

# Support Vector Machines and Decision Tree

CSIT375 AI for Cybersecurity

Dr Wei Zong

SCIT University of Wollongong

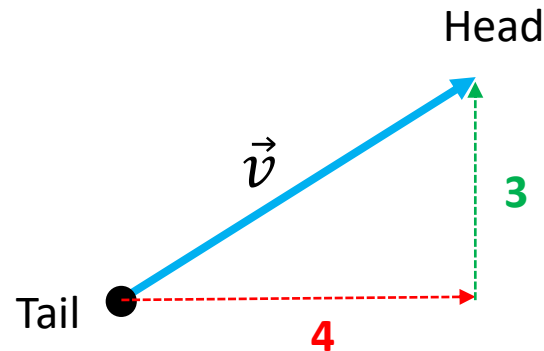
Disclaimer: The presentation materials  
come from various sources. For further  
information, check the references section

# Outline

- Preliminaries
  - vector magnitude
  - vector normalization
  - vector inner product – geometric point of view
- Introduction to Linear SVM
  - Our focus: Geometric intuition
- Malware detection
  - Common types of Malware
  - Introduction to malware analysis methodology
  - Decision tree
    - Information Gain
    - Build a Decision Tree

# Vectors

- A vector can be visually represented as an arrow



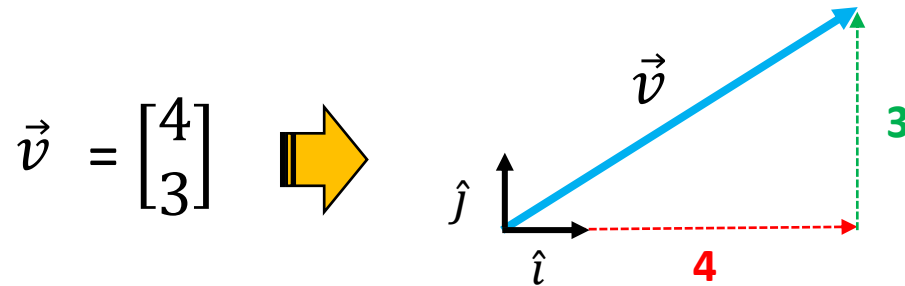
$$\vec{v} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Or:  $\vec{v} = (4, 3)$

- A vector can also be represented as an ordered list or a tuple by starting from its tail and asking how far away is its head in the:
  - Horizontal direction
  - Vertical direction

# Unit Vectors

- Any vector can also be represented as a *sum* of scaled up versions of specific **unit** vectors



$$\hat{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

It goes in the **horizontal direction** only and has length 1

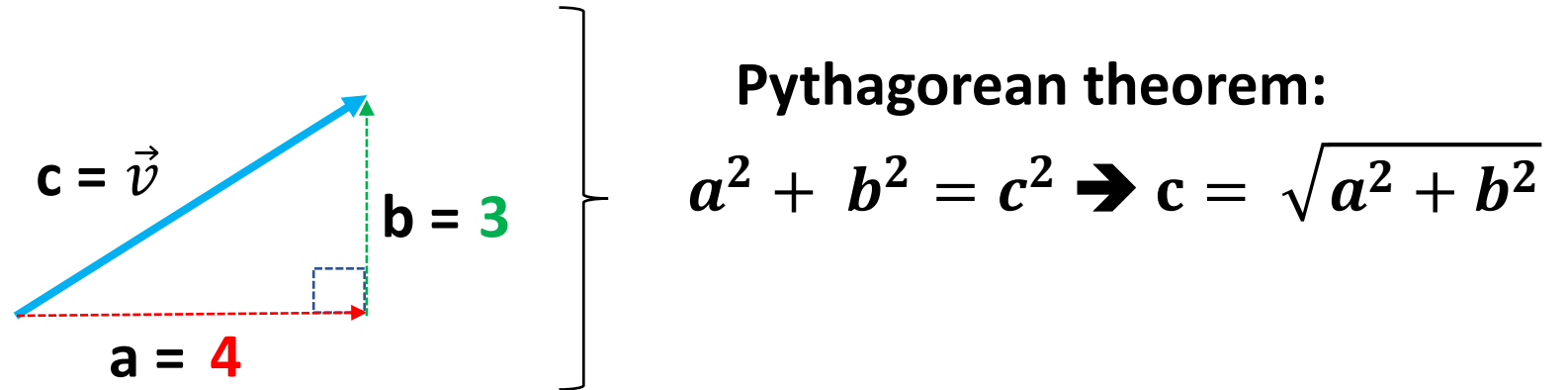
$$\hat{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

It goes in the **vertical direction** only and has length 1

$$\vec{v} = 4 \times \hat{i} + 3 \times \hat{j} = 4 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

# Vector Magnitude

- How can we calculate the length (or *magnitude*) of a vector?

