# Advanced Machine Learning and Deep Learning

**Dr. Mohammad Pourhomayoun**

Assistant Professor

Computer Science Department

California State University, Los Angeles

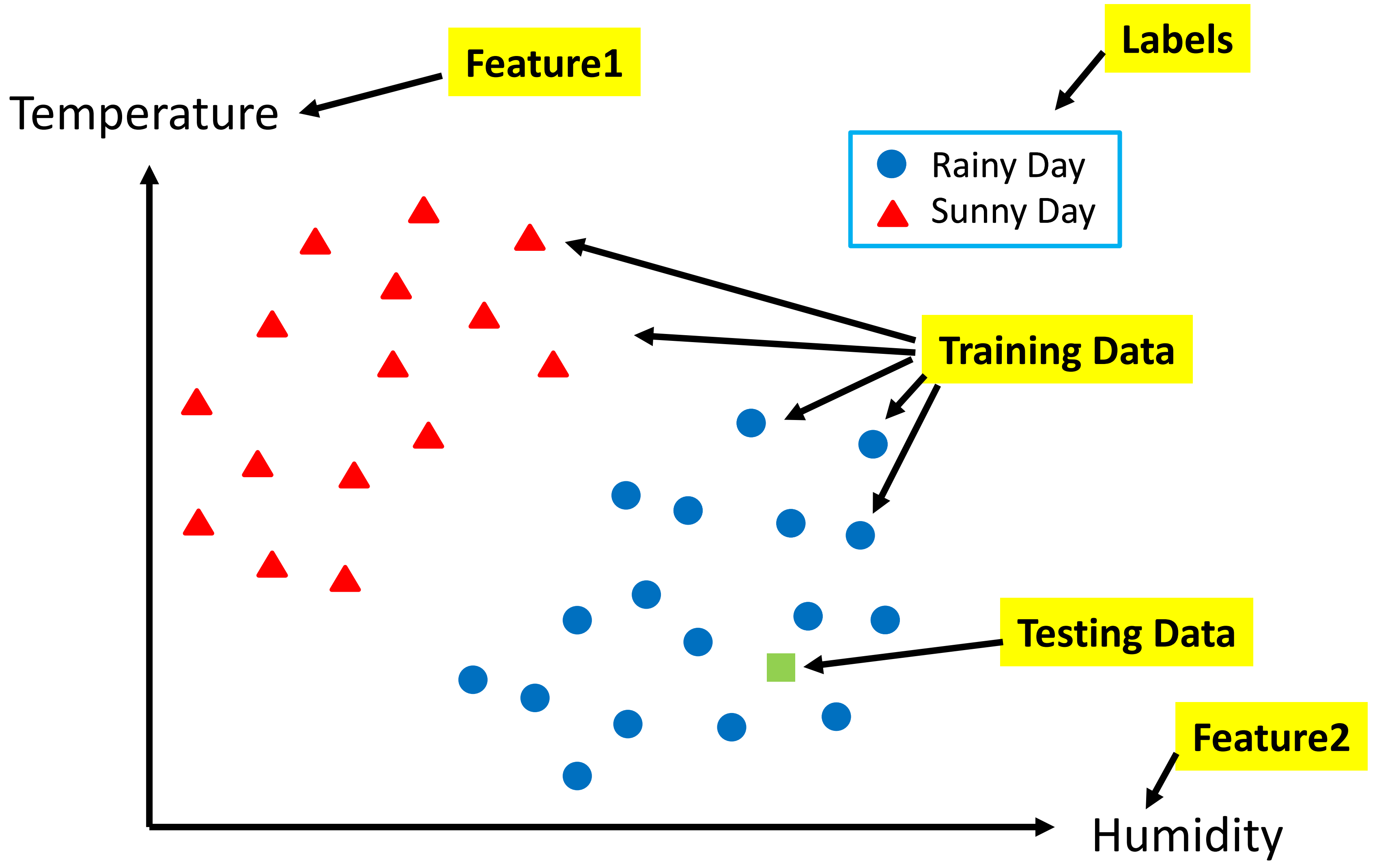
## Support Vector Machine (SVM)

