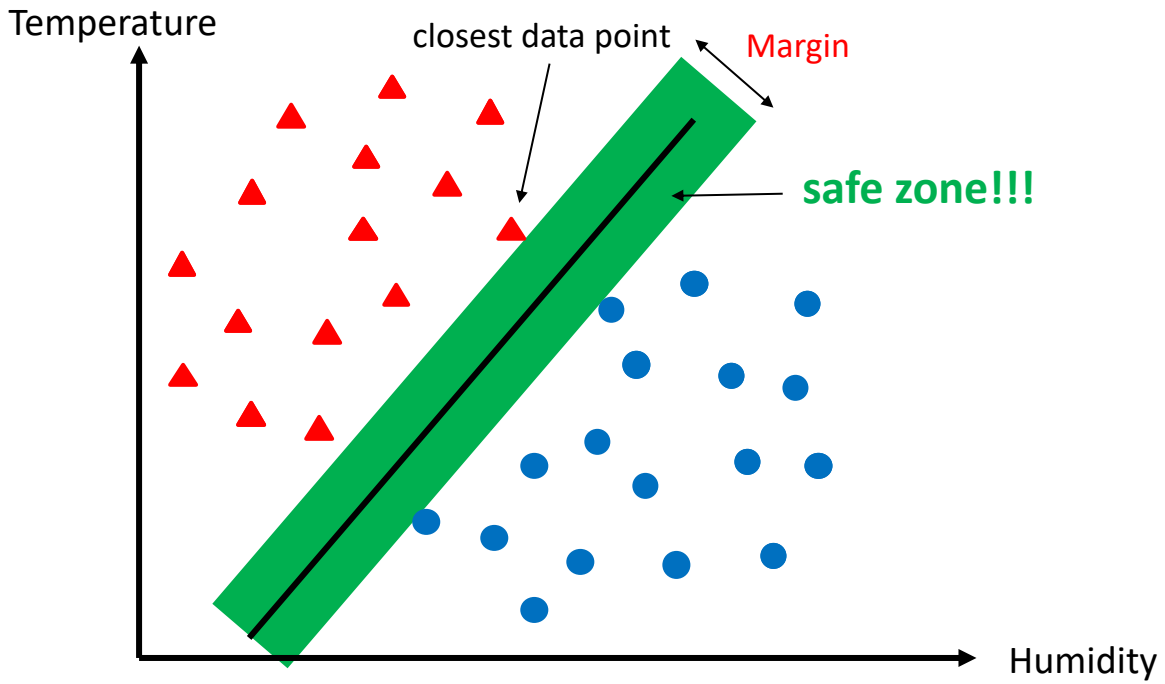
Advanced Machine Learning and Deep Learning

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

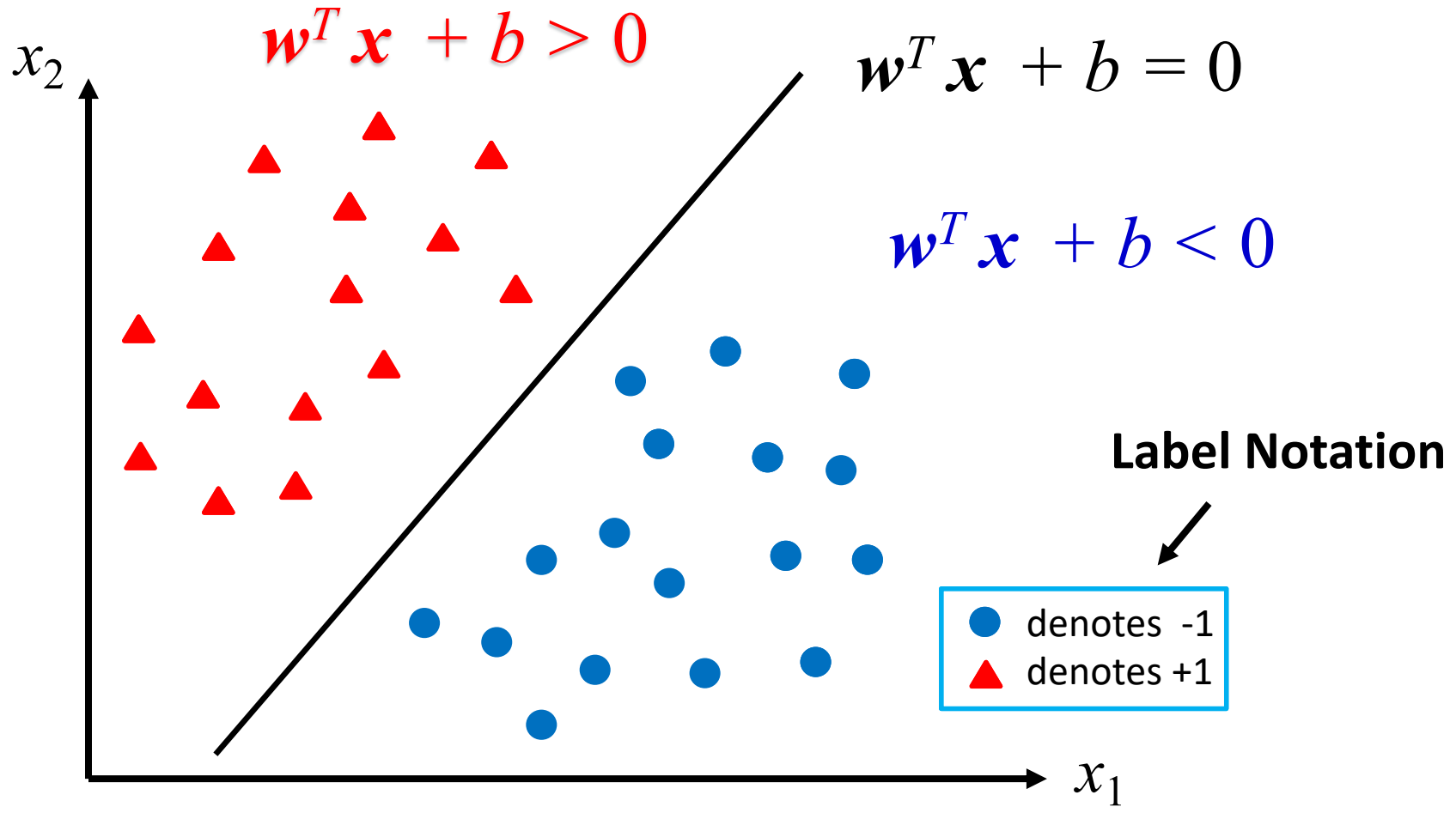
Support Vector Machine (SVM)

