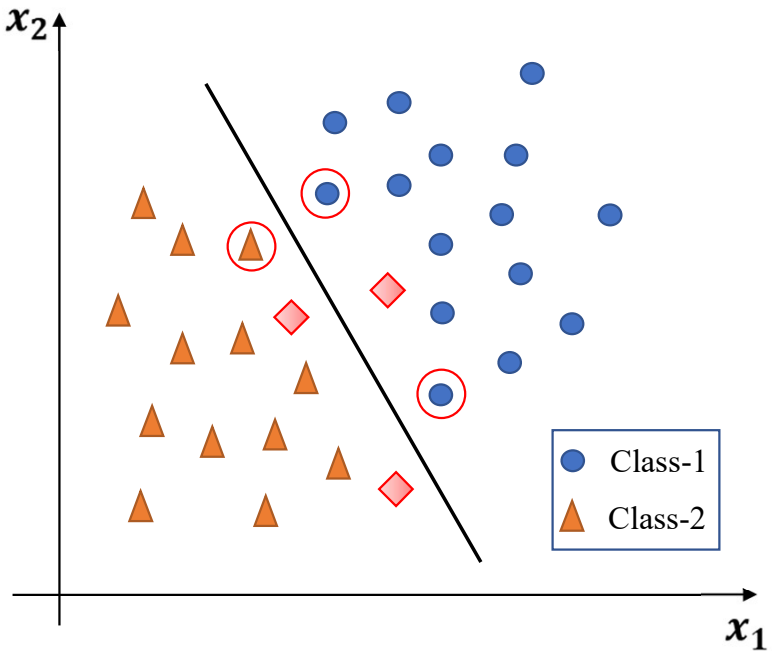
SUPPORT VECTORS: INTUITION



- 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

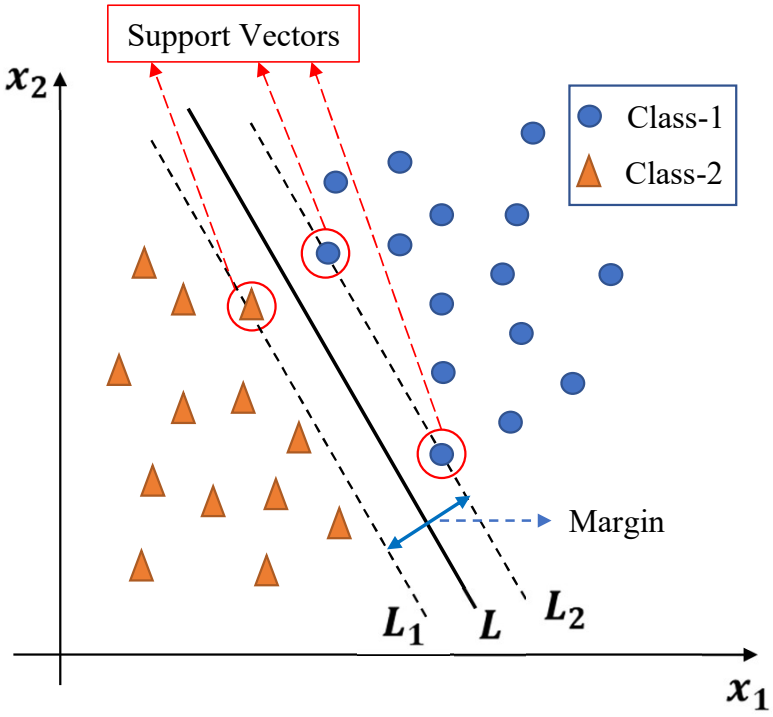


- **Answer:** The test points which are closer to the decision boundary are more likely to be misclassified.
- Hence, while designing the classifier, we need to give more emphasis on the training points which are closer to the border.
- The training points closest to the decision boundary which are most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