# Support Vector Machine (SVM)

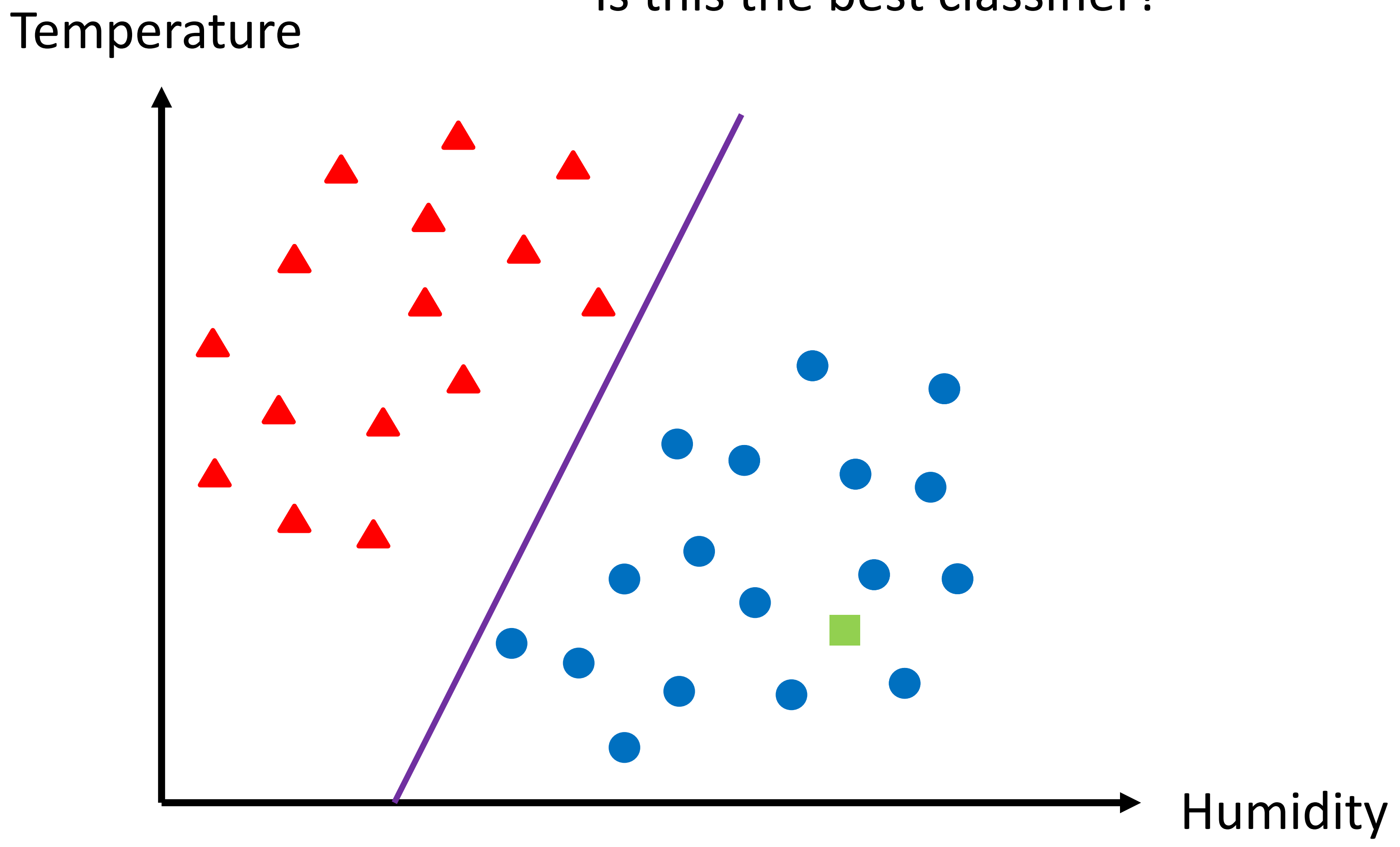
- **Support Vector Machine (SVM)** is a ***very popular*** machine learning algorithm originally invented by Vladimir N. Vapnik.
- It is ***one of the most accurate*** supervised learning method. It can even beat ANN in some cases, especially when the dataset is small with small dimensionality.
- It was originally a linear classifier. But, Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik later suggested a way to convert SVM into a non-linear model using ***Kernel Trick*** *(we will cover it later).*

# Support Vector Machine (SVM)

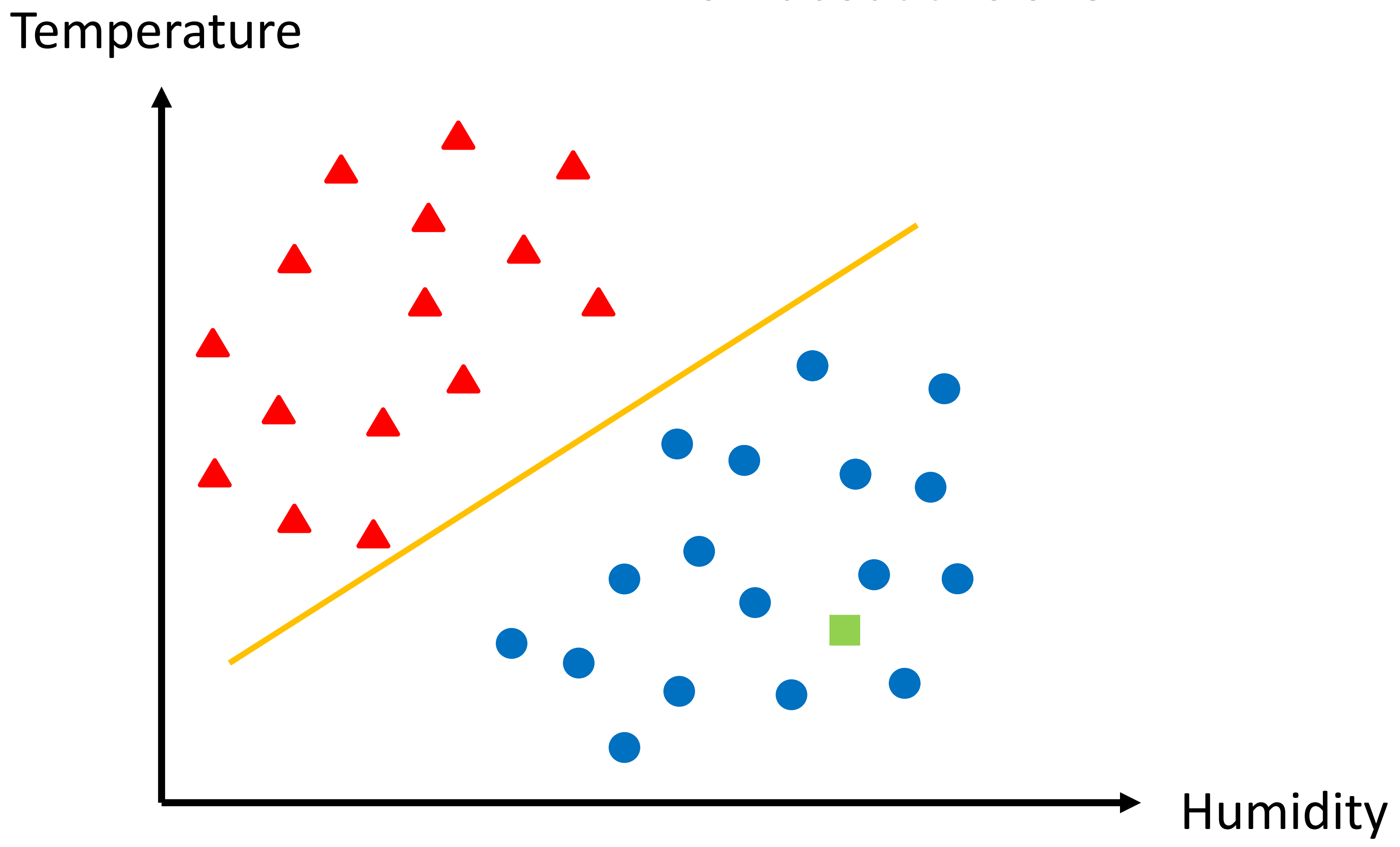
- SVM became famous when it achieved an accuracy comparable to artificial neural network using ***hand-designed features*** in a handwriting recognition.
- SVMs is being used for a wide range of real world applications.  SVMs are helpful in text mining, image classification, pattern recognition, hand-writing recognition, etc.
- The SVM algorithm has been widely applied in science: Biology, Medicine, Chemistry, Bioinformatics, Astronomy, Business, Finance, …
- **Generally speaking, SVM tries to find the best *hyperplane* (example: a line in 2–D with 2 features, or a plane in 3-D with 3 features, …), or set of hyperplanes in feature space, which can be *used for separating (classifying) data samples*.**