Review

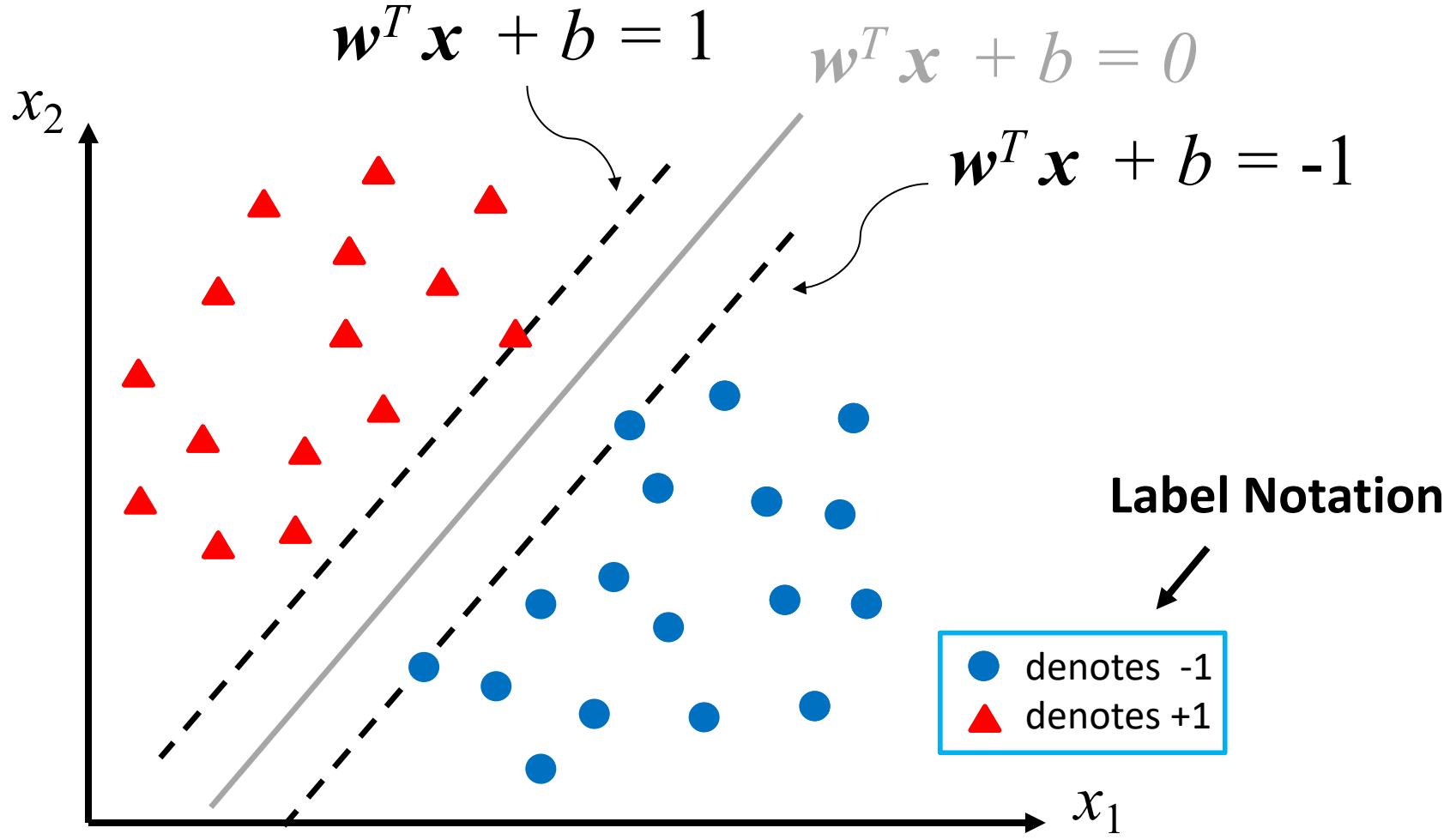
- The best classifier is the one with the **Maximum Margin**.
- **Margin** is widest boundary area before hitting a data point.