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

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

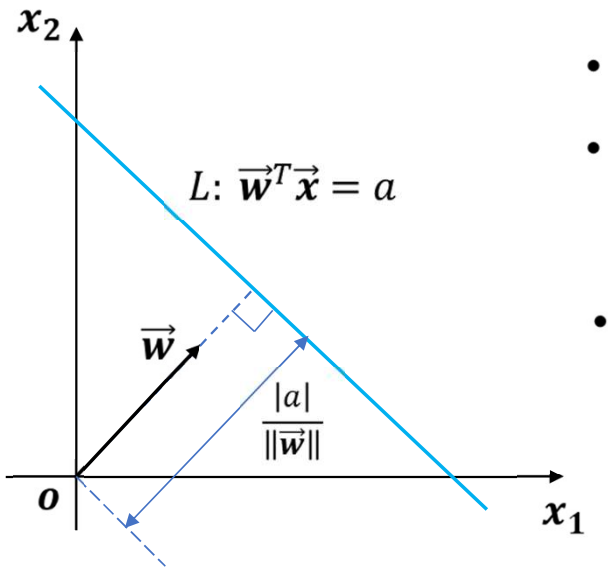
SUPPORT VECTOR MACHINE



- 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*.

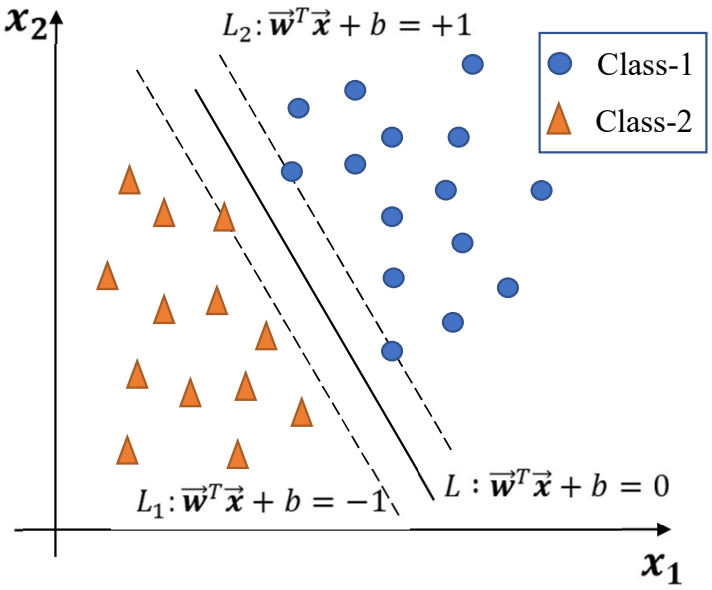
- The decision boundary is the line (hyperplane) that pass through the middle of $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$.

SUPPORT VECTOR MACHINE



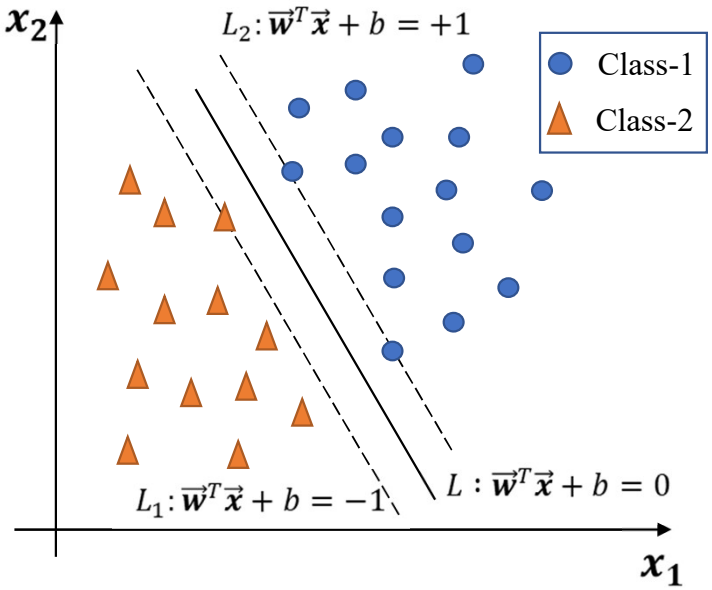
- $\vec{w}$ is the vector normal to the line (L) given by equation: $\vec{w}^T\vec{x} = a$
- The perpendicular distance of the line (L) from any point $\vec{u} = [u_1, u_2, u_3, \dots, u_n]^T$ is given by $d(\vec{u}, L) = \frac{|\vec{w}^T\vec{u} - a|}{\|\vec{w}\|}$
- Hence, perpendicular distance of 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: $L_1: \vec{w}^T\vec{x} + b = -1$, $L_2: \vec{w}^T\vec{x} + b = +1$ *and* $L: \vec{w}^T\vec{x} + b = 0$
- 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:

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

SUPPORT VECTOR MACHINE

- Therefore our overall optimization problem for SVM is:

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

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

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

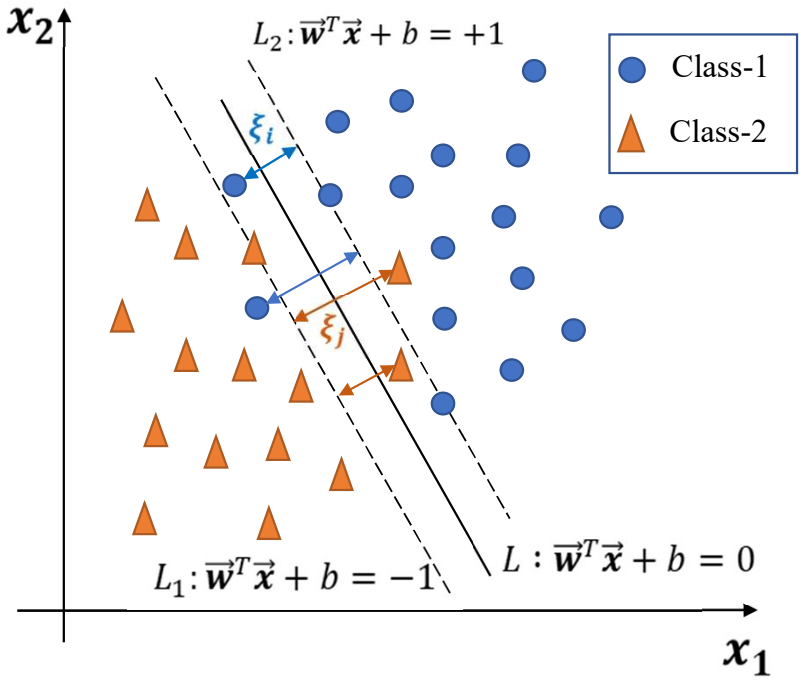
$$Class\ of\ \vec{x} = \text{sgn}(\vec{w}^T\vec{x} + b)$$

- 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.

SOFT MARGIN SVM

For **Hard Margin** SVM the constrains are: $y_i\left(\vec{\boldsymbol{w}}^{\boldsymbol{T}}\vec{\boldsymbol{x}_i} + b\right) \geq 1, \forall\, i$ and
for **Soft Margin** SVM the constraints are: $y_i\left(\vec{\boldsymbol{w}}^{\boldsymbol{T}}\vec{\boldsymbol{x}_i} + b\right) \geq 1 - \xi_i\,, \forall\, i$

These ξ_i are called slack variables and defined by:

$$\xi_i = \max\left(0, 1\ - y_i\left(\vec{\boldsymbol{w}}^{\boldsymbol{T}}\vec{\boldsymbol{x}_i} + b\right)\right) = \max\left(0, 1\ - y_i \cdot f(\vec{\boldsymbol{x}_i})\right)\ \ ;\ where\ f(\vec{\boldsymbol{x}_i}) = \left(\vec{\boldsymbol{w}}^{\boldsymbol{T}}\vec{\boldsymbol{x}_i} + b\right)$$

