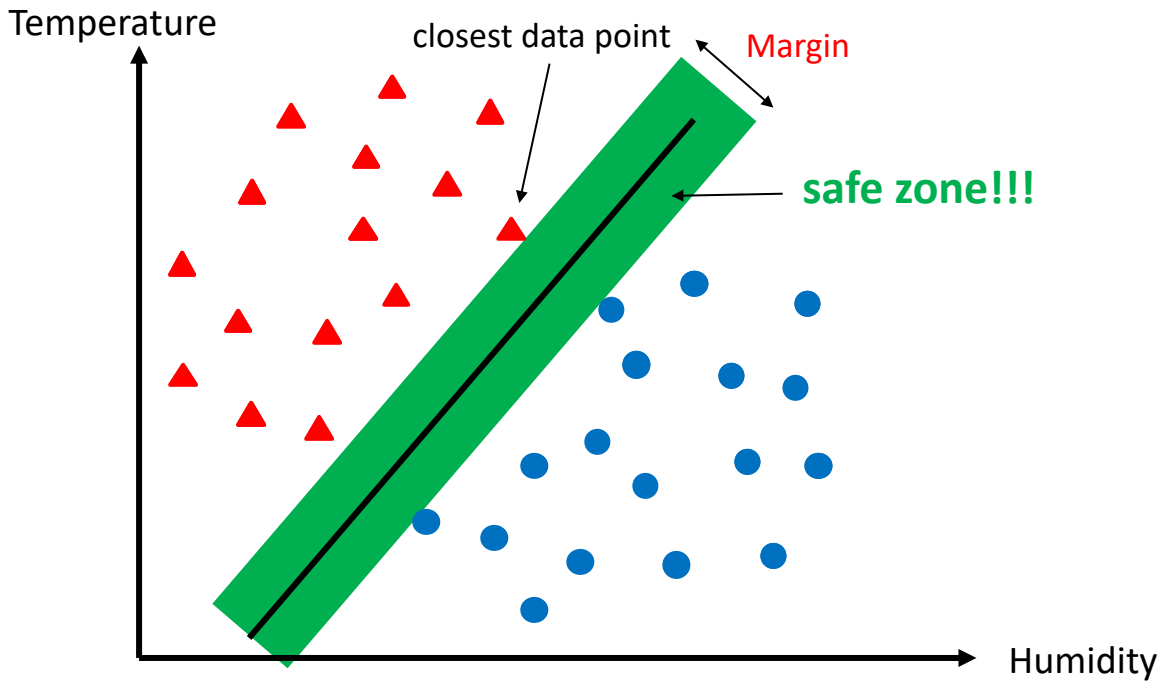
Advanced Machine Learning and Deep Learning

Dr. Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

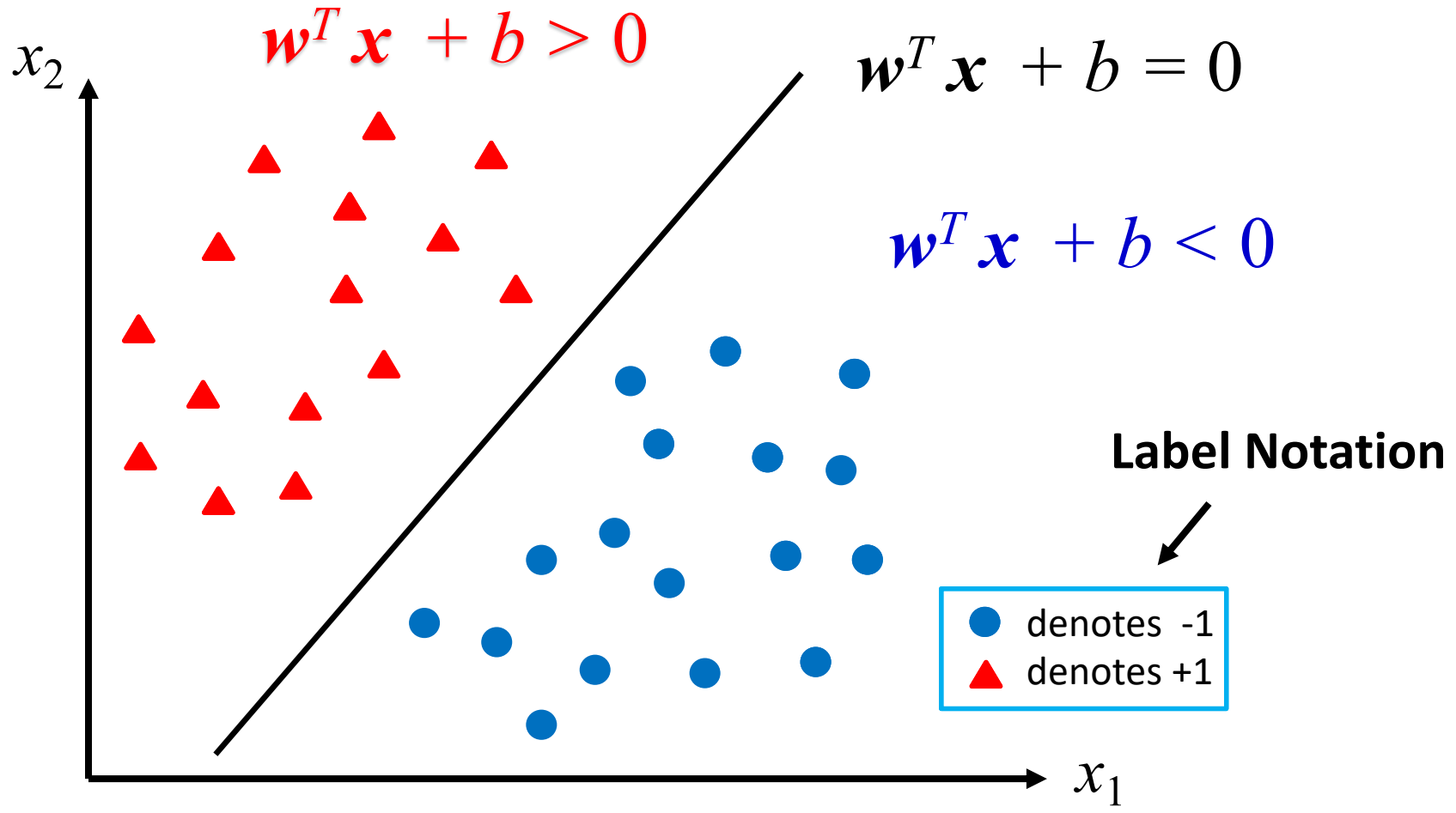
Support Vector Machine (SVM)