Feature1
Temperature
Labels
Rainy Day
Sunny Day
Training Data
Testing Data
Feature2
Humidity

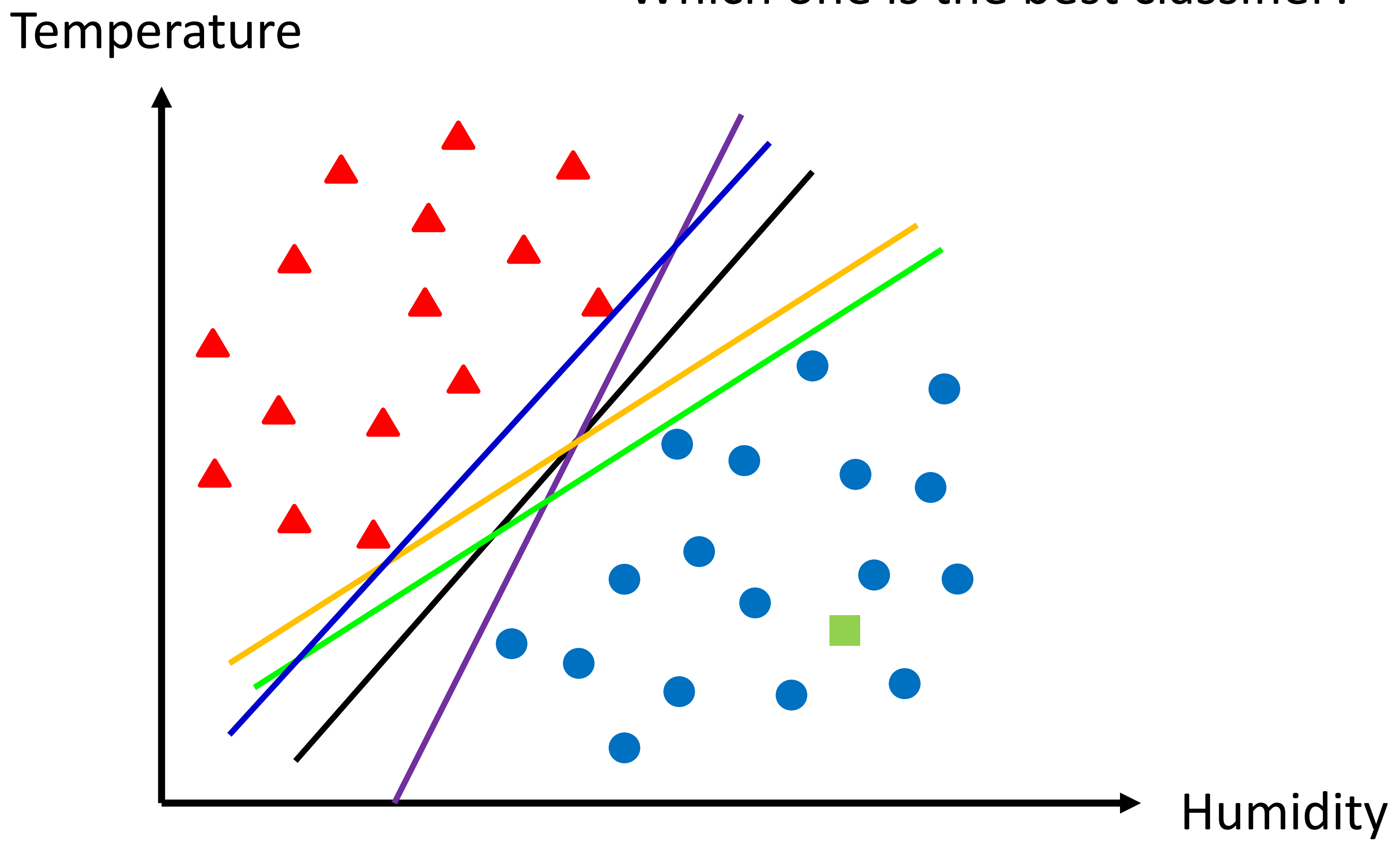
- How can we classify these samples?
- Is this the best classifier?



- How about this one?