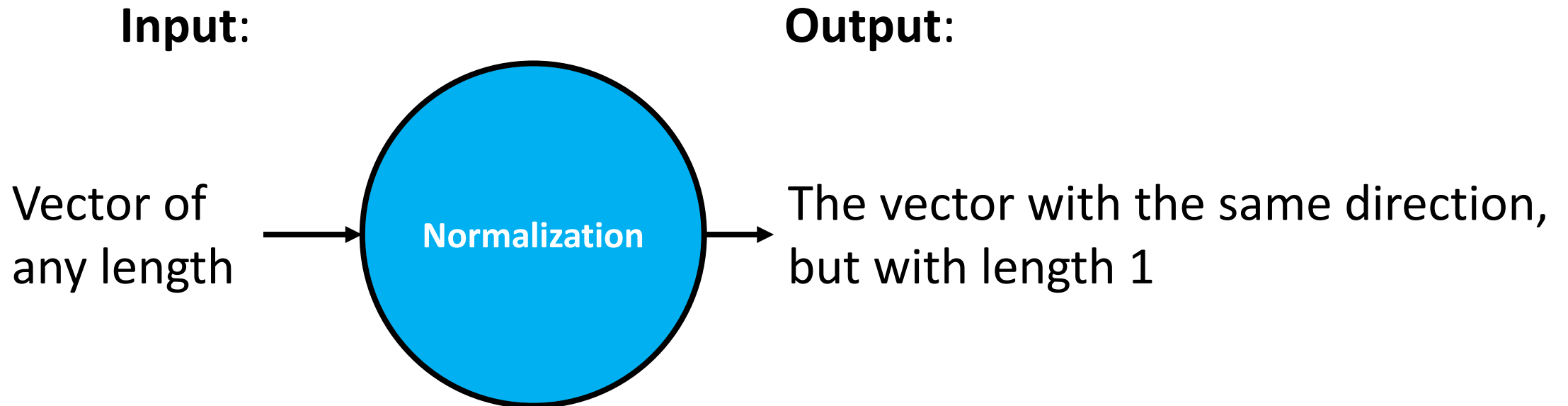


$$\text{Magnitude of the vector} = \|\vec{v}\| = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

How can we keep vector,  $\vec{v}$ , pointing to the same direction, but change its magnitude to 1 (i.e., turn it into a unit vector)?

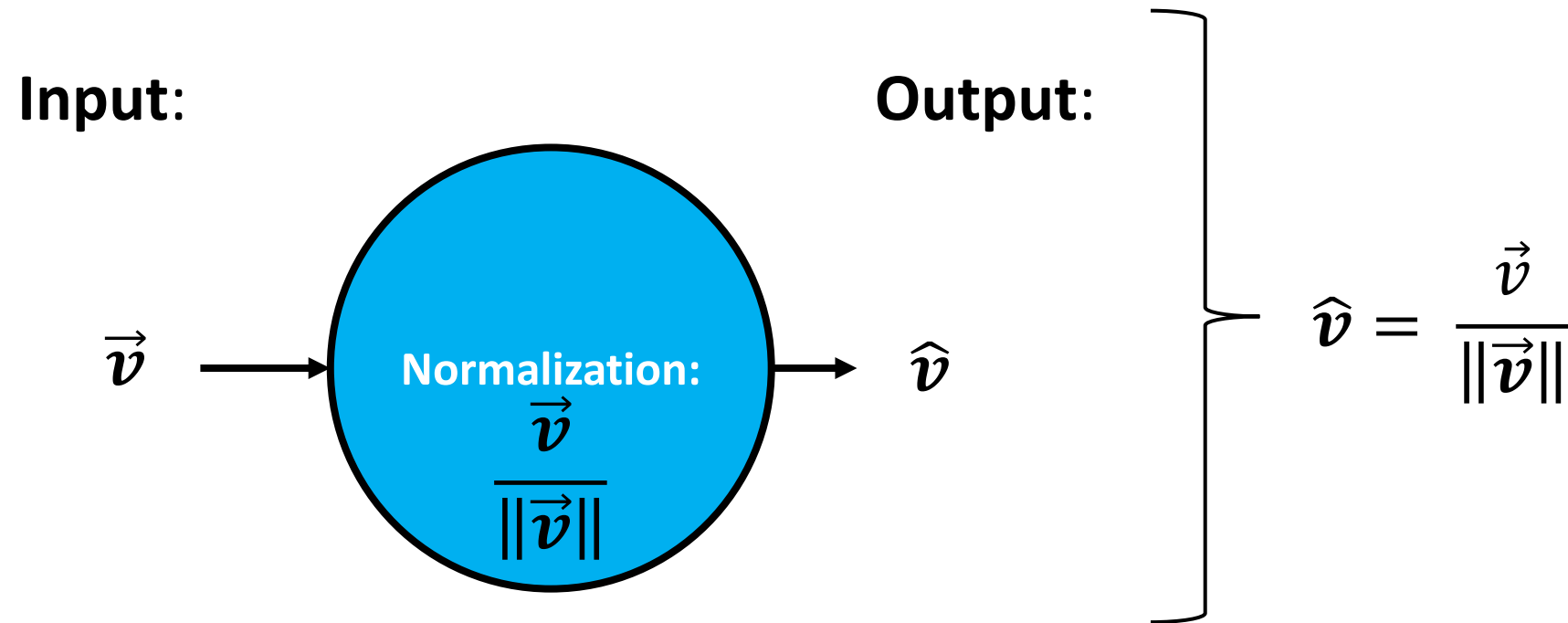
# Vector Normalization

- How can we construct a unit vector out of a given vector of any length?