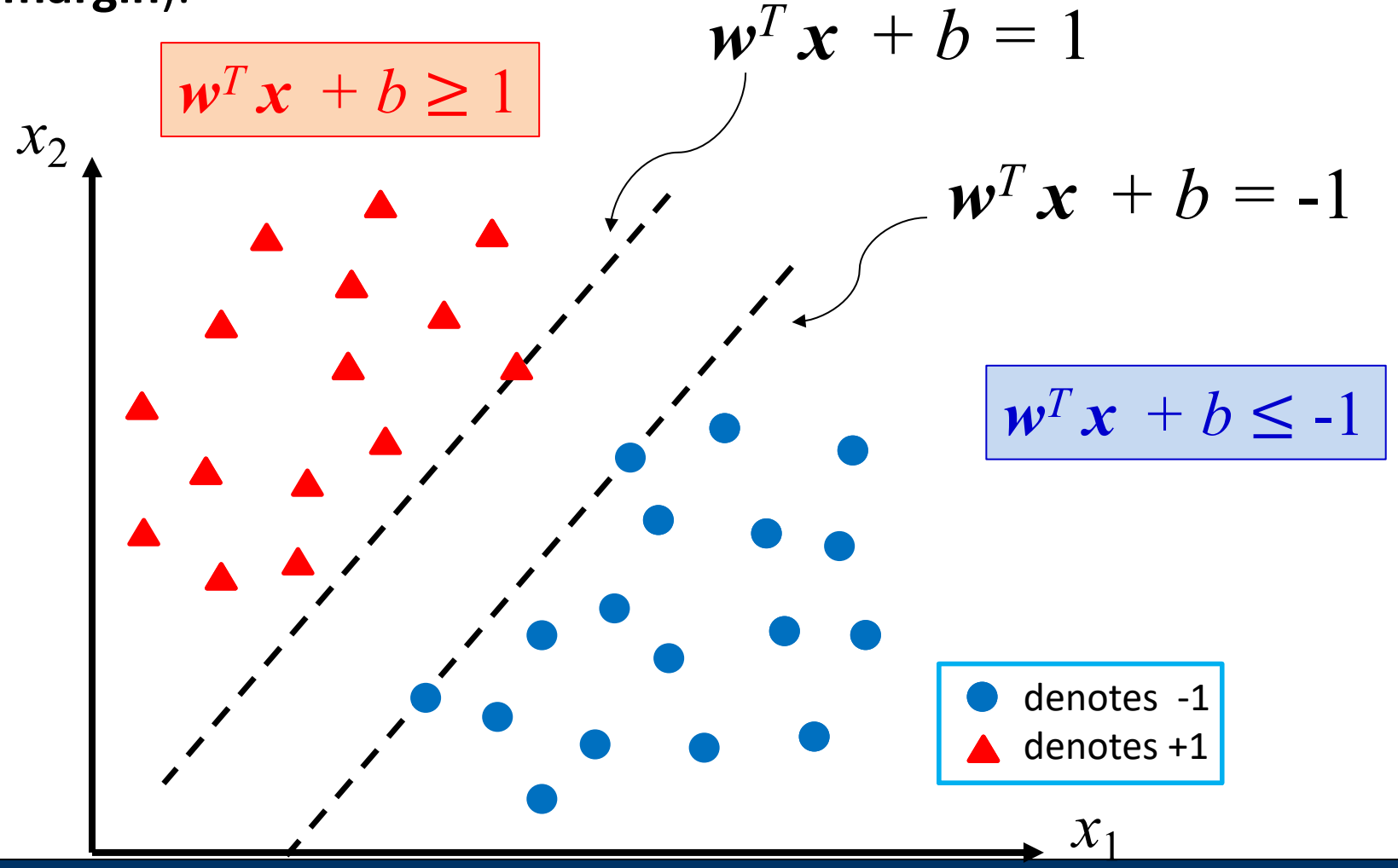
- Given a set of training data points, each point will be considered as a ***m-dimensional vector*** in space. We are looking for ***a line whose value is positive for red samples, and negative for blue samples.***

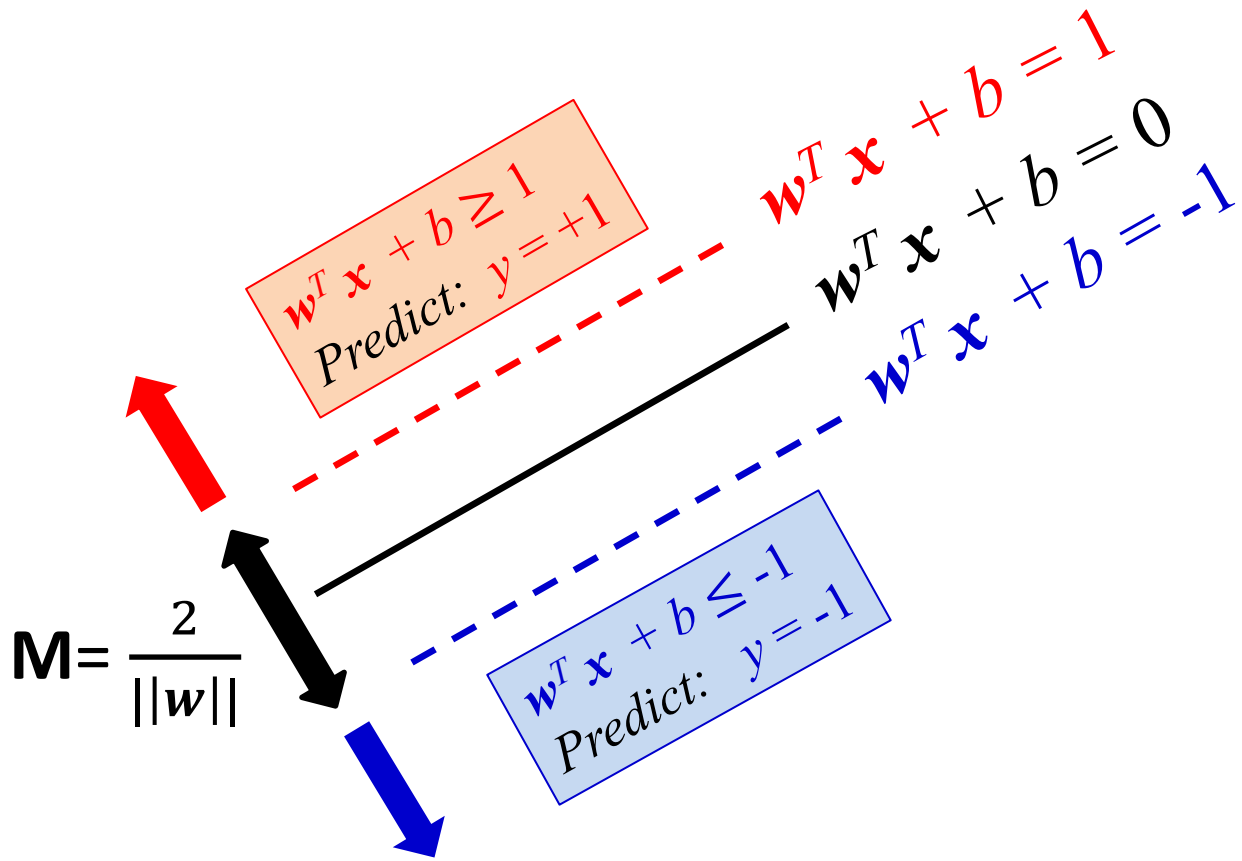


Let's do even better: We can find **two parallel lines** (rather than just one line) that separate the two classes of data! Red samples are ABOVE the TOP line, and blue samples are BELOW the BOTTOM line. These two hyperplanes can be described by the following equations:



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**).