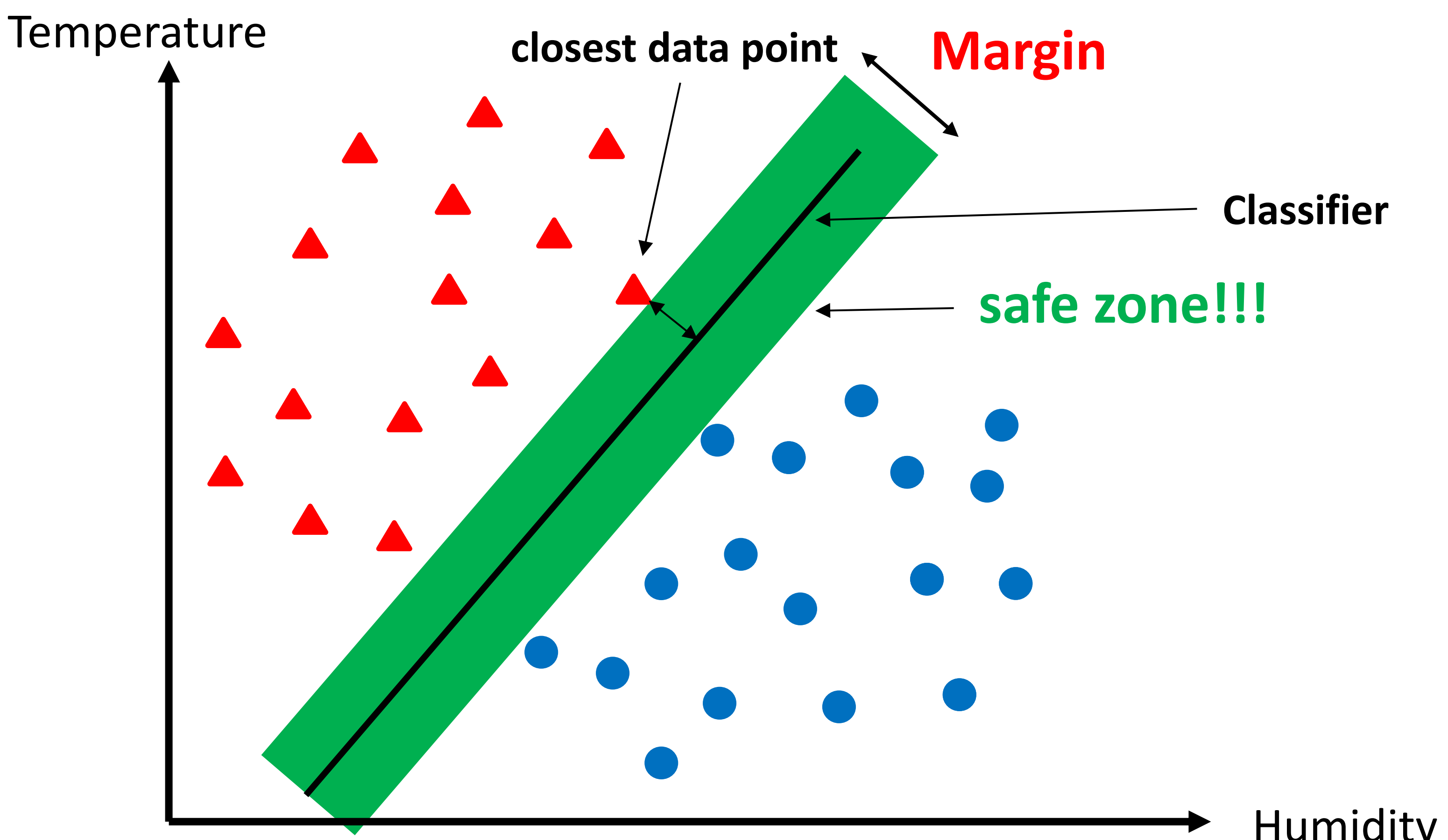


- Which one is the best classifier?



# Support Vector Machine (SVM)

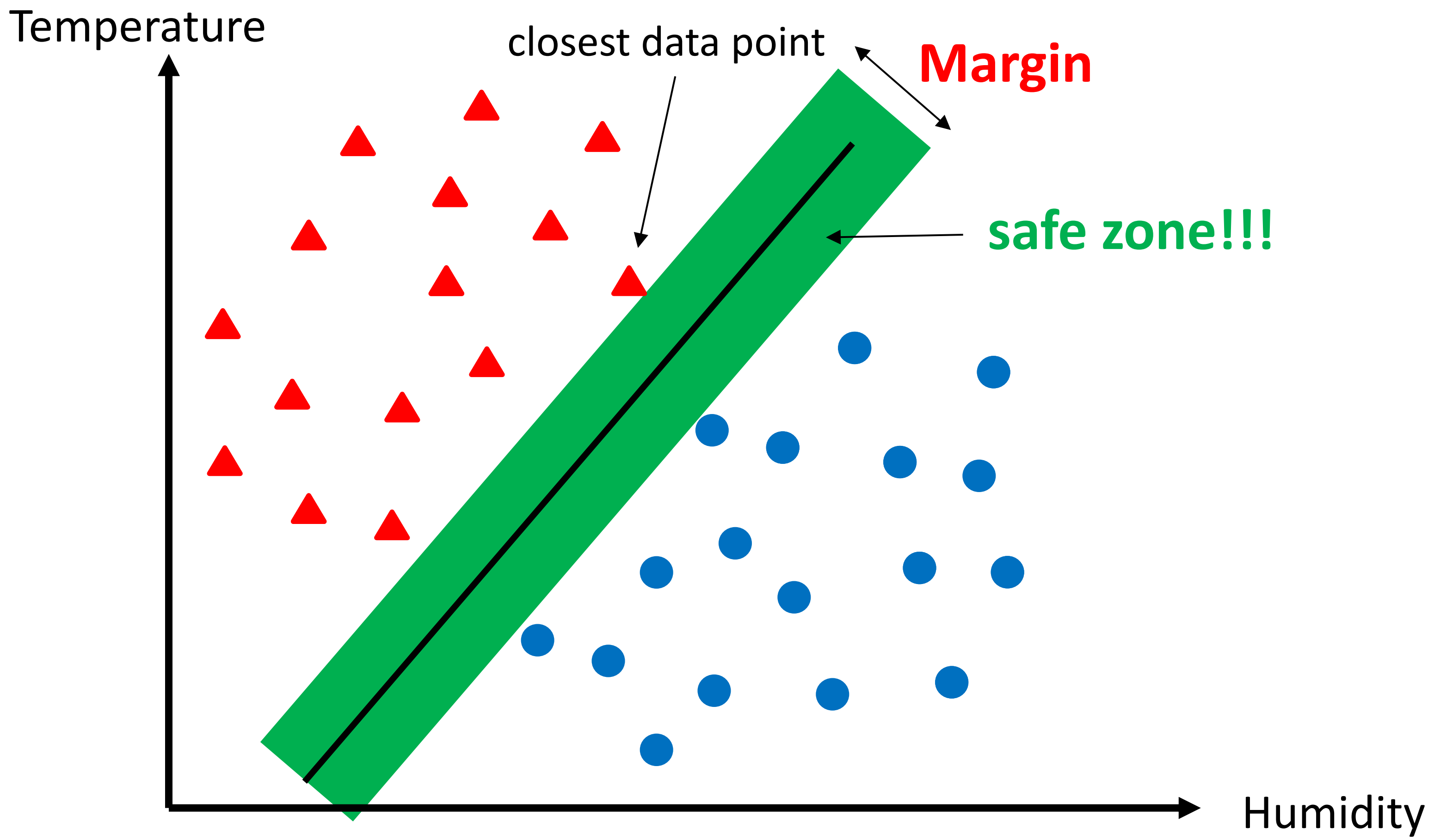
- SVM tries to constructs a ***hyperplane***, or set of hyperplanes in feature space, which can be used for classifying data samples.
- Note: A 2D Hyperplane is a line in 2D space, A 3D Hyperplane is a plane in 3D space, … . In N-dimensional space, we just call it a ***Hyperplane***.
- Which hyperplane is the best classifier?
- Intuitively, a good classifier can be a hyperplane that has the ***largest separation***, or in other word, the ***largest distance to the nearest data samples of any class***. This is called the ***largest margin (or maximum margin)***.