Review

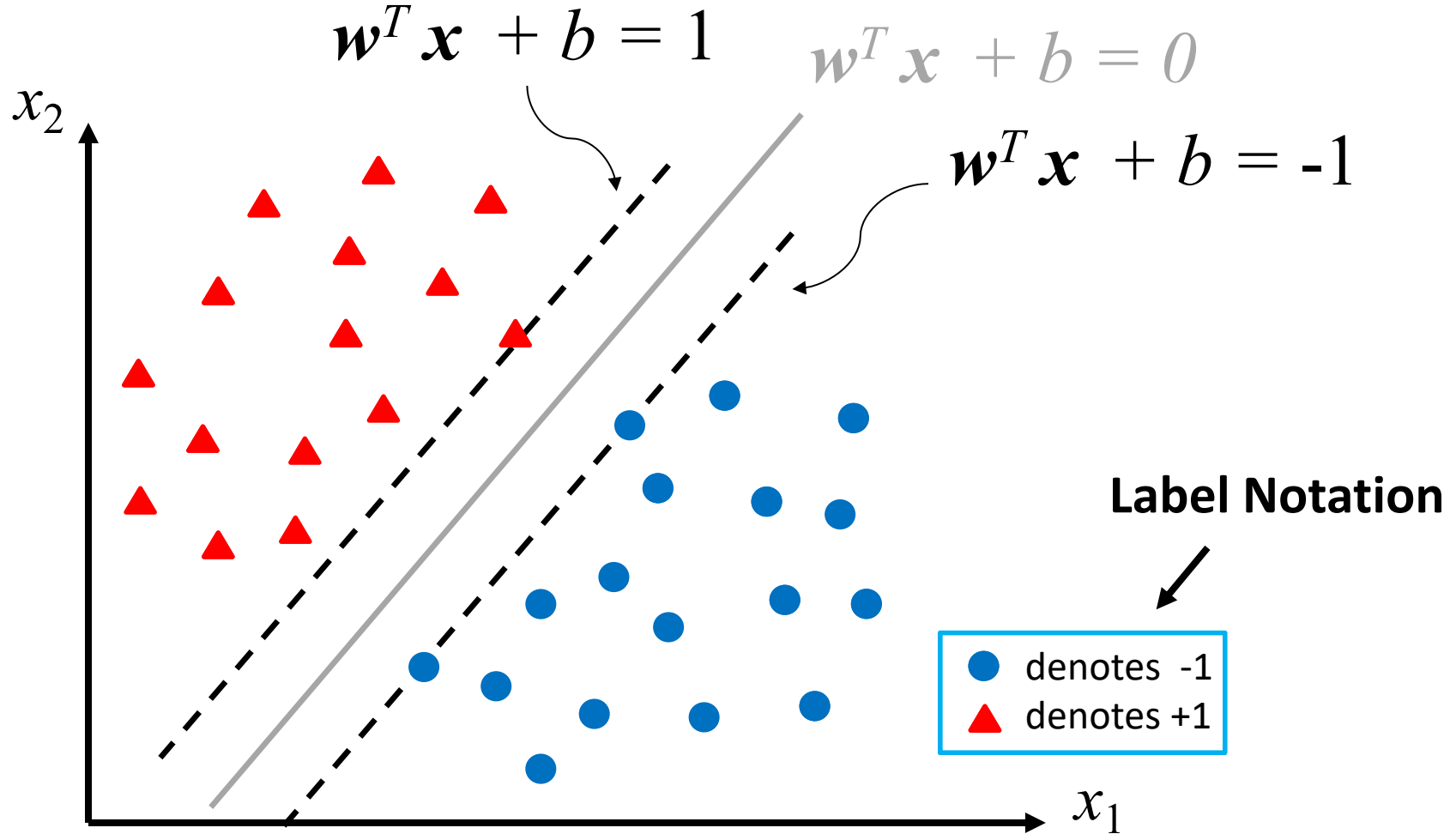
- The best classifier is the one with the **Maximum Margin**.
- **Margin** is widest boundary area before hitting a data point.



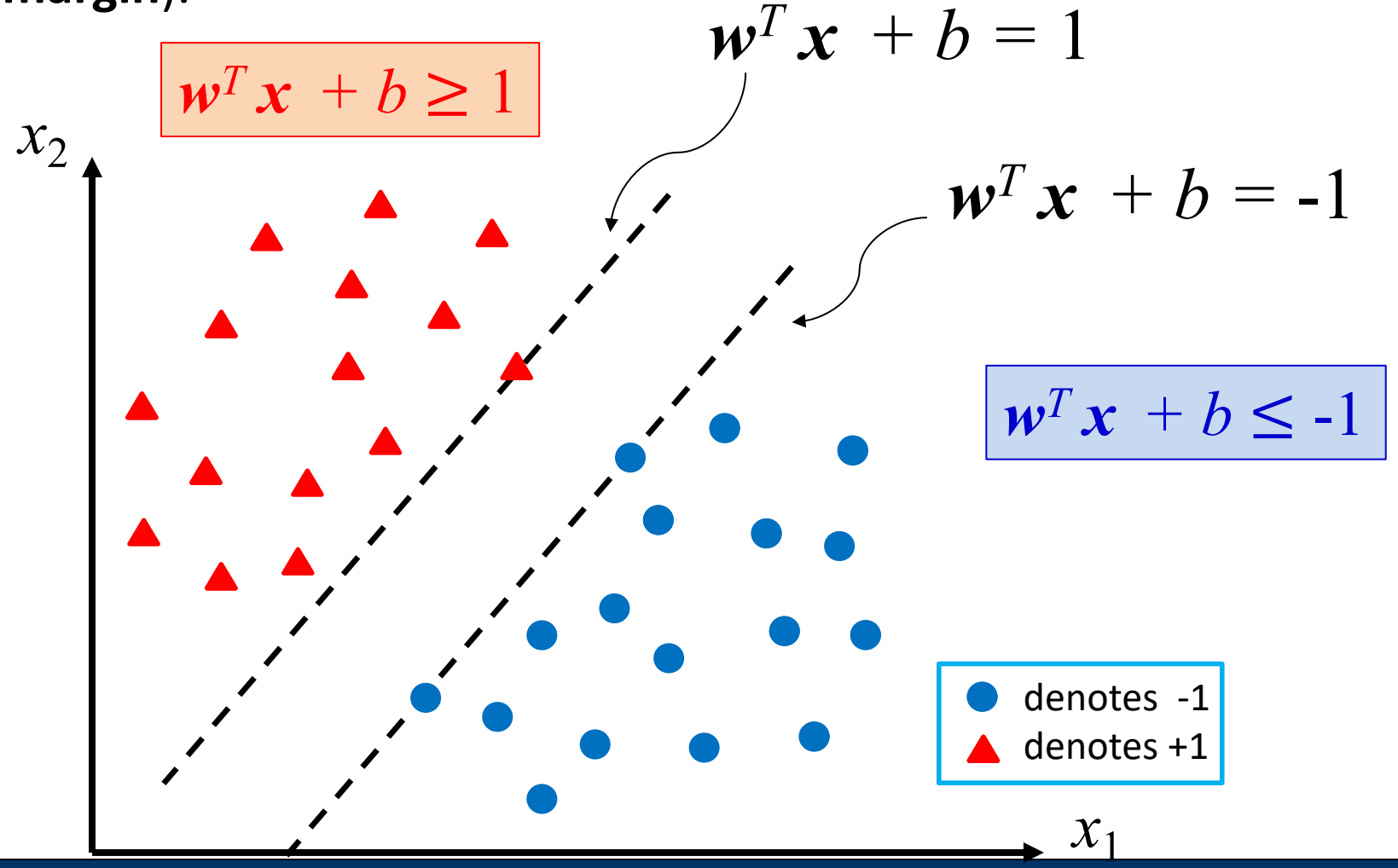
- Given a set of training data points, each point will be considered as a ***m-dimensional vector*** in space. We are looking for ***a line whose value is positive for red samples, and negative for blue samples.***



Let's do even better: We can find **two parallel lines** (rather than just one line) that separate the two classes of data! Red samples are ABOVE the TOP line, and blue samples are BELOW the BOTTOM line. These two hyperplanes can be described by the following equations:



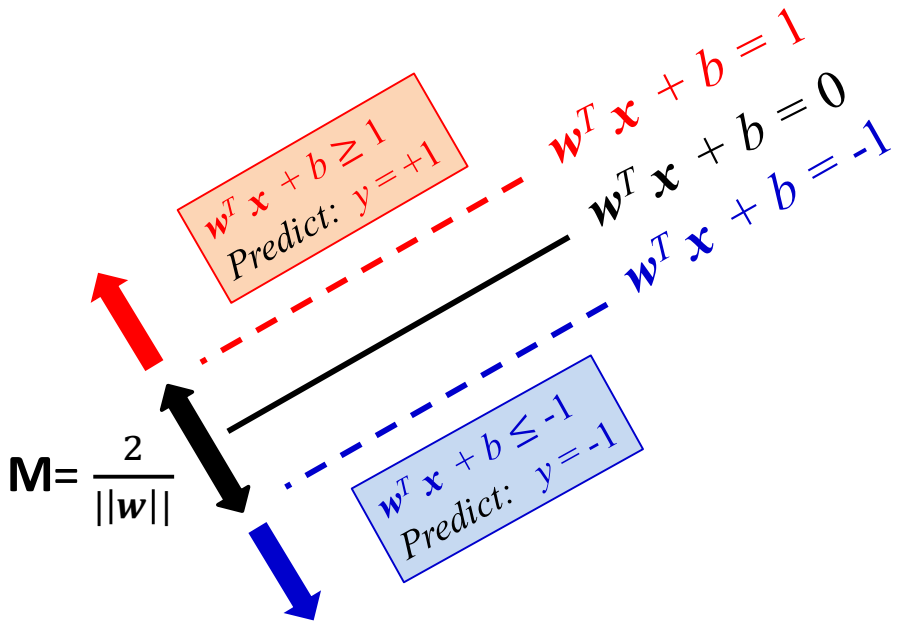
Now, Let's do even better than better: Now, let's find the **two parallel lines** that separate the two classes, **AND** the distance between them is as large as possible (**maximum margin**).



SVM Classifier

- ***Formulation:***

$$\begin{cases} \text{Maximize } \dfrac{2}{\|\boldsymbol{w}\|} \\ \text{Such that:} \\ \text{for } y = +1,\ \boldsymbol{w}^T\boldsymbol{x} + b \geq 1 \\ \text{for } y = -1,\ \boldsymbol{w}^T\boldsymbol{x} + b \leq -1 \end{cases}$$

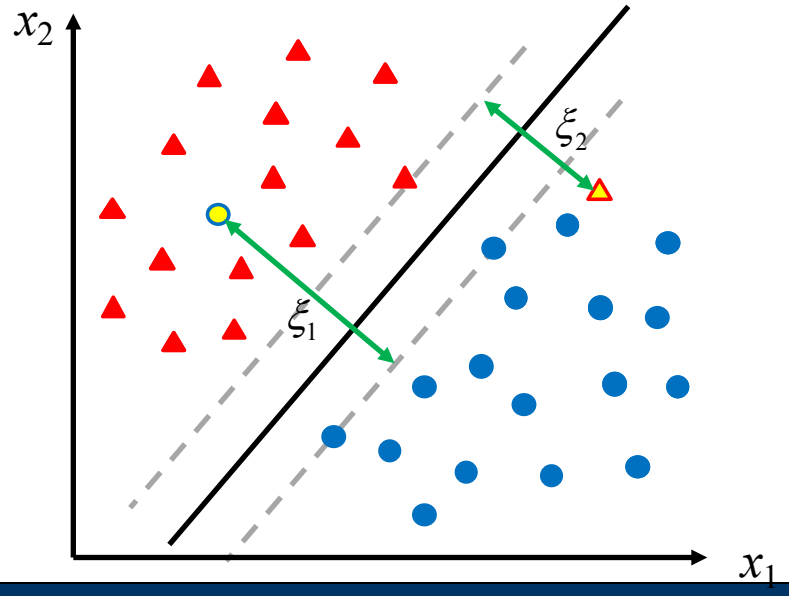


- **Hard Margin (our previous formula):**

$$\begin{cases} \text{Minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{Subject to: } y_i\,(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \\ \text{for } i = 1, 2, \ldots, n \end{cases}$$

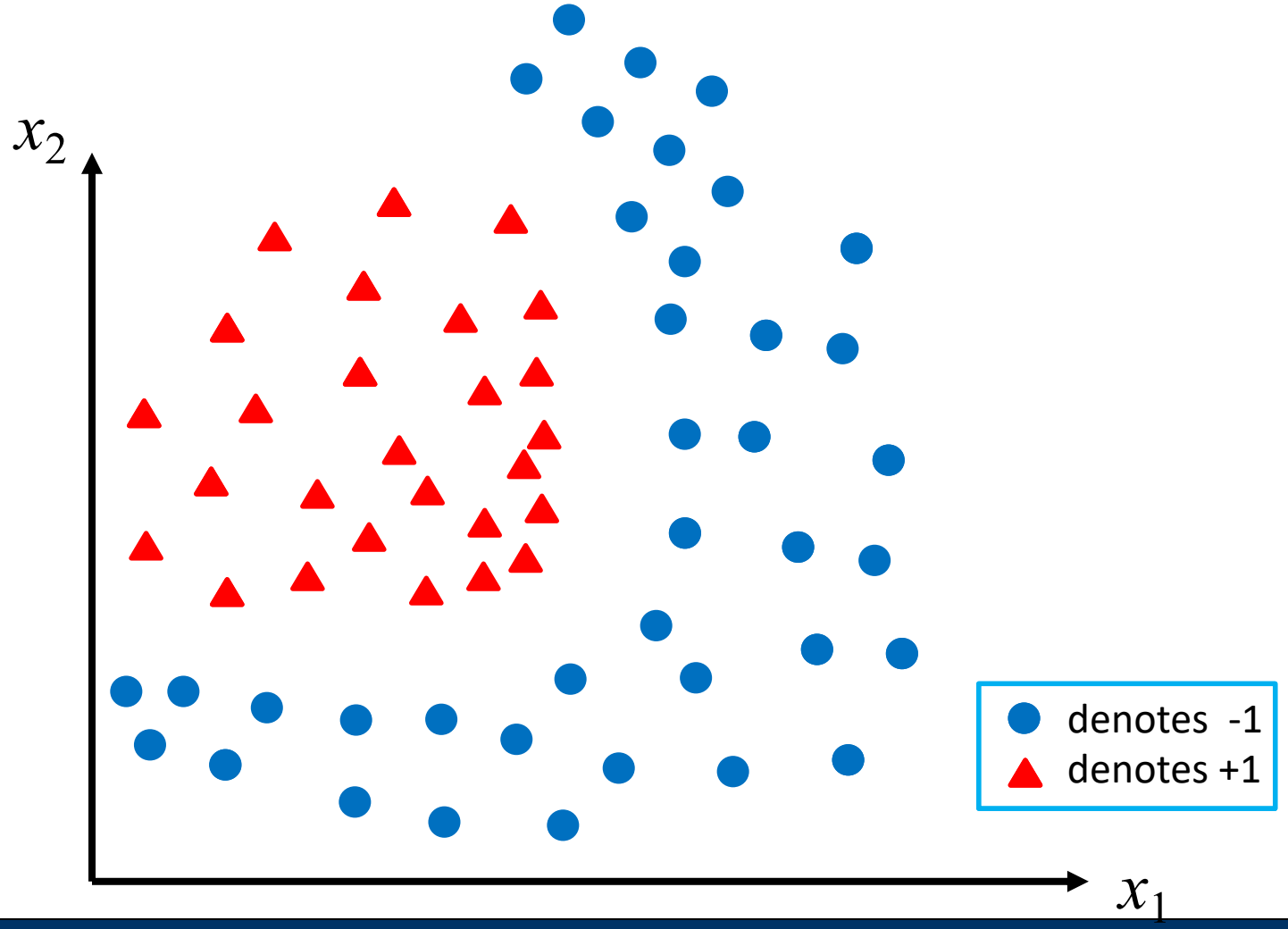
- **Soft Margin:**

$$\begin{cases} \text{Minimize } \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum \xi_i\right) \\ \text{Subject to: } y_i\,(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i \\ \text{for } i = 1, 2, \ldots, n \end{cases}$$