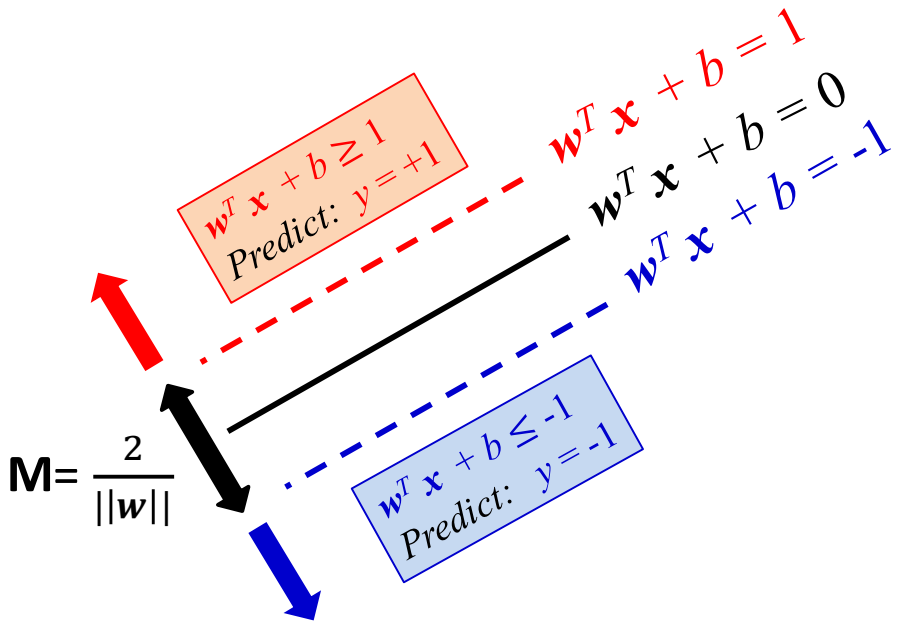


- Given a Training set, the goal is to find the best classifier that classifies the samples correctly and has the maximum margin (maximum M = $\frac{2}{||w||}$).

SVM Classifier

- ***Formulation:***

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



The Optimization Problem

- Thus, given a training set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$, the new formulation is:

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

The Optimization Problem (Optional)

Quadratic programming with linear constraints

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

This is a "***quadratic programming with linear constraints***". It can be solved using Lagrange Multipliers:

$$\min L_p(\boldsymbol{w}, b, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \right), \alpha_i \geq 0$$

The Optimization Problem (Optional)

This is a "***quadratic programming with linear constraints***". It can be solved using Lagrange Multipliers:

$$\min L_p(\boldsymbol{w}, b, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \right), \alpha_i \geq 0$$

$$\begin{cases} \dfrac{\partial L_p}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \\ \dfrac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

The Optimization Problem (Optional)

- We solve it by constructing a ***Lagrange dual problem***:

Find $\alpha_1 \ldots \alpha_N$:

$$\begin{cases} \text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \qquad \alpha_i \geq 0 \text{ , and } \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

The Optimization Problem (Optional)

- By solving this problem, we will find the best "$\boldsymbol{w}$" and "b" corresponding to the maximum margin:

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

$$b = y_k - \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}_{\mathbf{k}} \text{ for any } \boldsymbol{x}_{\mathbf{k}} \text{ such that } \alpha_k \neq 0$$

- It turns out that only Support Vectors have $\alpha_i \neq 0$
- Support Vectors are the samples which lie nearest to classifier (***on the border***)

SVM Classifier

- By solving this problem, we will find the best $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ and b corresponding to the maximum margin.
- Then, the ***middle line*** "$(\boldsymbol{w}^T \boldsymbol{x} + b) = 0$" will be our best SVM classifier.
- **SVM Classifier**:

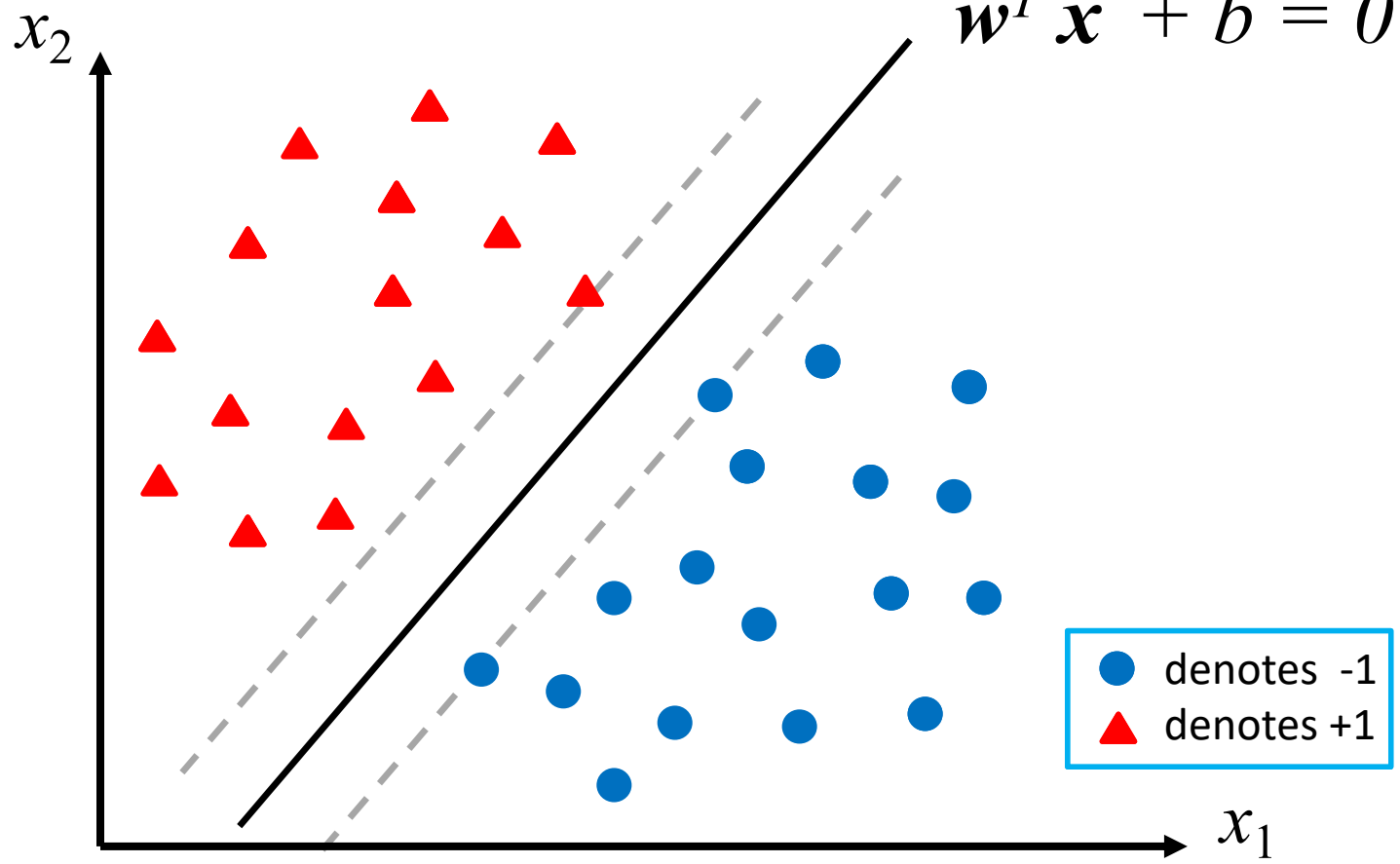
$$If \quad \boldsymbol{w}^T \boldsymbol{x} + b \geq 0 \rightarrow Predict: \quad y = +1$$
$$If \quad \boldsymbol{w}^T \boldsymbol{x} + b < 0 \rightarrow Predict: \quad y = -1$$