# Vector Normalization

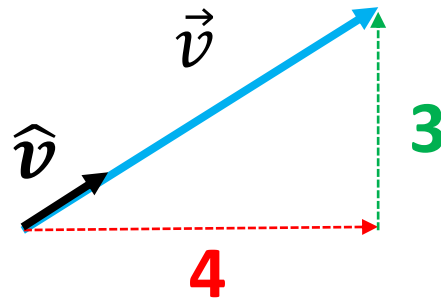
- How can we construct a unit vector out of a given vector of any length?



# Vector Normalization

- How can we construct a unit vector out of a given vector of any length?

$$\vec{v} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \Rightarrow$$



$$\|\vec{v}\| = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

$$\hat{v} = \frac{\vec{v}}{\|\vec{v}\|} = \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix}$$

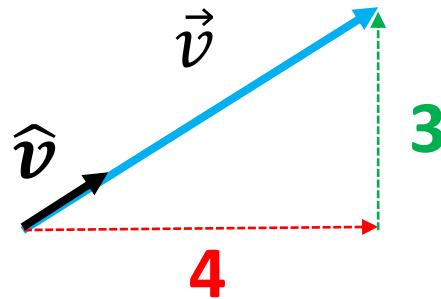
Let us verify that its length is 1



# Vector Normalization

- How can we construct a unit vector out of a given vector of any length?

$$\vec{v} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \Rightarrow$$



$$\|\vec{v}\| = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

$$\left. \begin{aligned} \hat{v} &= \frac{\vec{v}}{\|\vec{v}\|} = \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix} \\ \|\hat{v}\| &= \sqrt{\left(\frac{3}{5}\right)^2 + \left(\frac{4}{5}\right)^2} \\ &= \sqrt{(9 + 16)/25} = 1 \end{aligned} \right\}$$

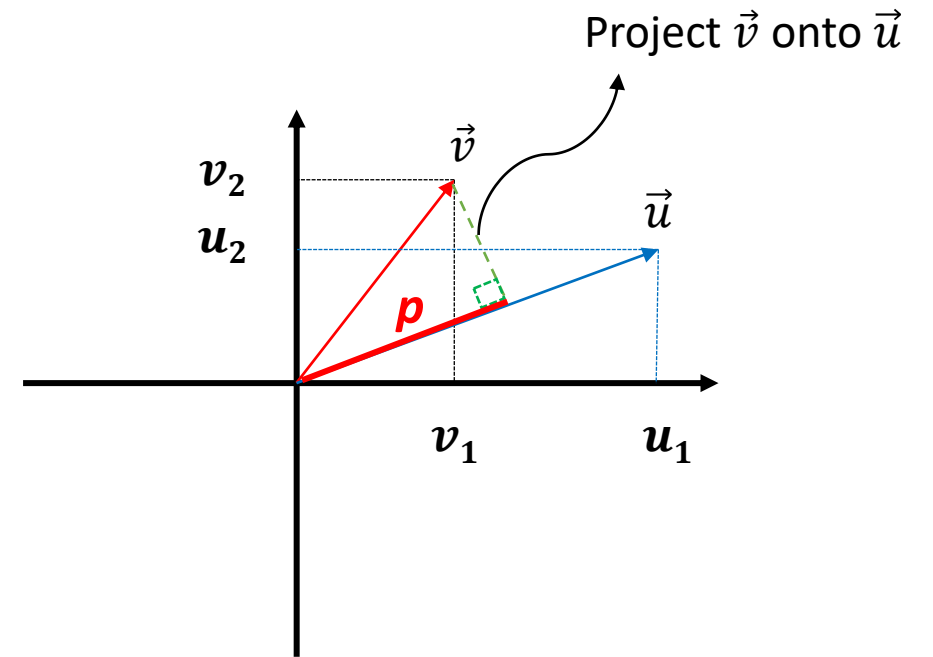
# Vector Inner Product

- Assume the following two vectors:

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- What is the inner product of  $\vec{u}$  and  $\vec{v}$ ?

$$\vec{u}^T \vec{v} = ?$$



$p$  is the length of the projection of  $\vec{v}$  onto  $\vec{u}$

# Vector Inner Product

- Assume the following two vectors:

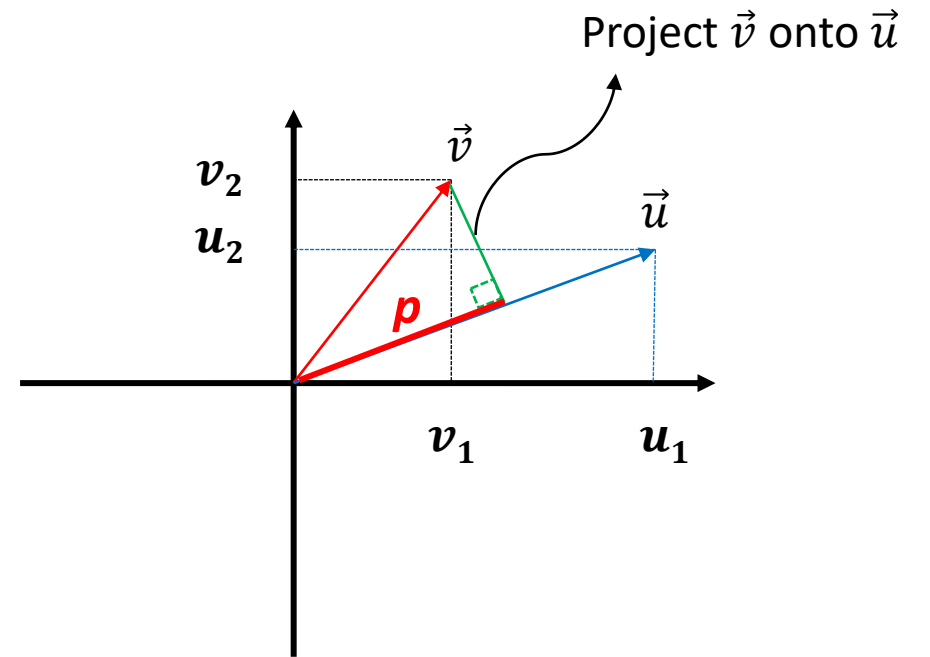
$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- What is the inner product of  $\vec{u}$  and  $\vec{v}$ ?