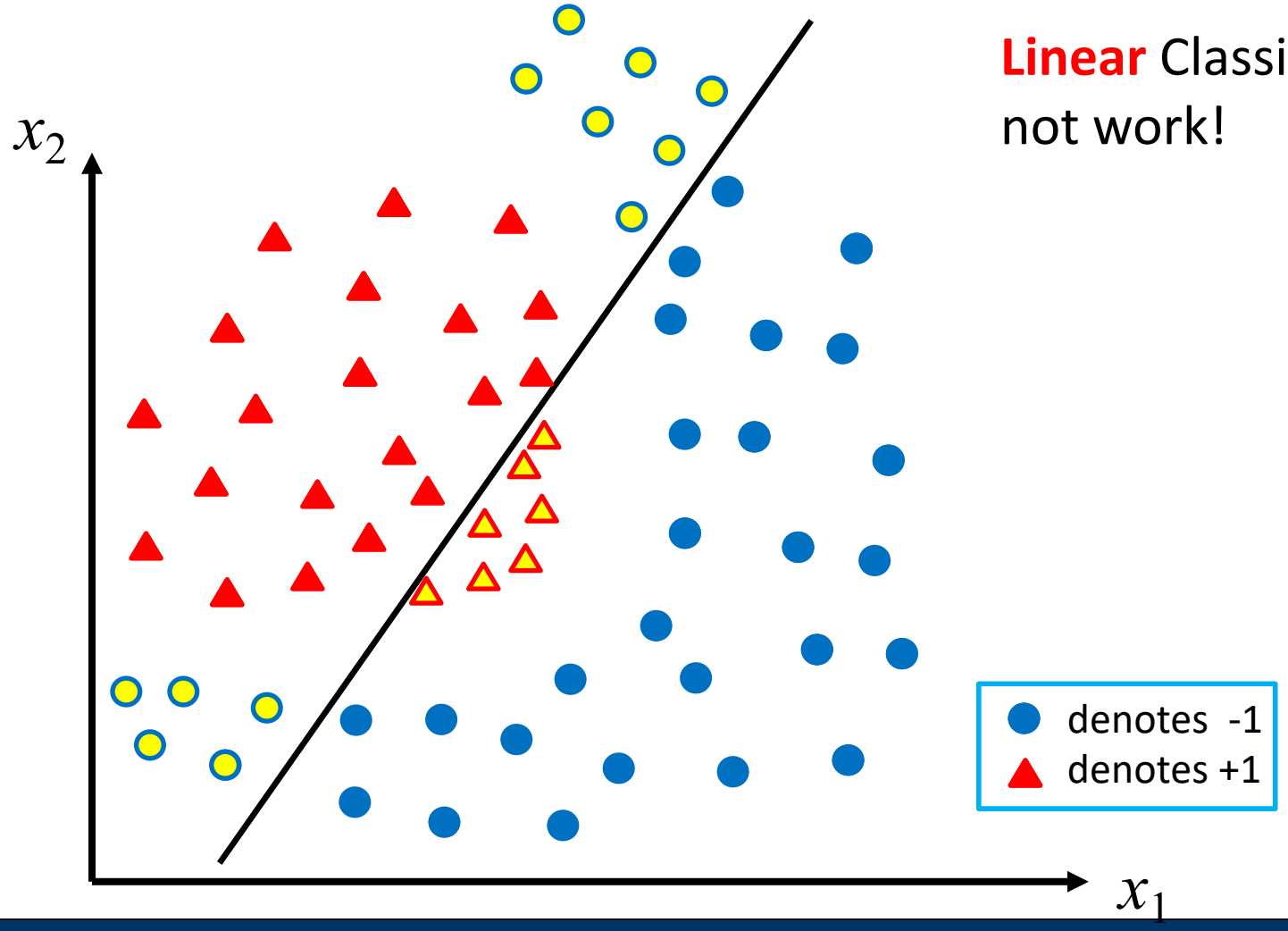


Non-Linear SVM and Kernel Trick

What if the data is not linearly separable!!?

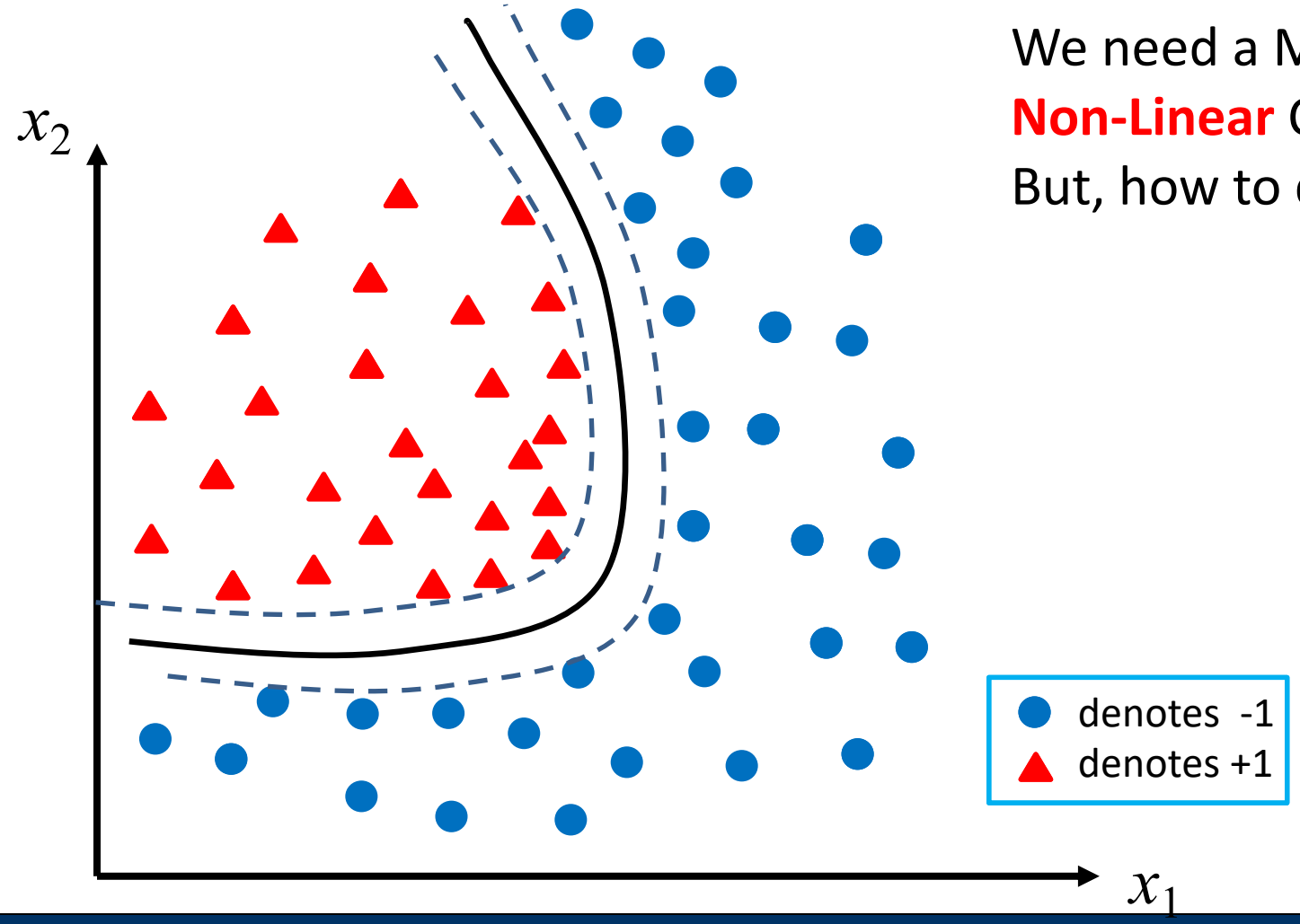


What if the data is not linearly separable!!?