Or Equivalently:

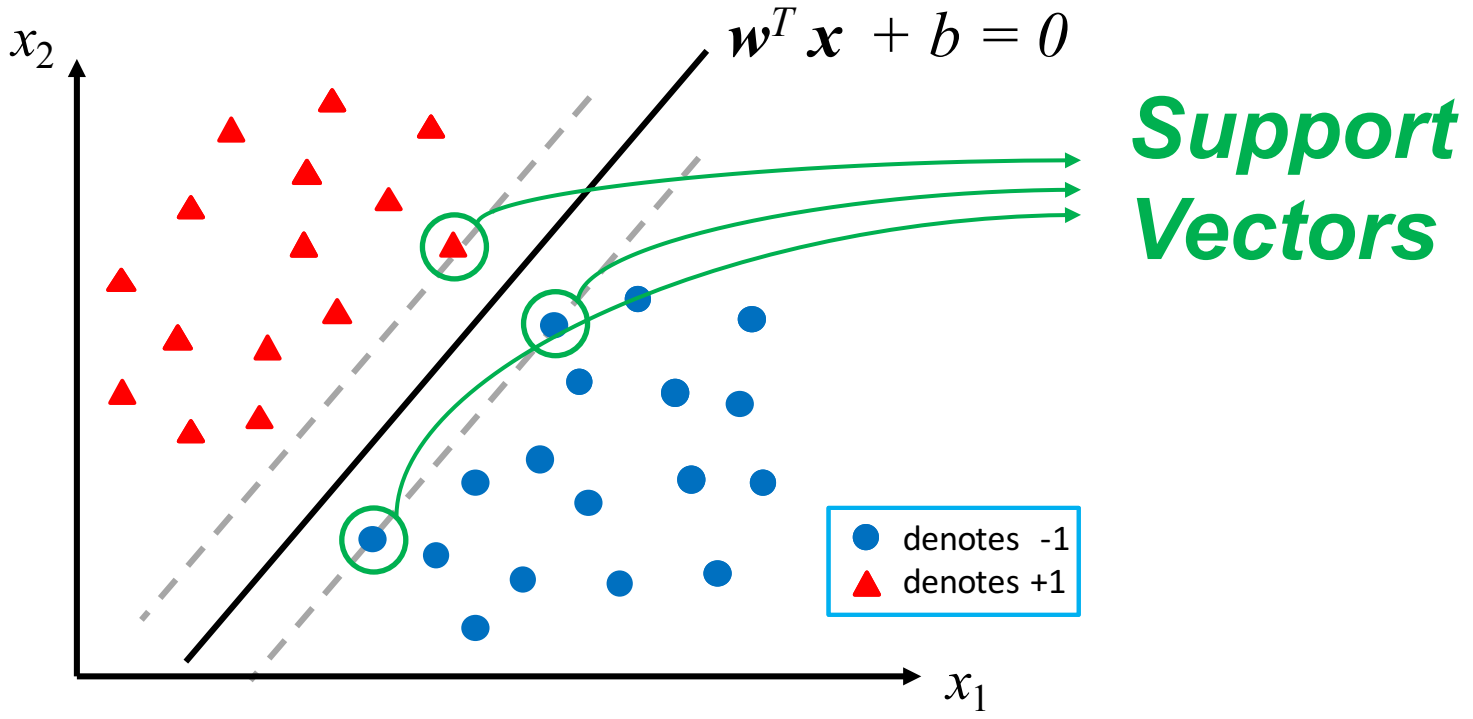
$$y = \boldsymbol{Sign}\ (\boldsymbol{w}^T \boldsymbol{x} + b) \qquad , \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

SVM Classifier:

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0$$



Important Note: The max-margin hyperplane is completely determined by those samples $\boldsymbol{x_i}$ which lie nearest to it (***on the border***). These $\boldsymbol{x_i}$'s are called ***support vectors.***