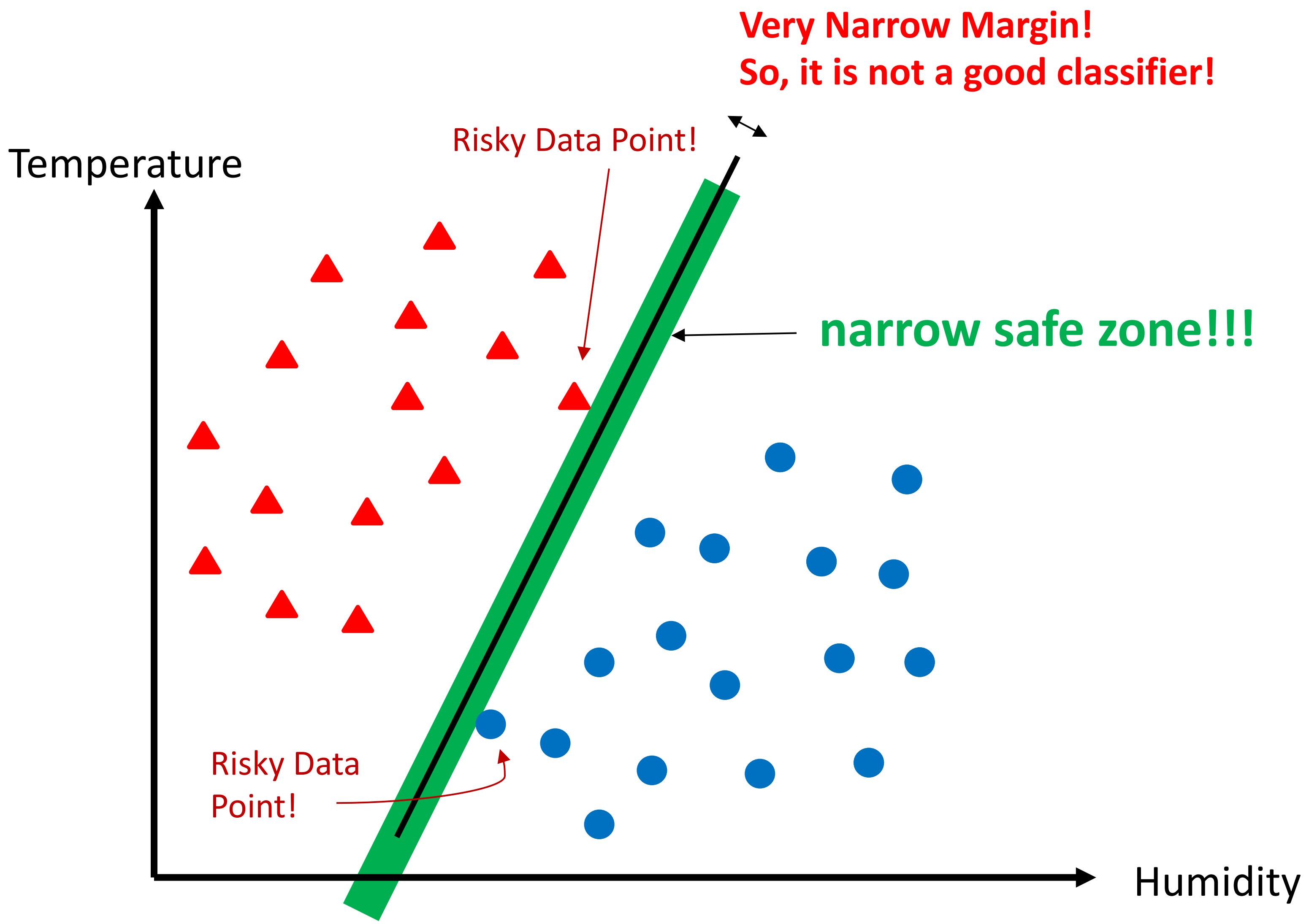

Temperature
closest data point
Margin
Classifier
safe zone!!!
Humidity

# Support Vector Machine (SVM)

- Intuitively, a good classifier can be a hyperplane that has the ***largest separation***, or in other word, the ***largest distance to the nearest data samples of any class***. This is called the ***largest margin (or maximum margin)***.
- Given a decision boundary (hyperplane), we can calculate the distance between the hyperplane and ***the closest data point***. If we double this distance, it is called the ***margin***. Thus, ***margin is the widest boundary area with no data point inside it!***
- **We would like to *maximize the margin* because the data samples near the decision boundary (near the hyperplane) represent very uncertain classification decisions (any small noise or error can move it to the other side of the line!!!)**

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




Very Narrow Margin!
So, it is not a good classifier!
Risky Data Point!
Temperature
narrow safe zone!!!
Risky Data Point!
Humidity

## Matrices and Vectors

- Before we continue and see how ***Support Vector Machine*** algorithm finds the largest margin classifier, we need to briefly review matrices and Vectors main properties and operations.