Linear Classifier does not work!

What if the data is not linearly separable!!?

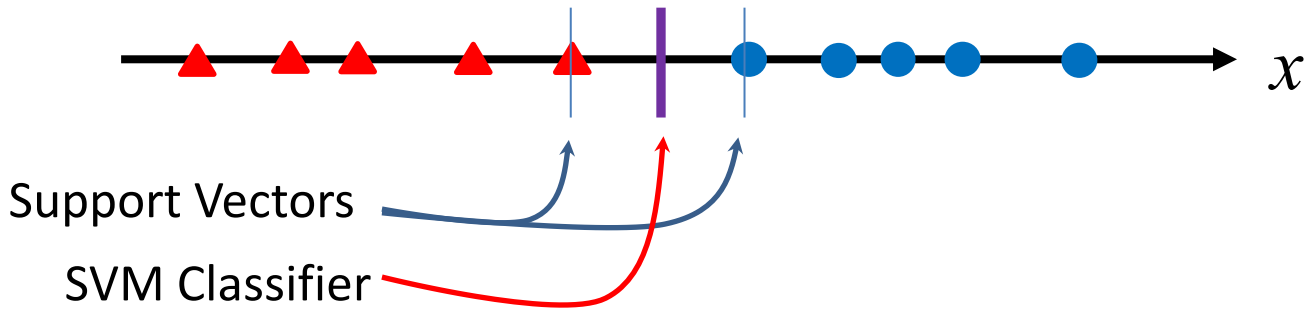


We need a Maximum-Margin **Non-Linear** Classifier!

But, how to do that?

A Simple Example in 1-D

- In this example, the Data points are linearly separable:



- In this example, the Data points are **NOT** linearly separable:

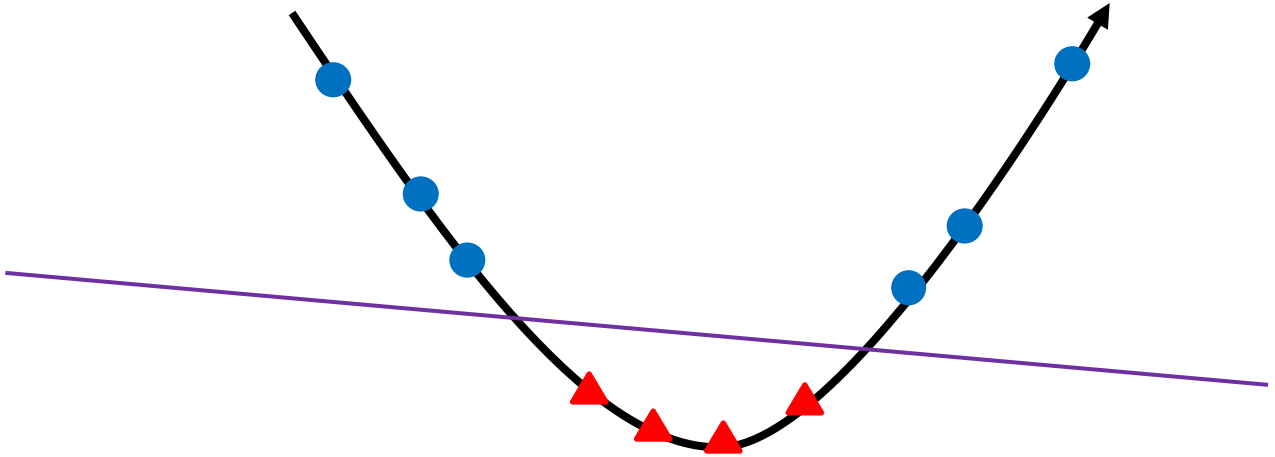
- **How to classify this dataset!!?**

Ref: Douglas Eck, Music and Machine Learning, University of Montreal

A Simple Example in 1-D

- Data points are NOT linearly separable:

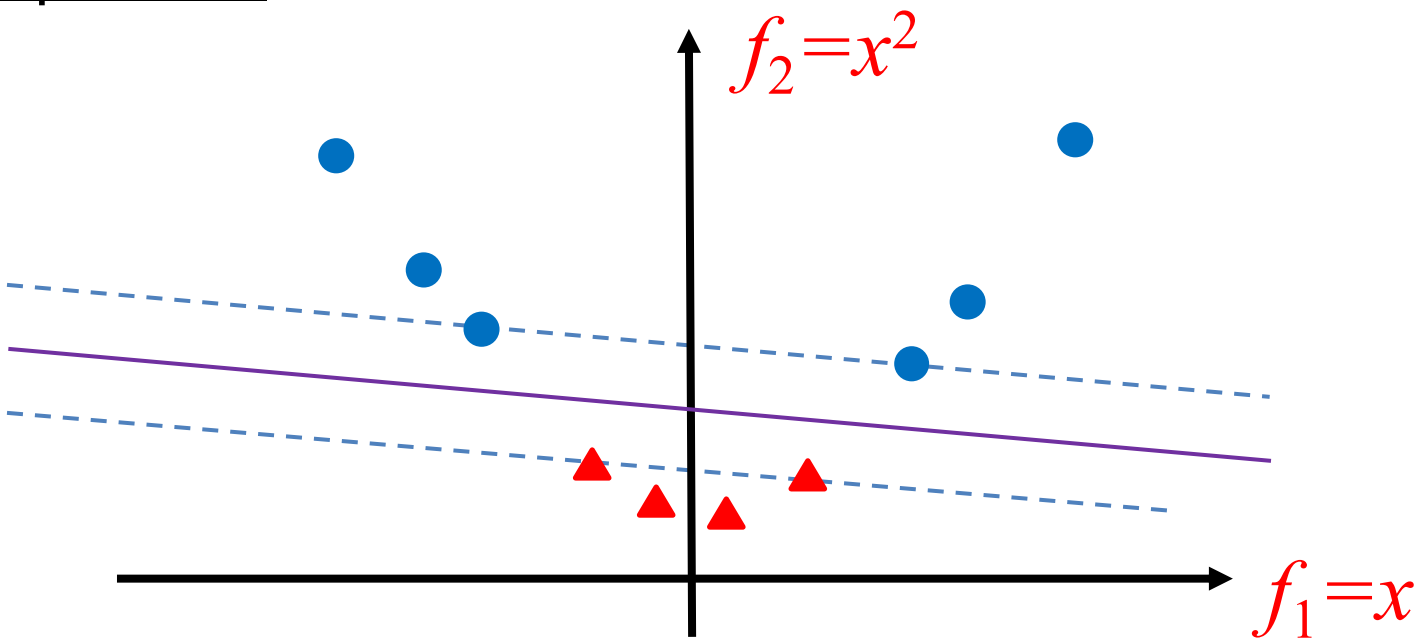
- What if we bend the space!!?
- We can **move the data into a higher-dimensional space** to make it linearly separable:



A Simple Example in 1-D

- Data points are NOT linearly separable:

- We can **move the data into a higher-dimensional space** to make it linearly separable:

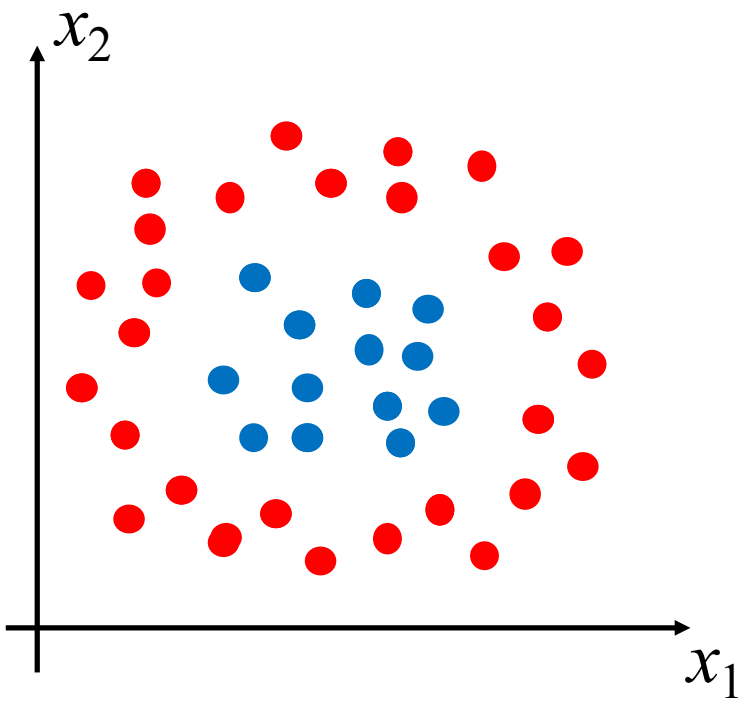


A Simple Example in 2-D

- In this example, the Data points are **NOT** linearly separable: you can **not** find a linear hyperplane (a line in 2D) to separate them:

- Can we **move the data into a higher-dimensional space** to make it linearly separable?

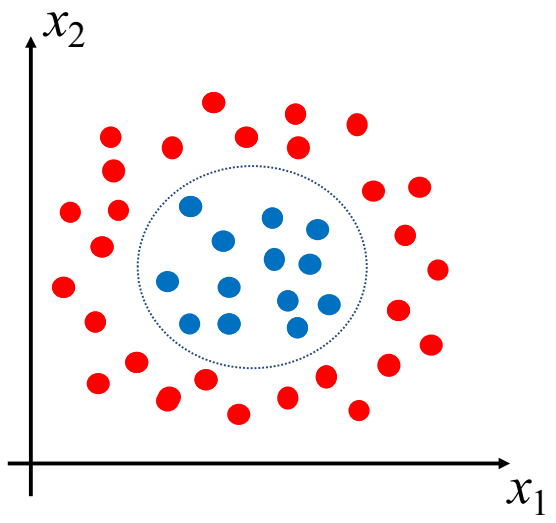
- We need a function Φ (/fai/) that moves data samples $\boldsymbol{x}$ to a higher-dimensional space:



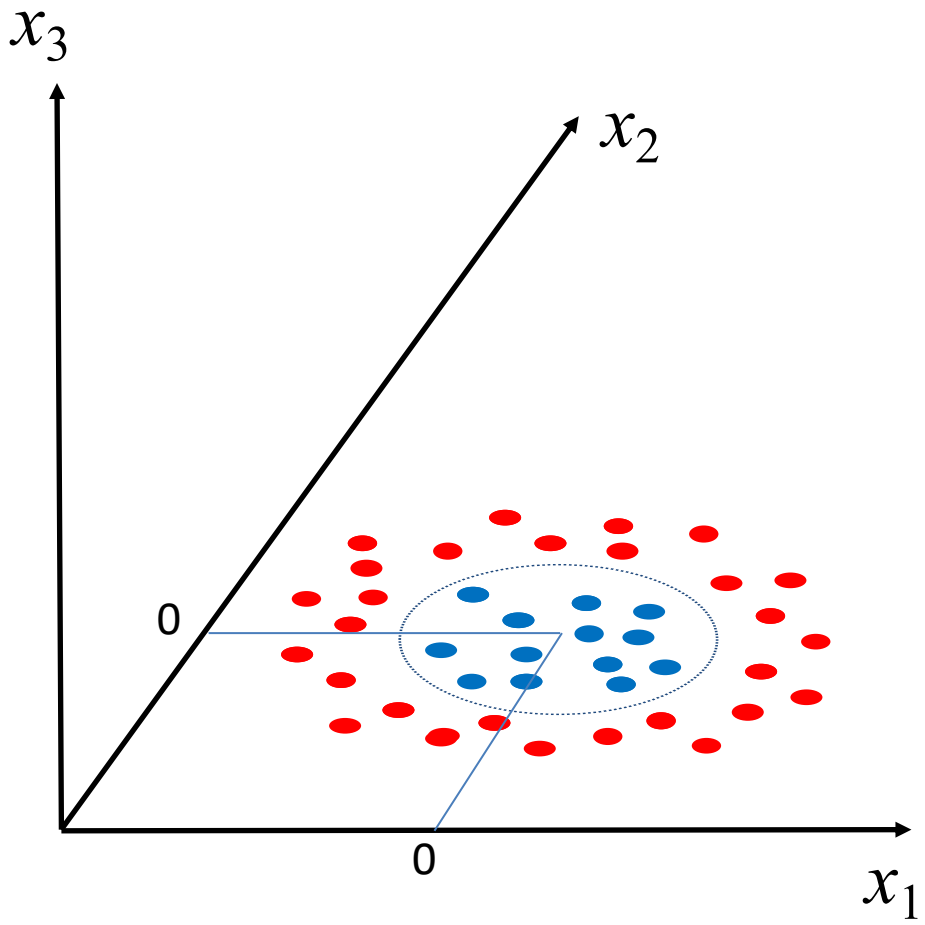
$$\Phi: \ \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$