SVM Classifier

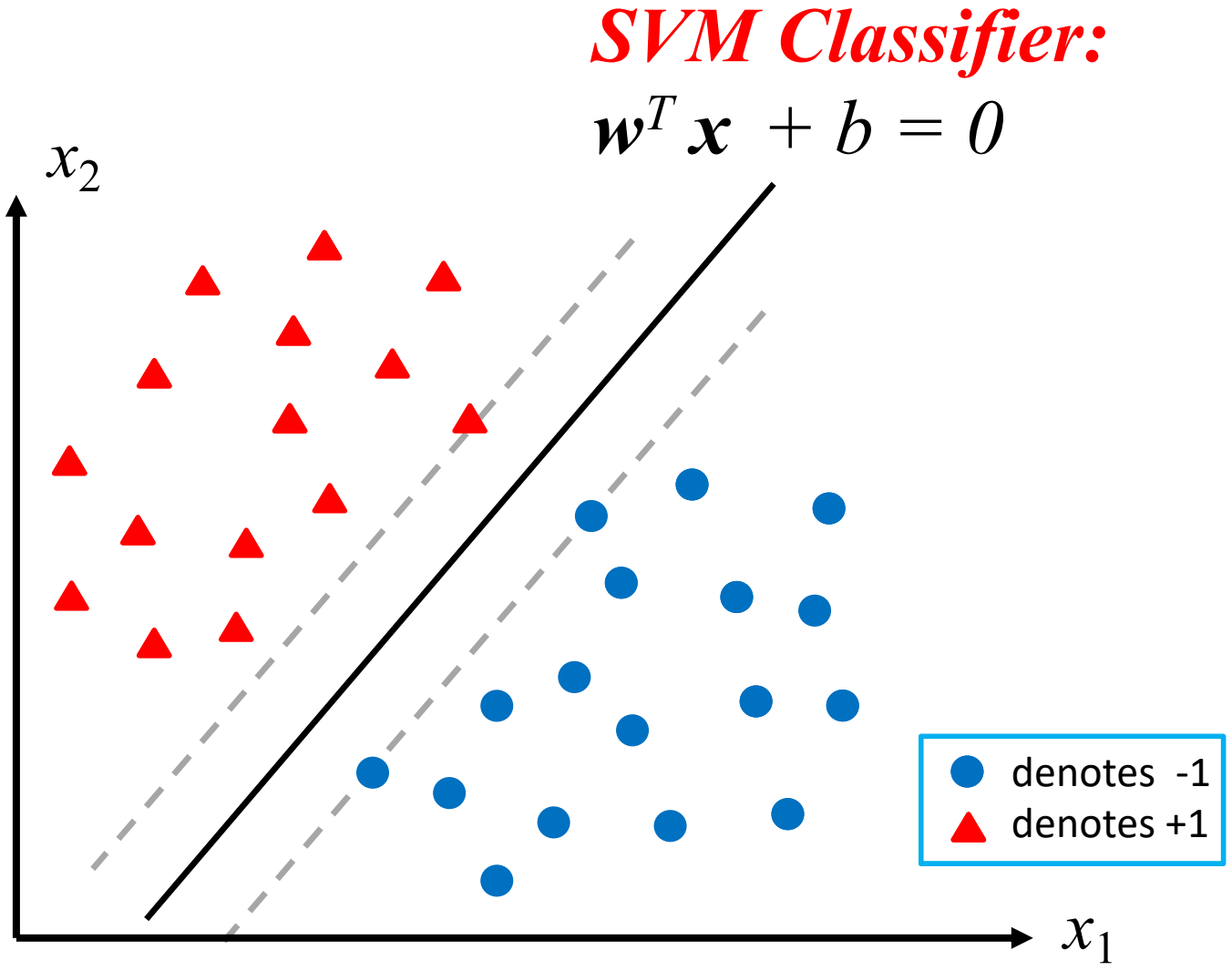
- By replacing $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ in $y = \boldsymbol{Sign}\ (\boldsymbol{w}^T \boldsymbol{x} + b)$, we will have:
- **SVM Classifier**: $y = \boldsymbol{Sign}\ (\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b\ )$

- **Note1 (Prediction):** The prediction is made by computing the *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x_i}$ (because only for the support vectors $\alpha_i \neq 0$).
- **Note2 (Training):** To Solve the optimization problem (find the best line), we need to calculate the inner products $\boldsymbol{x_i^T x_j}$ between all pairs of training points.

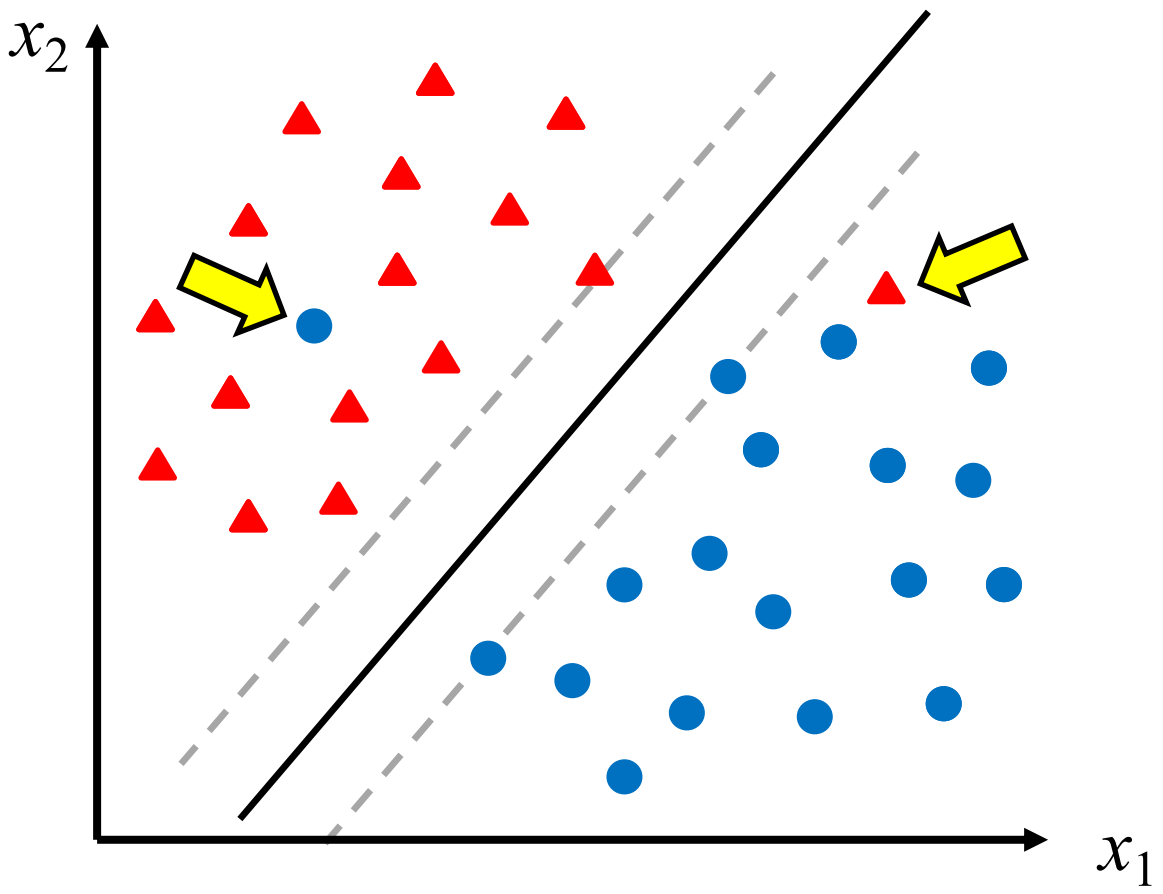
Reminder: Inner product between vector $\boldsymbol{a}$ and $\boldsymbol{b}$: $\boldsymbol{a^T b} = \boldsymbol{a.b} = \sum_{i=1}^{N} a_i\, b_i$

SVM Classifier

- Everything looks good so far!
- But, there is still a problem that should be addressed!
- So far, for sake of simplicity, we have assumed that all samples are separable!
- This approach called Hard Margin.
- How about Non-Separable samples???