- We will also use it in ***Dimensionality Reduction, Recommender Systems, Natural Language Processing, …*** .

## Linear Algebra Review: Matrices and Vectors

# Matrix

- **Matrix:** Matrix (plural matrices or matrixes) is a rectangular array (table) of numbers arranged in rows and columns.
- Dimension of matrix: (# of rows) x (# of columns)
- Example: ***M*** is 4x5

$$\boldsymbol{M} = \begin{bmatrix} 2 & 3 & 6 & 78 & 0 \\ 0 & -4 & 23 & 44 & 4 \\ -34 & 4 & 78 & 8 & 2 \\ 7 & 4 & 5 & 0 & 0 \end{bmatrix}$$

## Matrix Elements

- $a_{i,j} = \boldsymbol{A}[i,j]$: element $i,j$ in the $i^{\text{th}}$ row and $j^{\text{th}}$ column.

m-by-n matrix

$$\boldsymbol{A} = \begin{array}{cc} a_{i,j} & \text{n columns} \quad \xrightarrow{\text{j changes}} \\ \begin{array}{c} \text{m rows} \\ \Big\downarrow \text{i changes} \end{array} & \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array}$$

# Matrix Simple Operations

- Addition, Subtraction, Multiplication:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

**Addition: Just add elements**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

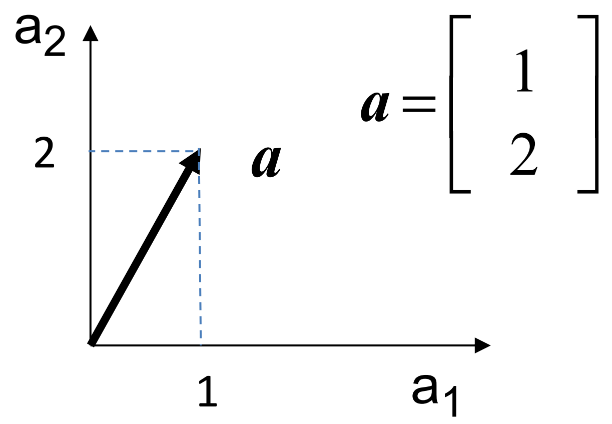
**Subtraction: Just subtract elements**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

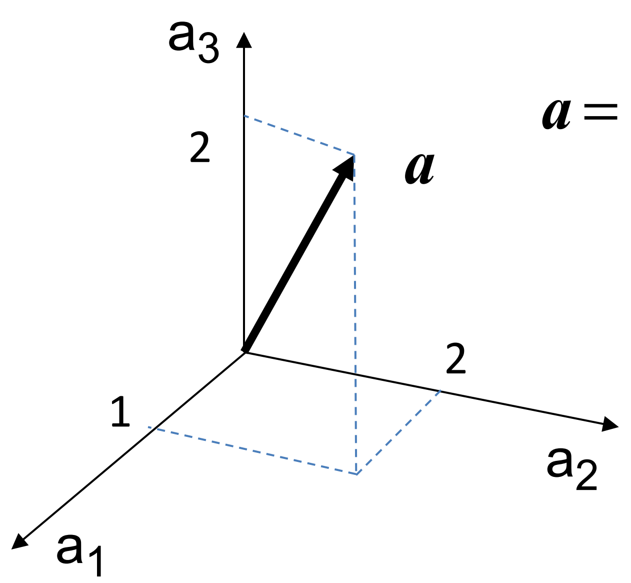
**Multiplication: Multiply each row by each column**

# Vector

- **Vector:** vector is a matrix with only one column (n x 1).
- $a_i = \boldsymbol{v}[i]$: element $i$ in the $i^{\text{th}}$ row.
- We can think of a vector as a ***directed line*** segment in N-dimensions! Thus, a vector has "**Length**" and "**Direction**":



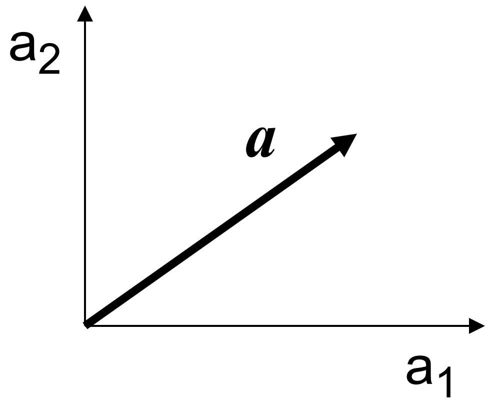
$$\boldsymbol{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



$$\boldsymbol{a} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

## Vector

- Three different representations of a vector:



$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}$$

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_N)$$

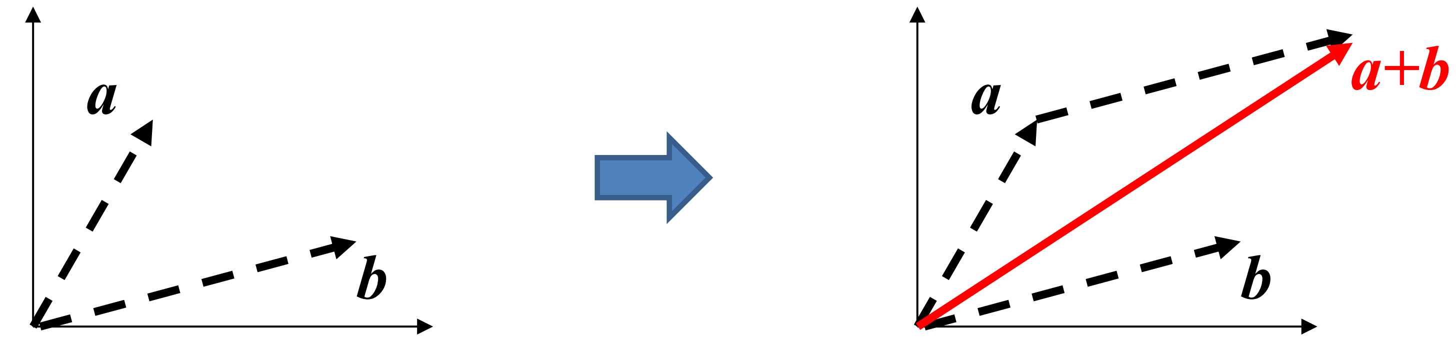
# Notation

- We usually use a **bold** CAPITAL *Italic* letter to name a matrix. E.g. $\boldsymbol{M}$, $\boldsymbol{A}$, …
- We usually use a **bold** lower-case *Italic* letter to name a vector. E.g. $\boldsymbol{v}$, $\boldsymbol{u}$, …
- We usually use a non-bold *Italic* letter to name a scalar (variable). E.g. $s$, $c$, …

# Vector

- **Vector Addition:** Add element-by-element:

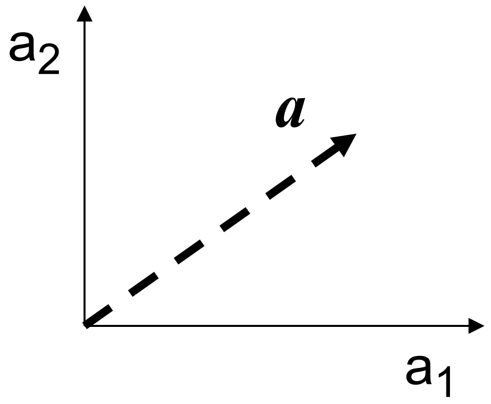
$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{a}+\boldsymbol{b} = \begin{bmatrix} a_1+b_1 \\ a_2+b_2 \\ \vdots \\ a_N+b_N \end{bmatrix}$$

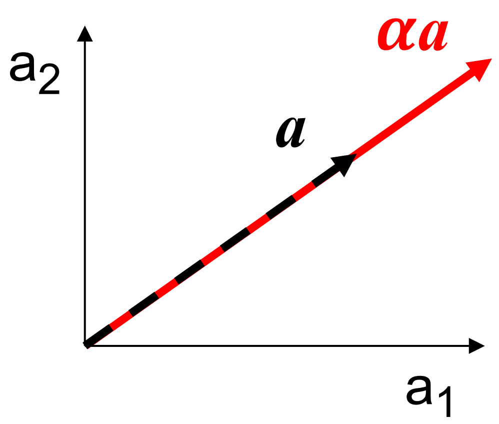


## Vector

- **Multiplying a Vector by a Scalar:**

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \Rightarrow \quad \alpha\boldsymbol{a} = \begin{bmatrix} \alpha a_1 \\ \alpha a_2 \\ \vdots \\ \alpha a_N \end{bmatrix}$$

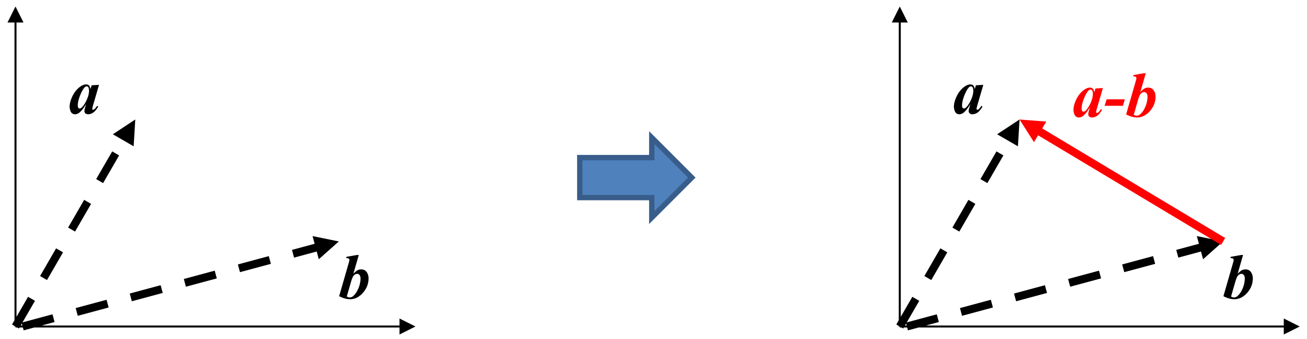




## Vector

- **Vector Subtraction:** Subtract element-by-element:

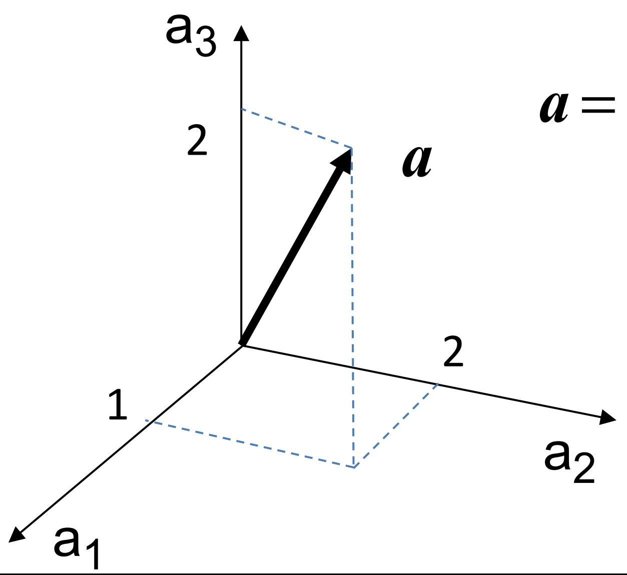
$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{a} - \boldsymbol{b} = \begin{bmatrix} a_1 - b_1 \\ a_2 - b_2 \\ \vdots \\ a_N - b_N \end{bmatrix}$$