$$\vec{u}^T \vec{v} = \textcolor{red}{p} \times \|\vec{u}\|$$

≡

$$\begin{aligned} \vec{u}^T \vec{v} &= [u_1 \ u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ &= u_1 \times v_1 + u_2 \times v_2 \end{aligned}$$



$\textcolor{red}{p}$  is the length of the projection of  $\vec{v}$  onto  $\vec{u}$   
→ Can be signed

# Vector Inner Product

- Assume the following two vectors:

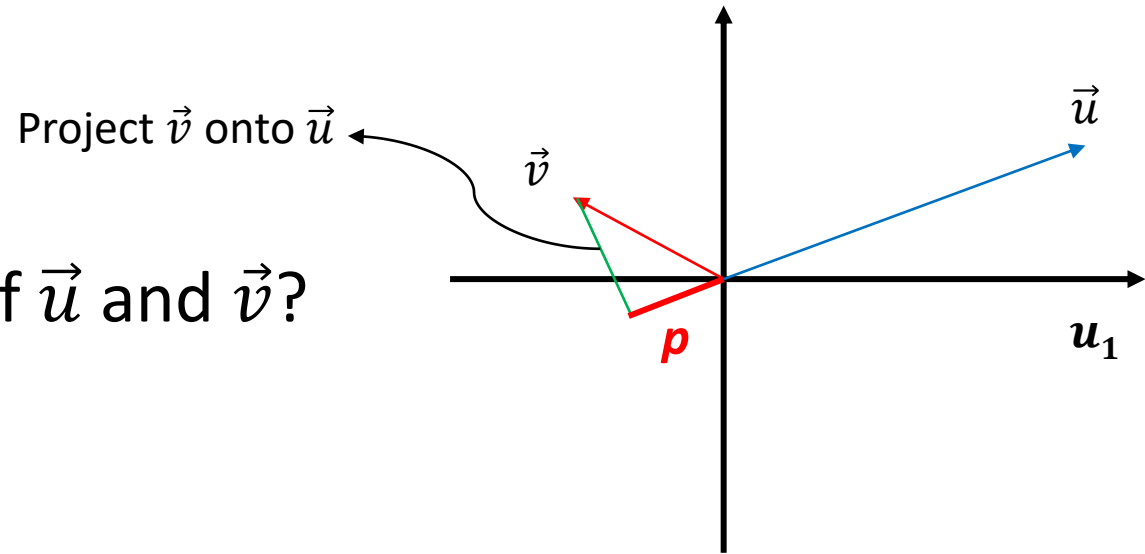
$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- What is the inner product of  $\vec{u}$  and  $\vec{v}$ ?

$$\vec{u}^T \vec{v} = \textcolor{red}{p} \times \|\vec{u}\|$$

$\equiv$

$$\begin{aligned} \vec{u}^T \vec{v} &= [u_1 \ u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ &= u_1 \times v_1 + u_2 \times v_2 \end{aligned}$$



$\textcolor{red}{p}$  is negative here

# Recall: Perceptron

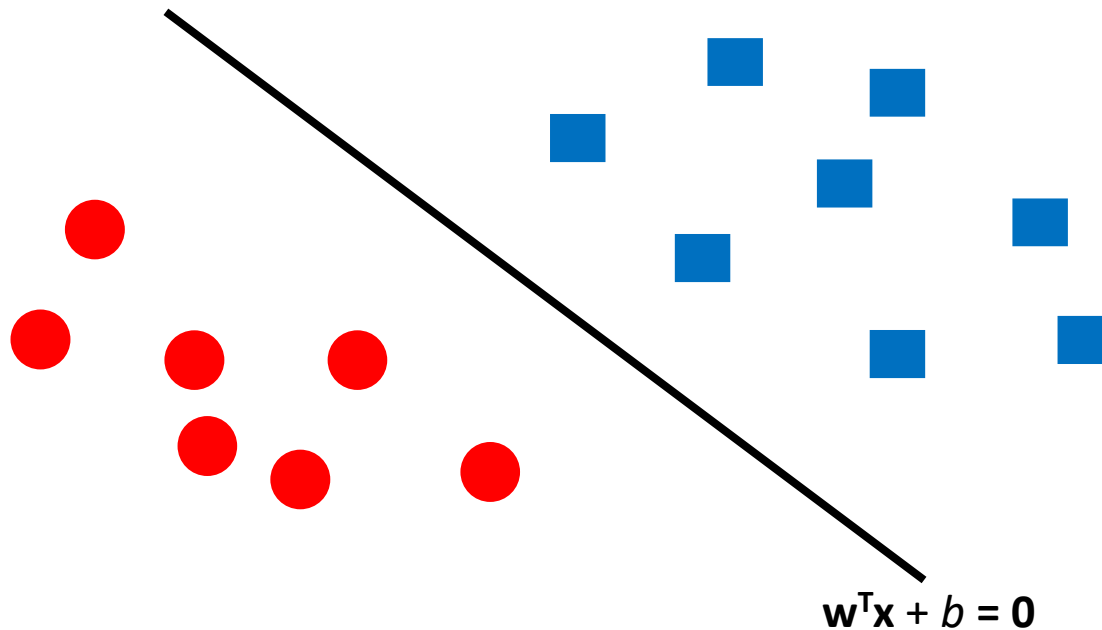
- Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ 
  - $\{x^1, x^2, \dots, x^n\}$  : data set, and  $y^i \in \{1, -1\}$  be the class label
- Perceptron
  - a binary classifier: maps its input to an output value  $f(\mathbf{x})$ :