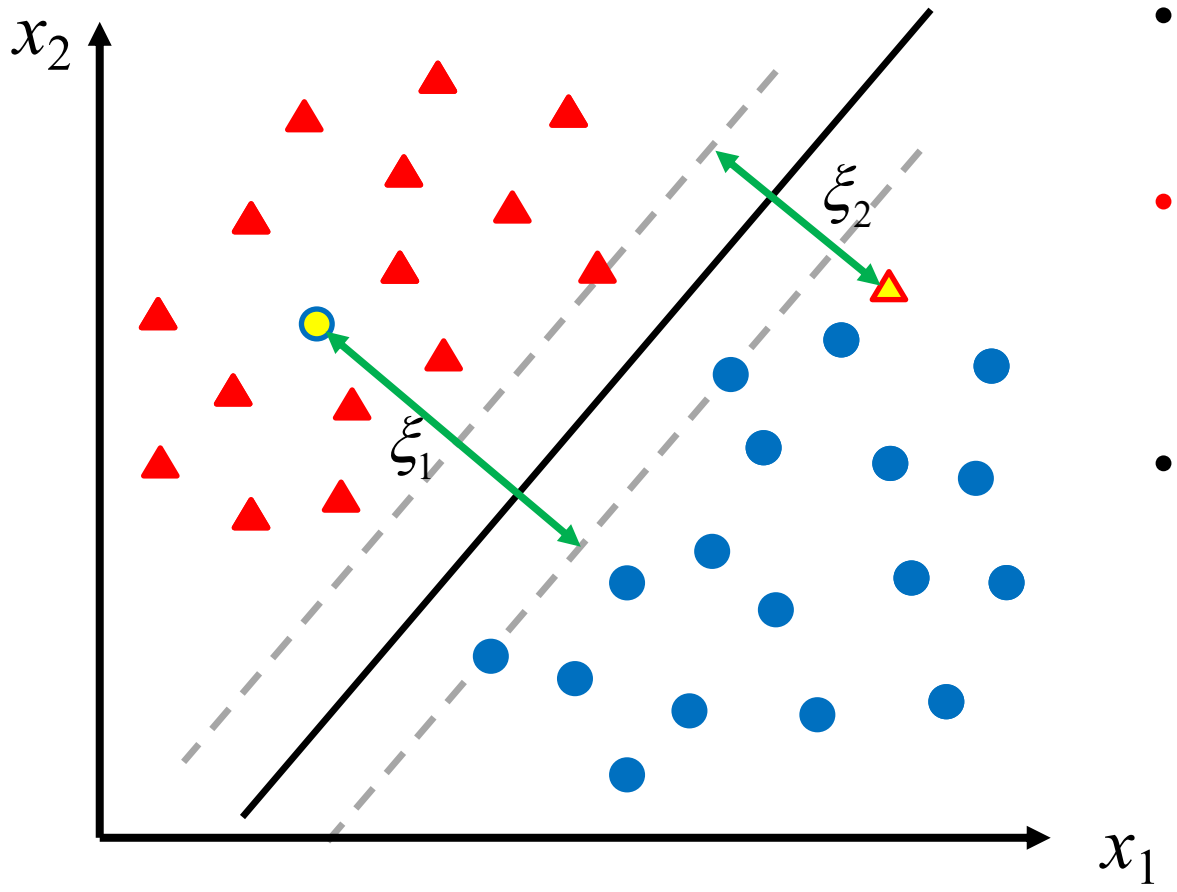


Non-Separable Samples



- So far, we assumed that all data points are separable, and will be classified correctly (no training error)!
- How about the samples that are not separable?

Non-Separable Samples



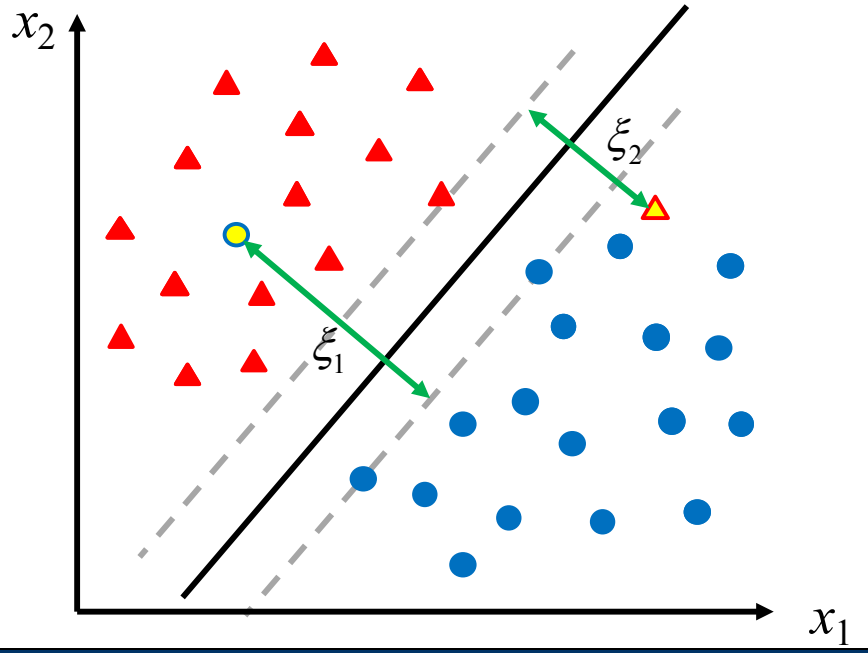
- How about the samples that are not separable?
- **Solution:** Don't ruin your model becuase of some bad data samples.
- Instead, we define **Slack Variables** ξ_i to allow mis-classification of a few difficult or bad/noisy data points. (ξ_i is pronounced /zai/)

Non-Separable Samples: Soft Margin

- **Solution:** Don't ruin your model becuase of some bad data samples.
- Instead, we define Slack Variables ξ_i to allow mis-classification of a few difficult or bad/noisy data points.
- In fact, we cheat a little by relaxing our criteria, and allow **some of the data samples** to be on the **wrong side** of the line.