Hinge Loss: The formula for hinge loss $L_H\left(y_i, f(\vec{\boldsymbol{x}_i})\right)$ is: $L_H\left(y_i, f(\vec{\boldsymbol{x}_i})\right) = \max\left(0, 1\ - y_i \cdot f(\vec{\boldsymbol{x}_i})\right)$

Scenario-1: Correctly classified and outside the margin

This is the ideal case. The point is on the correct side of the decision boundary and far enough away from it (beyond the margin).

In this case, for a positive class $(y_i = +1)$, $f(\vec{\boldsymbol{x}_i})$ is large positive number, ideally ≥ 1.

Example: $f(\vec{\boldsymbol{x}_i})$ = 2.5 , $\therefore\ 1\ - y_i \cdot f(\vec{\boldsymbol{x}_i}) = -1.5 \Rightarrow L_H = \max(0, -1.5) = 0$

For a negative class $(y_i = -1)$, $f(\vec{\boldsymbol{x}_i})$ is large negative number, ideally ≤ -1.

Example: $f(\vec{\boldsymbol{x}_i})$ = -3.0 , $\therefore\ 1\ - y_i \cdot f(\vec{\boldsymbol{x}_i}) = -2.0 \Rightarrow L_H = \max(0, -2.0) = 0$

So, Hinge Loss is 0 when the data points are correctly classified and outside the margin

SOFT MARGIN SVM

Scenario-2: Correctly classified but within the margin

The point is on the correct side of the decision boundary, but it's too close, falling within the margin.

In this case, for a positive class $(y_i = +1)$, $f(\overrightarrow{x_i})$ is positive, but between 0 to 1.

Example: $f(\overrightarrow{x_i})$ = 0.5 , $\therefore 1 - y_i \cdot f(\overrightarrow{x_i}) = 0.5 \Rightarrow L_H = \max(0, 0.5) = 0.5$

For a negative class $(y_i = -1)$, $f(\overrightarrow{x_i})$ is negative number, but between -1 to 0.

Example: $f(\overrightarrow{x_i})$ = -0.8 , $\therefore 1 - y_i \cdot f(\overrightarrow{x_i}) = 0.2 \Rightarrow L_H = \max(0, 0.2) = 0.2$

If $0 < y_i \cdot f(\overrightarrow{x_i}) < 1$, the hinge loss is positive. This means even if the point is technically on the right side, it's within thee margin, so there's a penalty. The closer it is to the decision boundary (or the further it is into the margin), the higher the loss.

Scenario-3: Incorrectly classified

The point is on the wrong side of the decision boundary.

In this case, for a positive class $(y_i = +1)$, $f(\overrightarrow{x_i})$ is negative.

Example: $f(\overrightarrow{x_i})$ = -0.6 , $\therefore 1 - y_i \cdot f(\overrightarrow{x_i}) = 1.6 \Rightarrow L_H = \max(0, 1.6) = 1.6$