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1, & \text{if } \mathbf{w}^T \mathbf{x} + b < 0 \end{cases}$$

- parameters:  $\mathbf{w}$  is a vector of weights,  $b$  is the bias
- linear classifier: a classification algorithm that makes its predictions based on a linear function combining a set of weights with the feature vector.
- find a hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$  that minimizes the number of misclassified points by varying  $\mathbf{w}$  and  $b$

# Perceptron: Intuition

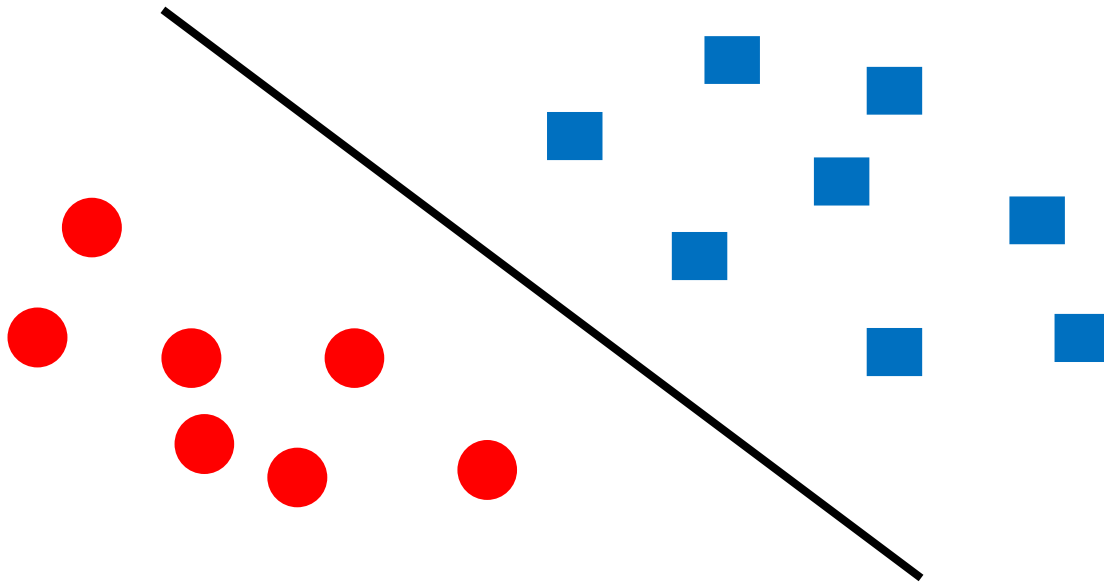
- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes



**A perceptron divides a space by a hyperplane into two half-spaces**

# Perceptron: Intuition

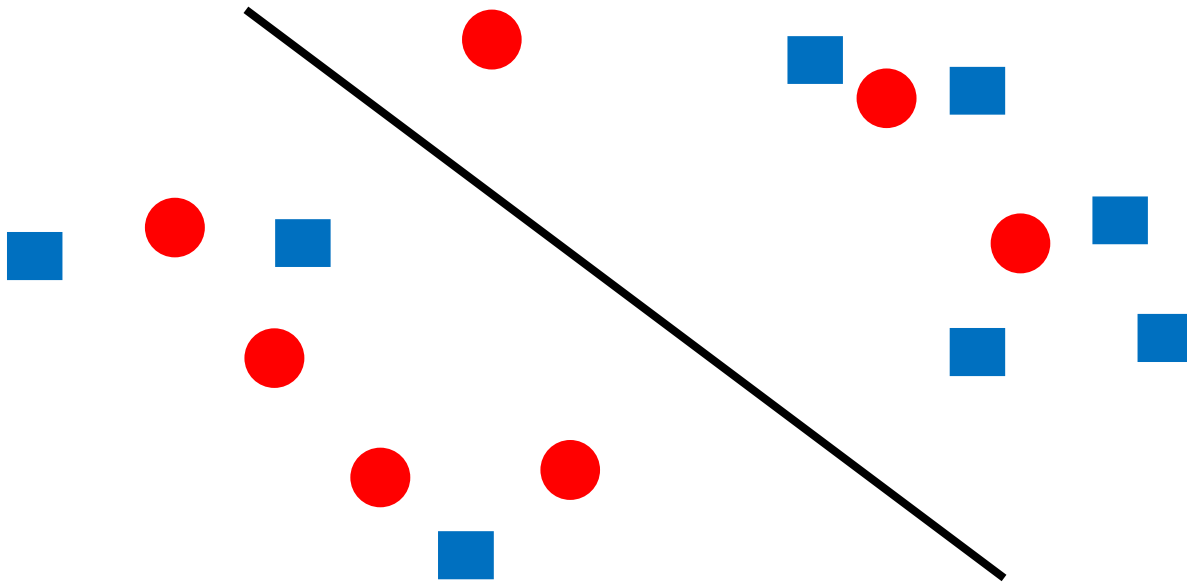
- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes



This entails that the space has to be *linearly separable* (or otherwise, will not be able to correctly classify all examples)

# Perceptron: Intuition

- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes

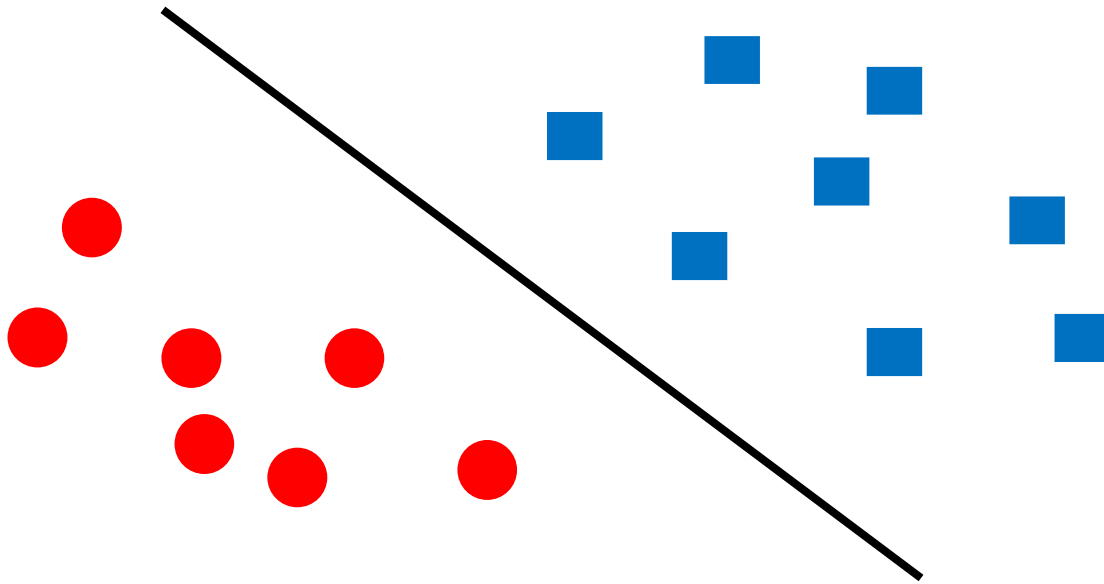


**NOT a linearly separable space,  
hence, perceptron will not  
be effective!**



# Perceptron: Intuition

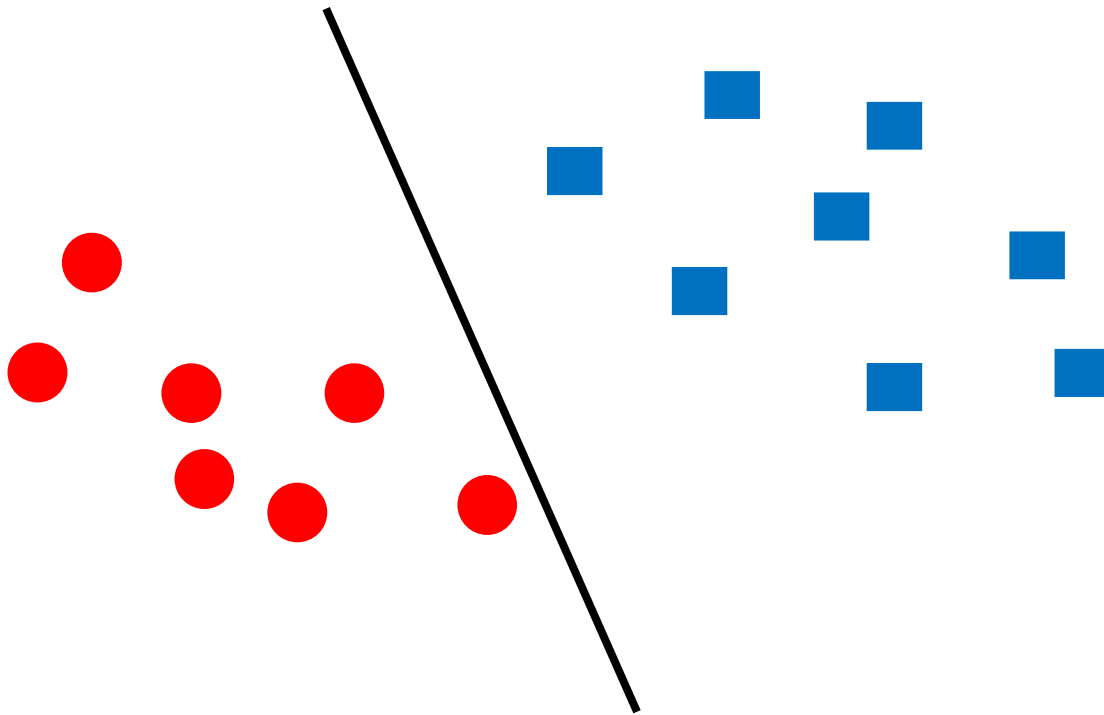
- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes



A *linearly separable* space and  
a workable perceptron

# Perceptron: Intuition

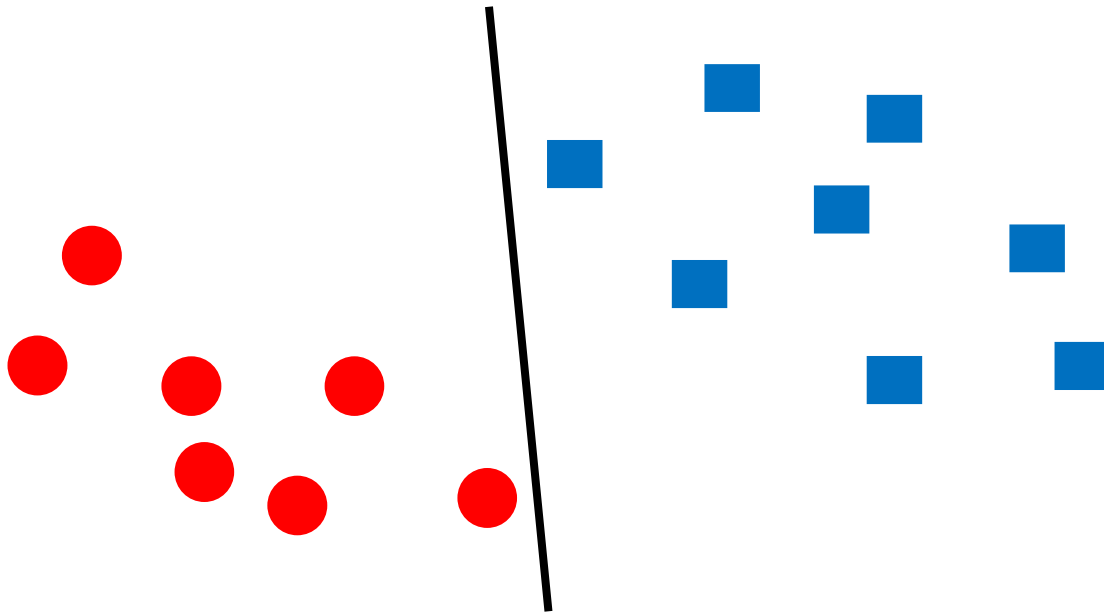
- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes



A *linearly separable* space and another workable perceptron!

# Perceptron: Intuition

- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes

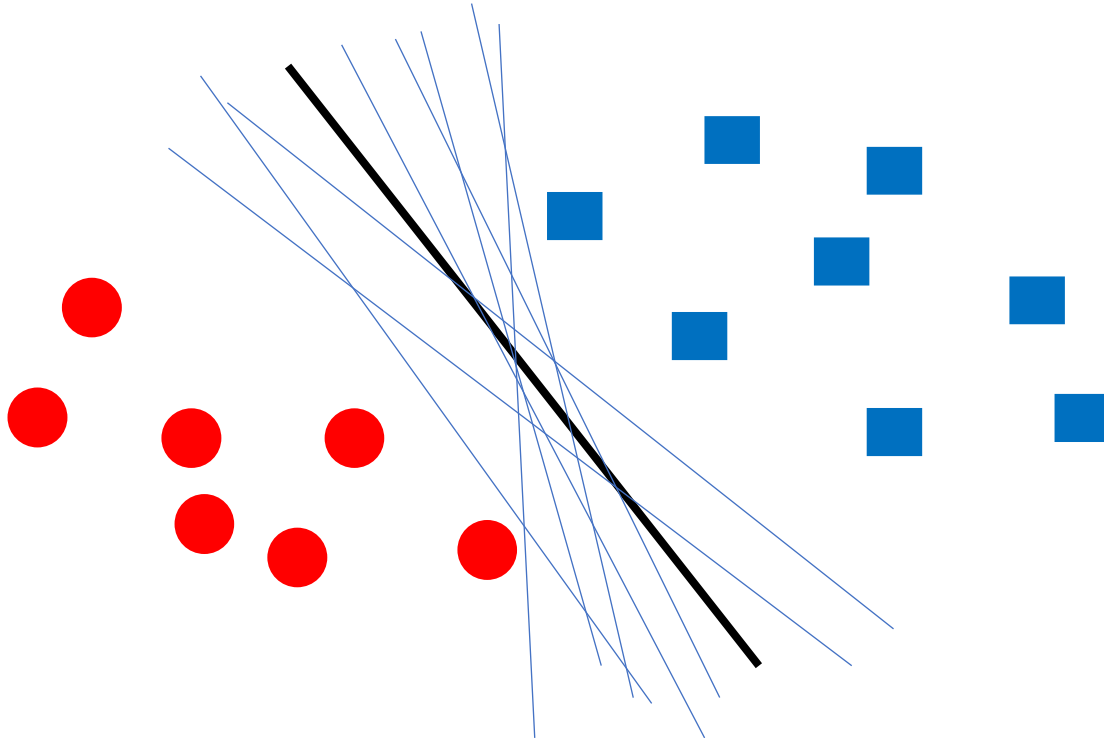


**Yet, another valid hyperplane that can be learnt by a perceptron!**

*If there are many hyperplanes, the perceptron will converge to one of them & classify correctly all examples*