# Vector Norm

- **Vector Norm2 (or simply called Vector Norm or Vector Length):**

$$\boxed{\|\boldsymbol{v}\| = \sqrt{\sum_{i=1}^{N} v_i^2}} \qquad \|\boldsymbol{v}\|^2 = \sum_{i=1}^{N} v_i^2$$



$$\boldsymbol{a} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

Vector Length (norm2) $= \sqrt{1^2 + 2^2 + 2^2} = 3$

# Vector Norm

- **Vector Norm2 (or simply called Vector Norm or Vector Length):**

$$\boxed{\left\|\boldsymbol{v}\right\| = \sqrt{\sum_{i=1}^{N} v_i^2}} \qquad \left\|\boldsymbol{v}\right\|^2 = \sum_{i=1}^{N} v_i^2$$

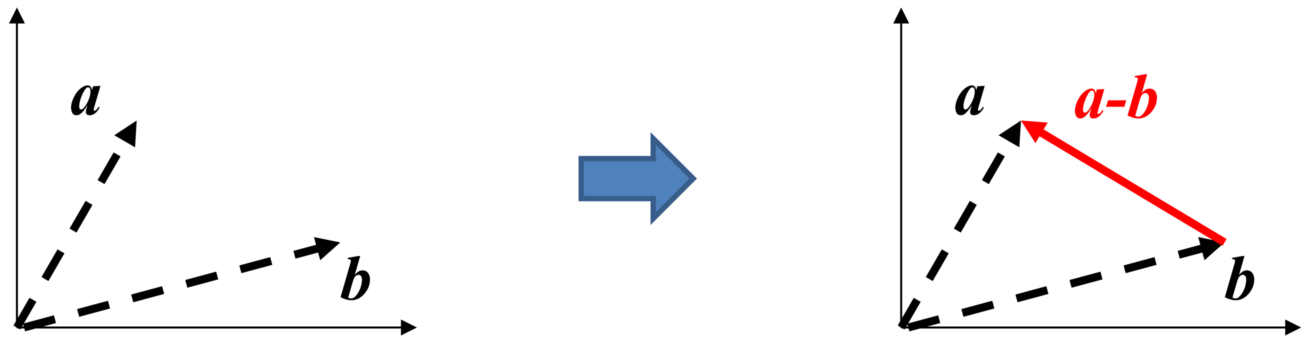
- **Vector Norm1:**

$$\boxed{\left\|\boldsymbol{v}\right\|_1 = \sum_{i=1}^{N} \left| v_i \right|}$$

## Distance Between Vectors

- **Distance Between Vectors**: The distance between two vectors in a vector space is defined by:

$$d(\boldsymbol{a},\boldsymbol{b}) = \left\|(\boldsymbol{a}-\boldsymbol{b})\right\|$$



**Note:** $d(\boldsymbol{a},\boldsymbol{b}) = 0$ if and only if $\boldsymbol{a} = \boldsymbol{b}$.

# Vector Transposition

- **Transpose**: reflect vector/matrix on line (***replace columns with rows!***):

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \Rightarrow \boldsymbol{a}^T = [a_1 \; a_2 \; \dots \; a_N]$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

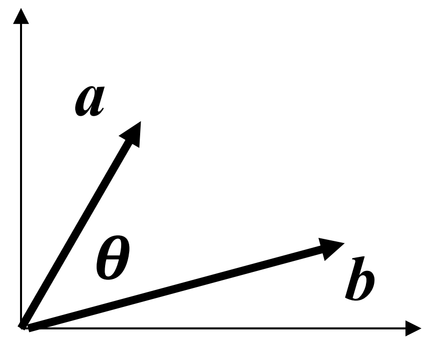
# Inner Product (Dot Product)

- **Inner Product Between Vectors :** Define the inner product between two N-dimensional vectors by:

$$\boldsymbol{a}.\boldsymbol{b} = \sum_{i=1}^{N} a_i\, b_i$$

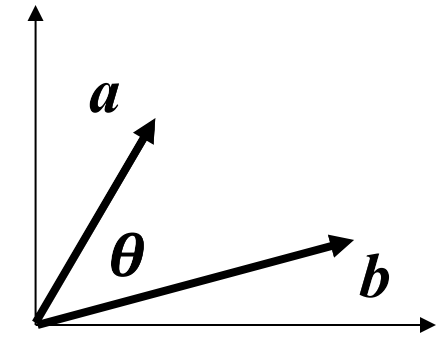
$$= \left\|\boldsymbol{a}\right\| \times \left\|\boldsymbol{b}\right\| \times \mathrm{Cos}\,\theta$$

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$



# Inner Product (Dot Product)

$$\boldsymbol{a}.\boldsymbol{b} = \sum_{i=1}^{N} a_i\, b_i = \|\boldsymbol{a}\| \times \|\boldsymbol{b}\| \times \mathrm{Cos}\,\theta$$



$$\Rightarrow \quad \mathrm{Cos}\,\theta = \frac{\boldsymbol{a}.\boldsymbol{b}}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|} = \frac{\sum_{i=1}^{N} a_i\, b_i}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|}$$

- (i) This lies between -1 and 1.
- (ii) It measures directional **SIMILARITY** of ***a*** and ***b***

  = +1 when ***a*** and ***b*** point to the same direction.

  = 0 when ***a*** and ***b*** are a “right angle”.

  = -1 when ***a*** and ***b*** point to opposite directions.

*Thank You!*

**Questions?**