Low-dimensional space High-dimensional space

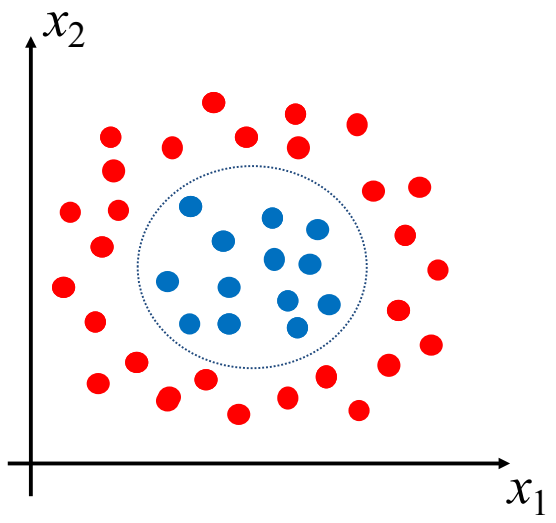
A Simple Example in 2-D



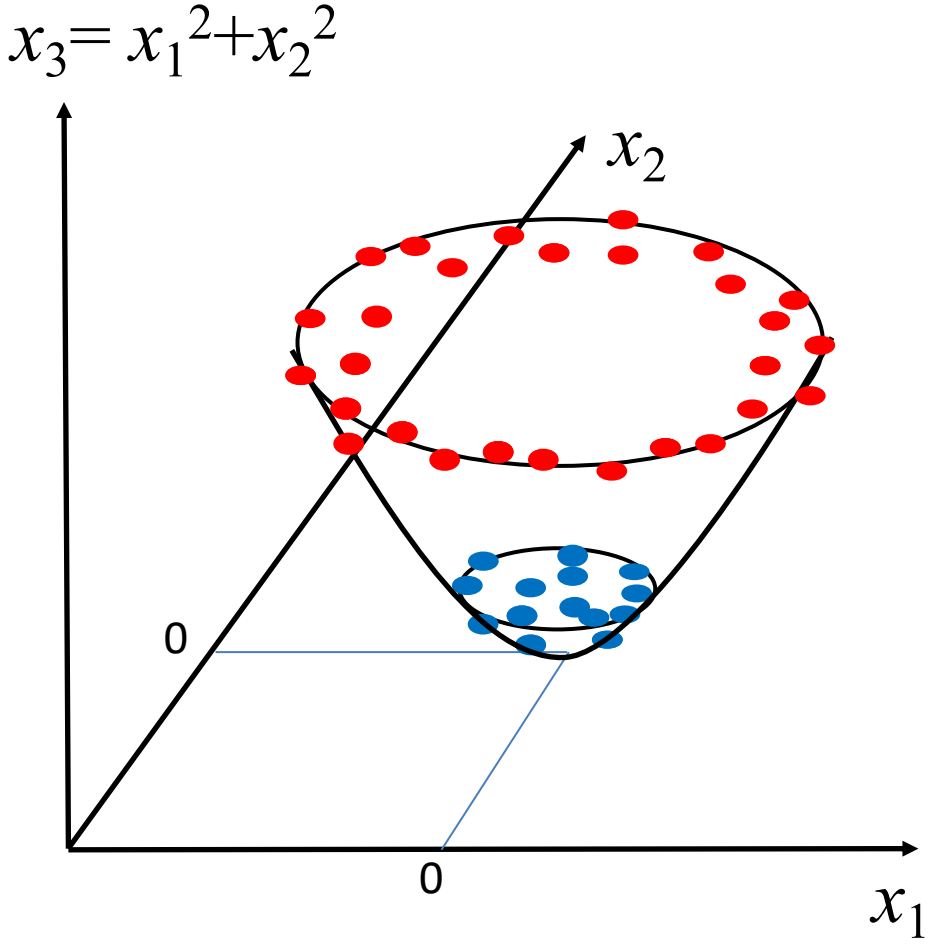
$$\Phi: \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$



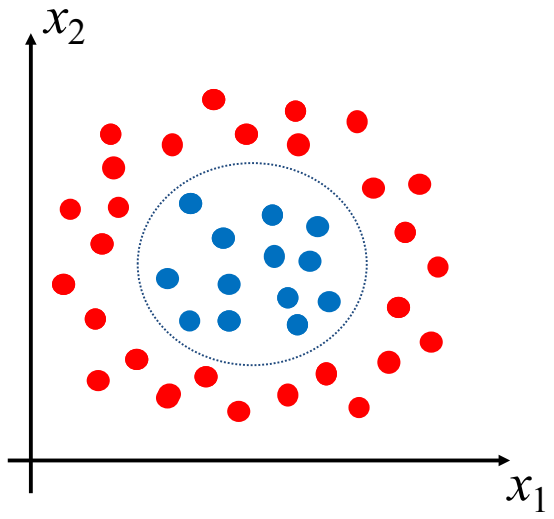
A Simple Example in 2-D



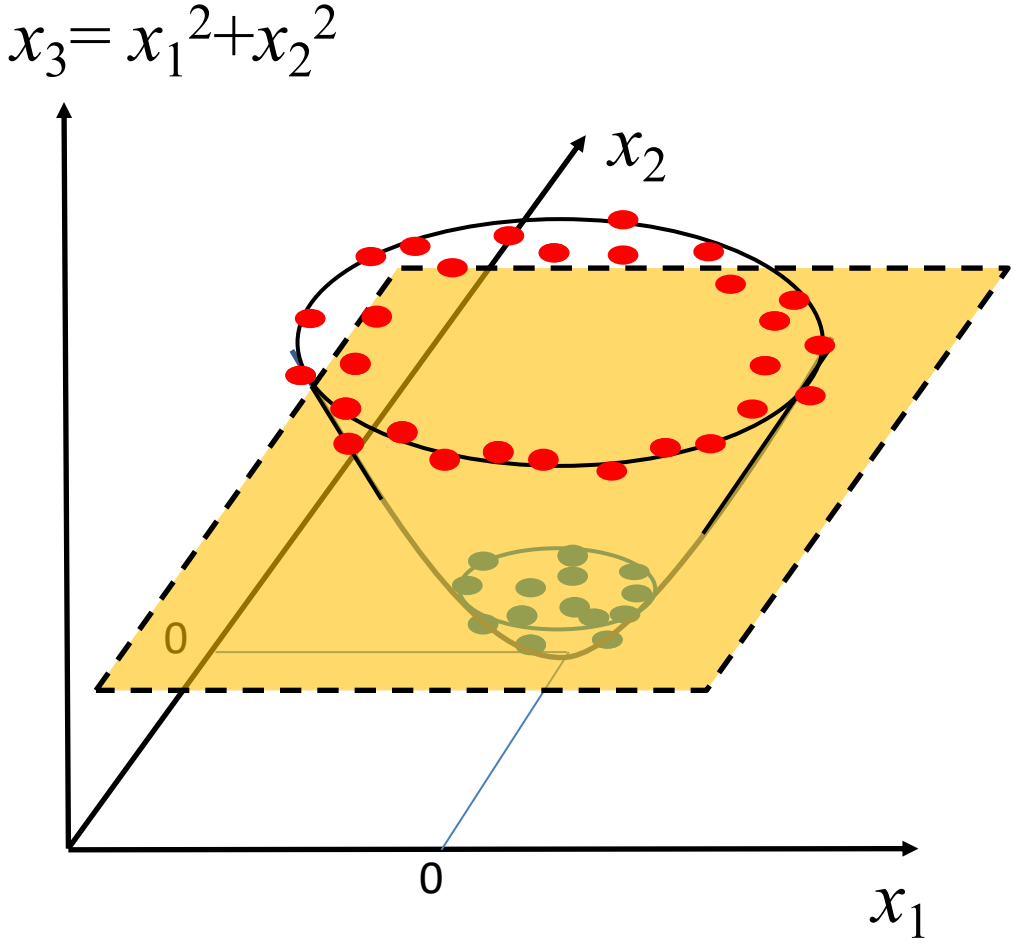
$$\Phi: \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$



A Simple Example in 2-D



$$\Phi: \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$



Nonlinear SVM

- **The main idea for non-linear SVM**: The original data can be transformed to some **higher-dimensional feature space where the data is linearly separable**.
- **Any Problem with this idea?**
  - What is the best new Space?
  - In general, we don't know the best function Φ that maps the data samples to a new space.
  - Even if we know it, the computation in the new space can be costly because it is high dimensional.
- **Solution?**
  - The **kernel Trick** comes to help!

Review: SVM Classifier

- By putting $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$, in $y = \boldsymbol{Sign}\,(\boldsymbol{w}^T \boldsymbol{x} + b)$, we will have:

$$y = \boldsymbol{Sign}\,\left(\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b \right)$$

Reminder: Inner product between vector $\boldsymbol{a}$ and $\boldsymbol{b}$: $\boldsymbol{a}^T\boldsymbol{b} = \boldsymbol{a}.\boldsymbol{b} = \sum_{i=1}^{N} a_i\, b_i$

- **Note1:** The prediction is made by computing the *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x_i}$ (where $\alpha_i \neq 0$).
- **Note2:** To Solve the optimization problem we need to calculate the inner products $\boldsymbol{x_i}^T\boldsymbol{x_j}$ between all pairs of training points.
- **THUS, everything appears in the form of inner products!**

Nonlinear SVM

- **Original Space:**

$$y = \textbf{Sign}\left(\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b\right)$$

- **Now, If we map the data into a higher-dimensional space using a function Φ , then our prediction function is:**

$$\Phi: \ \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$

Low-dimensional High-dimensional

$$y = \textbf{Sign}\left(\sum_{i=1}^{n} \alpha_i y_i \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}) + b\right)$$

Nonlinear SVM

- High-Dimensional Space (Φ: $\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$):

$$y = \textbf{\textit{Sign}}\left(\sum_{i=1}^{n} \alpha_i y_i \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}) + b\right)$$

- **Again, everything appears in the form of inner products** $\varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x})$. So, everything will become a scalar number again**!**
- **Thus, we really do NOT need to know** $\varphi(\boldsymbol{x})$**. We just need to find** $\varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x})$.
- This is called **Kernel Function**: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_j)$

Nonlinear SVM: Kernel Trick!