- This way, we still keep our previous model and just **let it tolerate some errors**.
- However, remember that we still need to **minimize** this amount of error. So, we add some penalty on Slack Variables!

- **This is called Soft Margin.**

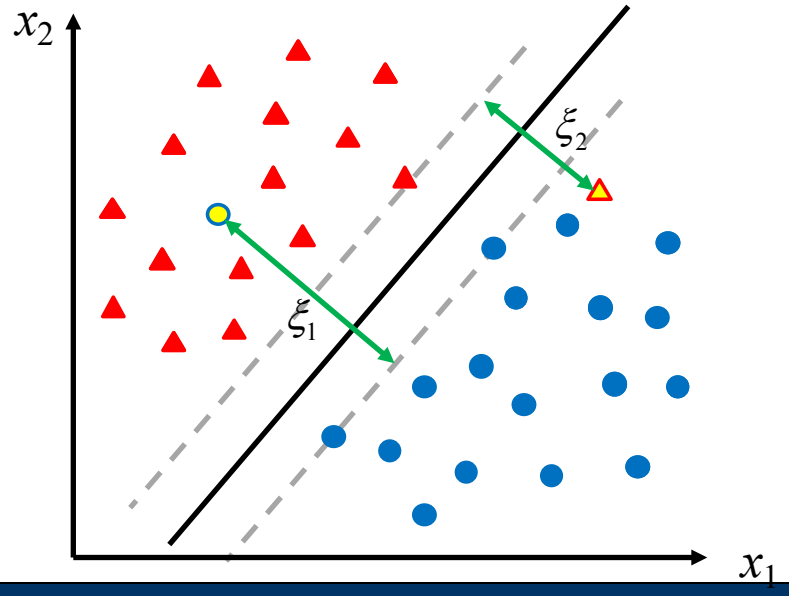


- **Hard Margin (our previous formula):**

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

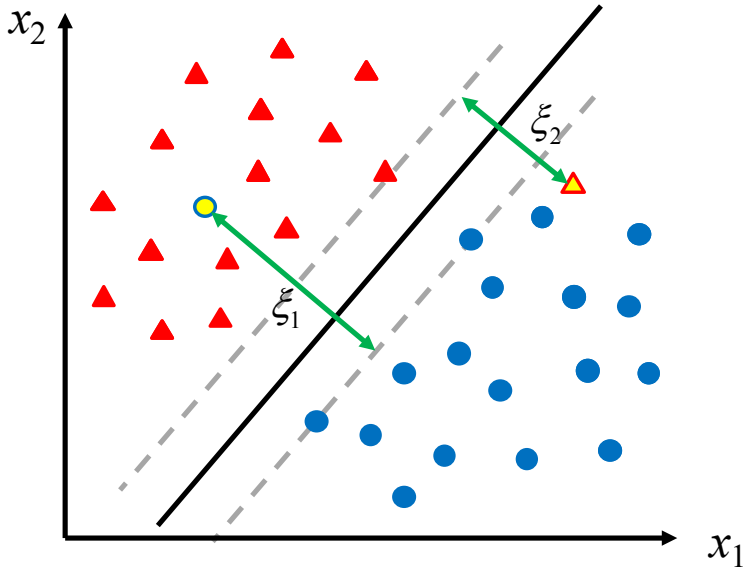
- **Soft Margin:**

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



Soft Margin:

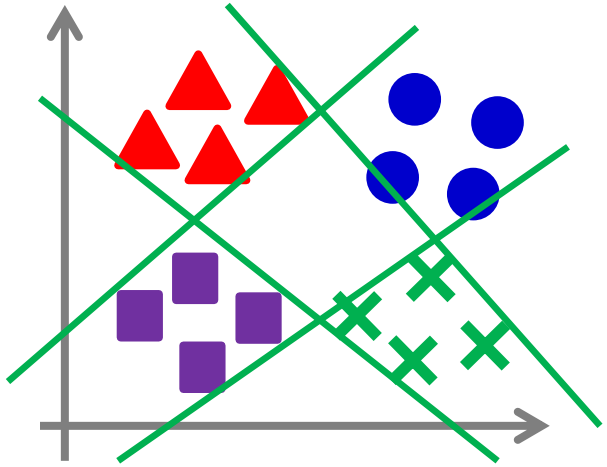
$$\begin{cases} \text{Minimize } \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum \xi_i \right) \\ \text{Subject to: } y_i\,(\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1 - \boldsymbol{\xi_i} \\ \text{for } i = 1, 2, \ldots, n \end{cases}$$



- "**C**" is a very important parameter to adjust SVM results and control Overfitting:
  - **Large C:** Penalize ξ ➔ No tolerance for misclassification ➔ Low bias, high variance ➔ prone to overfitting.
  - **Small C:** Let larger ξ ➔ allow misclassification ➔ High bias, low variance ➔ prone to underfitting.

Multi-Class Classification

- SVM is originally a binary classifier (2 labels). However, it is possible to perform multiclass classification using ***One-vs-All method***.



- **One-vs-All method**: In this approach, assuming that we have K different labels, we train K SVM-classifiers, one to distinguish class k from the rest (for $k=1,2,..,K$), and learn the weights $\boldsymbol{w}^{(k)}$ for each classifier ($\boldsymbol{w}^{(1)}$ for classifying class1 vs. the rest, $\boldsymbol{w}^{(2)}$ for classifying class2 vs. the rest, …). Then, in testing stage, for a new sample $\boldsymbol{x}$, pick the class with the largest $(\boldsymbol{w}^T \boldsymbol{x}_i + b)$.

SVM in Python

- Linear SVM:

```
from sklearn.svm import LinearSVC
my_SVM = LinearSVC(parameters)
my_SVM.fit (X_train, y_train)
y_predict = my_SVM.predict (X_test)
```

- Non-Linear SVM (will cover it next):

```
from sklearn.svm import SVC
my_SVM = SVC(parameters)
my_SVM.fit (X_train, y_train)
y_predict = my_SVM.predict (X_test)
```

Thank You!

Questions?