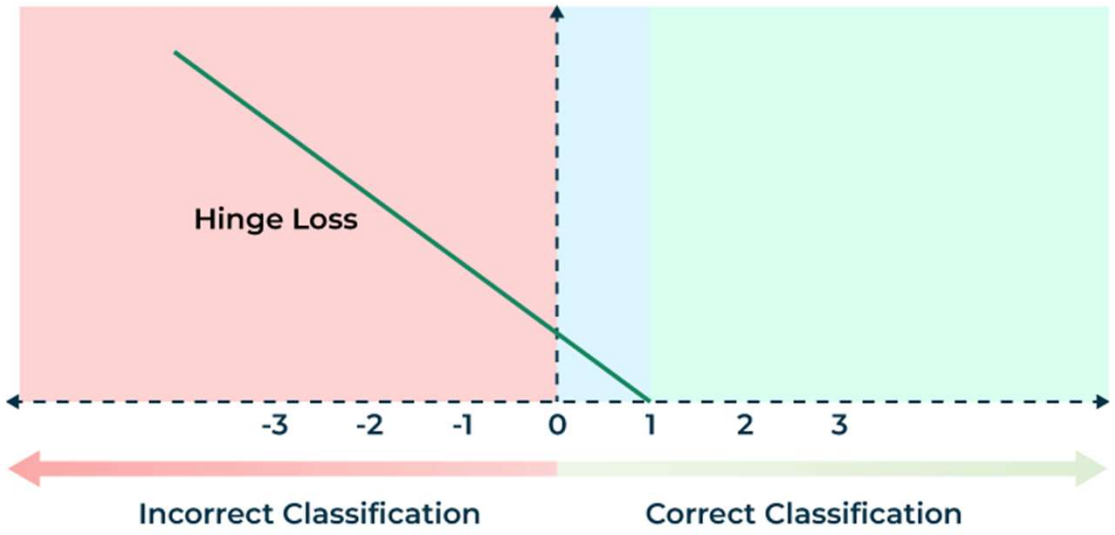
In this case, for a negative class $(y_i = -1)$, $f(\overrightarrow{x_i})$ is positive.

Example: $f(\overrightarrow{x_i})$ = 1.2 , $\therefore 1 - y_i \cdot f(\overrightarrow{x_i}) = 2.2 \Rightarrow L_H = \max(0, 2.2) = 2.2$

This mans, as $y_i \cdot f(\overrightarrow{x_i})$ become more negative, the Hinge loss linearly increases. i.e. There's a significant penalty for misclassifications.

SOFT MARGIN SVM

Hinge Loss: $L_H\left(y_i, f(\vec{x_i})\right) = \max\left(0, 1 - y_i \cdot f(\vec{x_i})\right)$



Now, in soft margin SVM optimization problem, the objective function is:

$$\min_{\vec{w},b} \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i \Rightarrow \min_{\vec{w},b} \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\vec{w}^T\vec{x_i} + b\right)\right)$$

This term is inversely related to the margin width. Minimizing it means maximizing the margin

This term penalizes the mis-classifications and the points that are within the margin.

SOFT MARGIN SVM

The parameter $\boldsymbol{C}$ is called the regularization / controlling parameter and is user defined

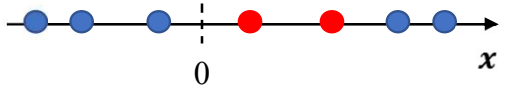
- Large $\boldsymbol{C}$: Stronger penalty for Hinge Loss. The SVM will try very hard to classify all training points correctly, even if it means a smaller margin (prone to overfitting).
- Small $\boldsymbol{C}$: Weaker penalty for Hinge Loss. The SVM allows more misclassifications and margin violations to achieve a larger, more generalizable margin (stronger regularization).

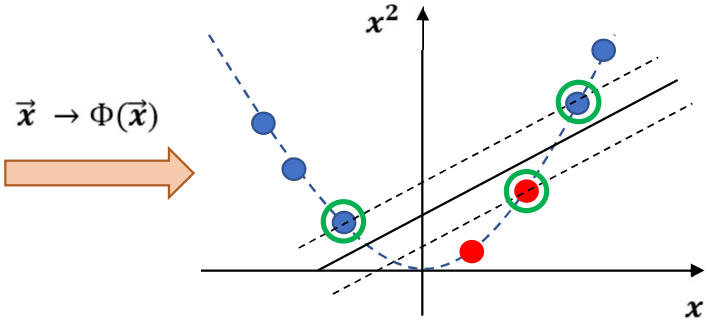
Key Characteristics of Hinge Loss:

- **Margin oriented:** Directly promotes the maximum margin principle of SVMs. It penalizes points that are not sufficiently outside margin or miss-classified.
- **Zero loss for well classified points:** If a point is correctly classified and is outside the margin, its loss is zero. This means the model doesn't need to work on these "easy" points anymore during optimization.
- **Linear penalty:** The penalty for miss-classified points or points inside the margin increases linearly.
- **Non-differentiable at $\boldsymbol{y_i \cdot f(\overrightarrow{x_i}) = 1}$**: The "hinge" or "kink" at this point means it's not smoothly differentiable (like LASSO regularization), which can pose challenges for some optimization algorithms (but specialized methods like sub-gradient descent or quadratic programming handle this).
- **Convex:** Despite the non-differentiability, hinge loss is a convex function, which guarantees that optimization algorithms can find a global minimum.

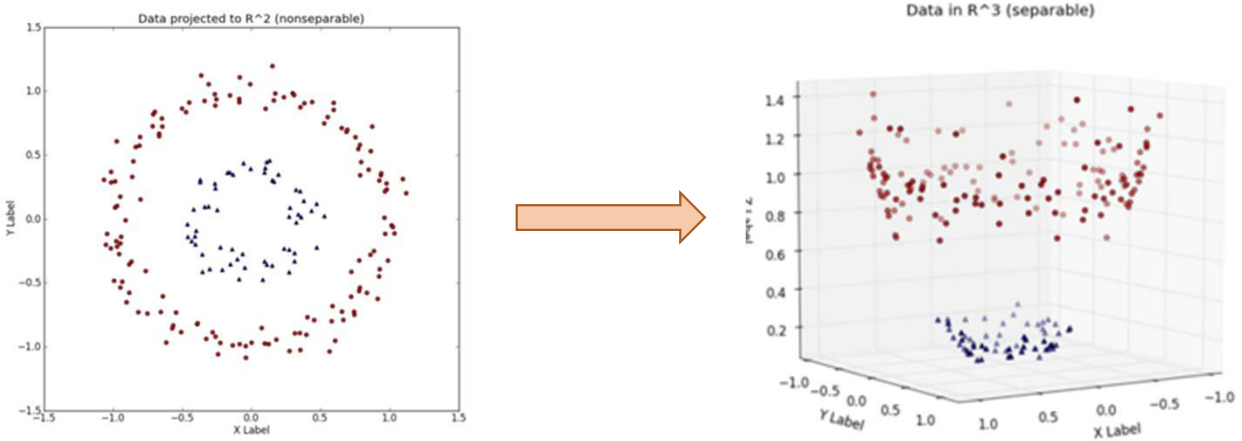
NON-LINEAR SVM

- 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: $K(\vec{x}_1, \vec{x}_2) = \vec{x}_1^T \vec{x}_2$
  - Polynomial Kernel of degree d: $K_{\gamma,r,d}(\vec{x}_1, \vec{x}_2) = (\gamma \cdot \vec{x}_1^T \vec{x}_2 + r)^d$
  - Radial Basis Function (RBF) or Gaussian Kernel: $K_\gamma(\vec{x}_1, \vec{x}_2) = exp(-\gamma ||\vec{x}_1 - \vec{x}_2||^2)$
  - Sigmoid Kernel: $K_{\gamma,r}(\vec{x}_1, \vec{x}_2) = \tanh(\gamma \cdot \vec{x}_1^T \vec{x}_2 + r)$
- 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:

- High Training Time: The core of SVM training involves solving a quadratic programming (QP) problem. The time complexity of standard SVM implementations is typically between $O(n^2)$ and $O(n^3)$, where n is the number of training samples. This makes SVMs computationally very expensive and slow for large datasets (e.g., millions of samples). Hence, it is not suitable for "Big Data".
- SVMs, especially with non-linear kernels, often require storing a kernel matrix, which grows with the square of the number of samples (n^2). This can lead to significant memory overhead.
- Support vector machines are prone to noise. If the data points are noisy or some the data points are mis-labelled then it completely jeopardizes the decision boundary and the margin.

Thank You