- By putting $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j)$, the new classification rule based on **Kernel Trick** is:

$$y = \boldsymbol{Sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$$

- **A *kernel function* is a function that corresponds to an inner product in some high-dimensional expanded feature space.**
- In other word, The kernel function plays the role of the inner product in high dimensional space.

Nonlinear SVM: Kernel Trick!

- **Summary:**

I. The main idea: The original data can be mapped to some higher-dimensional feature space, where the data is linearly separable.

II. Assume that Φ: $\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$ is the function that can map the data to some higher-dimensional space.

III. It turns out that in every calculation in both testing and training stages, every time, $\varphi(\boldsymbol{x})$ only appears in the form of inner product $\varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j)$.

IV. Thus, we really do not need to know the mapping function φ explicitly!

V. Instead, we define and use a *kernel function* as a function that corresponds to the inner product of two feature vectors in some expanded high-dimensional feature space: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j)$.

Nonlinear SVM: Kernel Trick

- Examples of most commonly-used kernel functions:
  - Polynomial kernel:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\mathrm{r} + \gamma\, \boldsymbol{x_i}^{\mathrm{T}} \boldsymbol{x_j})^{d}$$

  - Gaussian Radial-Basis Function (RBF):

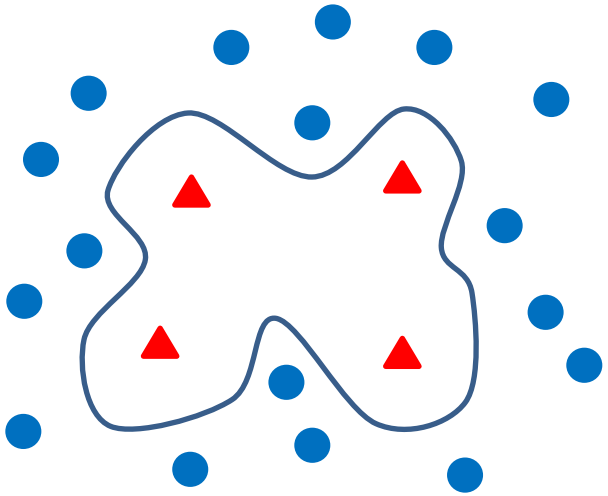
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2}{2\sigma^2}\right)$$

  - Hyperbolic Tangent:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tanh(\mathrm{r} + \gamma\, \boldsymbol{x_i}^{\mathrm{T}} \boldsymbol{x_j})$$

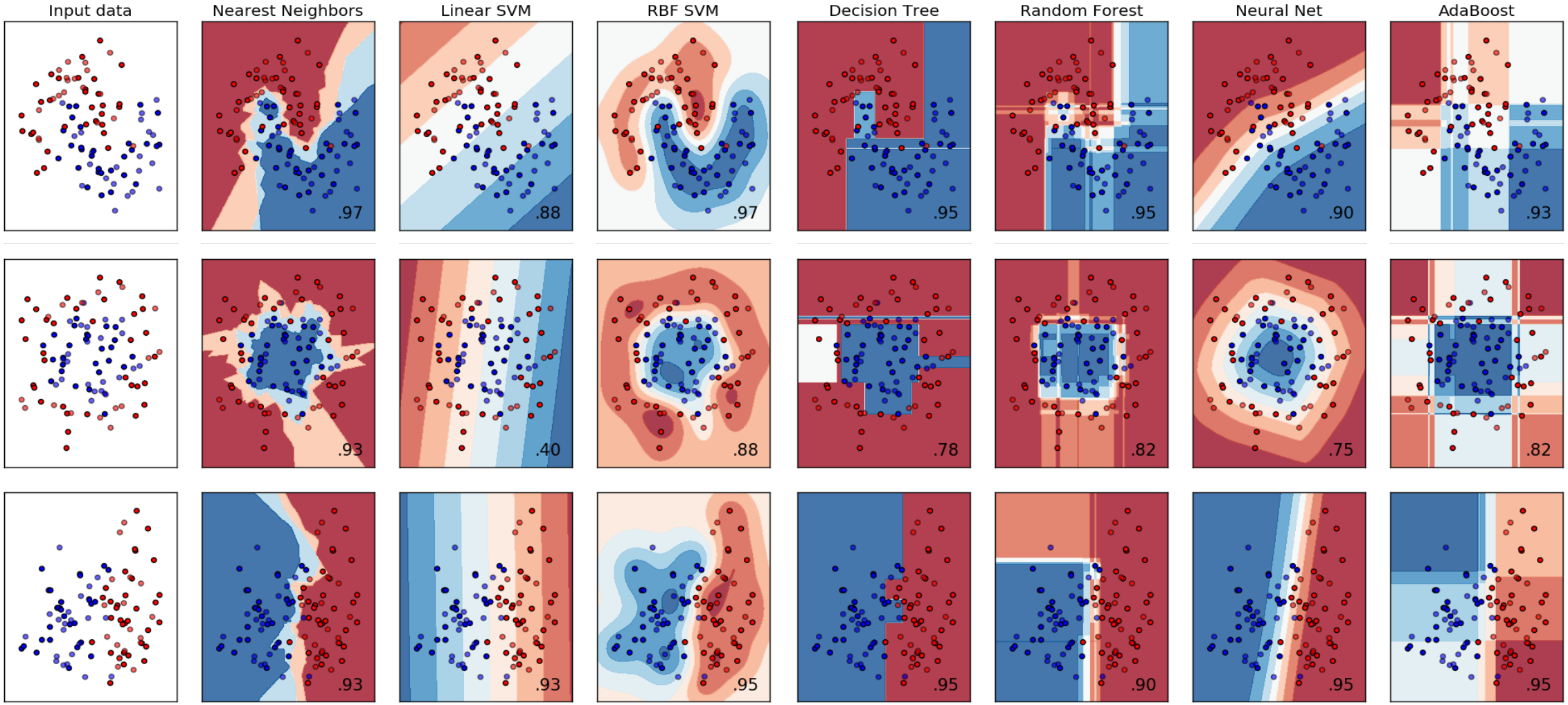
Another Interpretation (Optional)

- Kernel function represents some kind of **similarity** (or closeness) between two vectors in the space. That is why sometimes it is called ***Similarity Function***.
- Thus, the Kernel between two vectors in training and testing stages, can measure the closeness of a testing vector to the training vectors.
- This closeness, somehow defines a **non-linear** hidden border that can be our decision boundary.



- Now, after CS4661 and CS4662, you know how the **most popular** and the **most effective** data analytics and machine learning algorithms work! More importantly, you know how to use them! ☺

Classifier Comparison



Ref: Sklearn Documentation

Project Progress Report

- **Due Date for Project Progress Report: Sunday, March 28.**
- Please submit your progress report on Canvas by March 28.
- **Your report should include:**
  - Team members information and their responsibilities.
  - Project description/details.
  - Your results, methods, progress so far.
- Note: One submission per group (the team leader submits on behalf of everyone).

Thank You!

Questions?