# Perceptron: Intuition

- The way a perceptron works is by learning a hyperplane that clearly separates examples into two classes

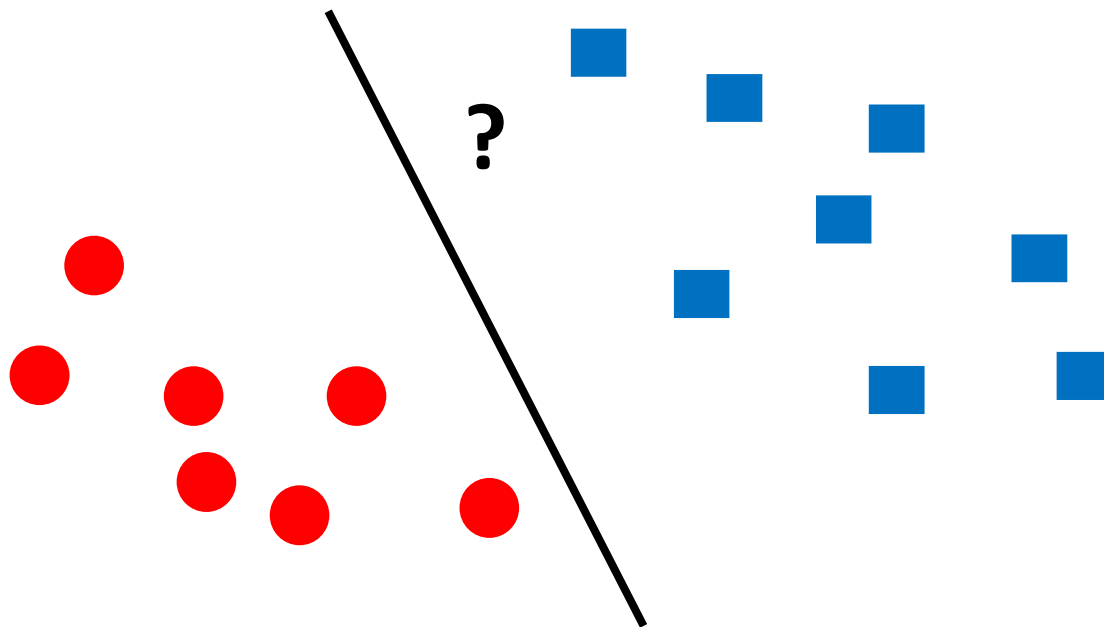


Any of these would be fine..

..but which is best?

# Limitations of Perceptron

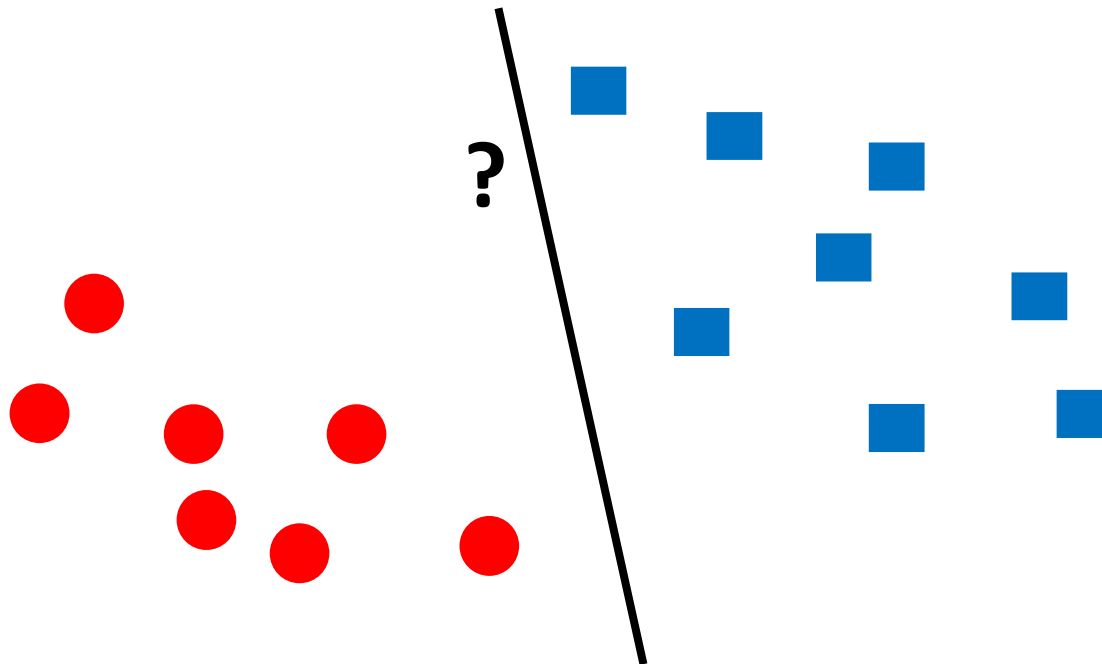
- Perceptrons exhibit various limitations in their ability to classify data
  - There could be many hyperplanes, which are not all equally good



**An acceptable hyperplane and the new example indicated by “?” will be classified as a **square****

# Limitations of Perceptron

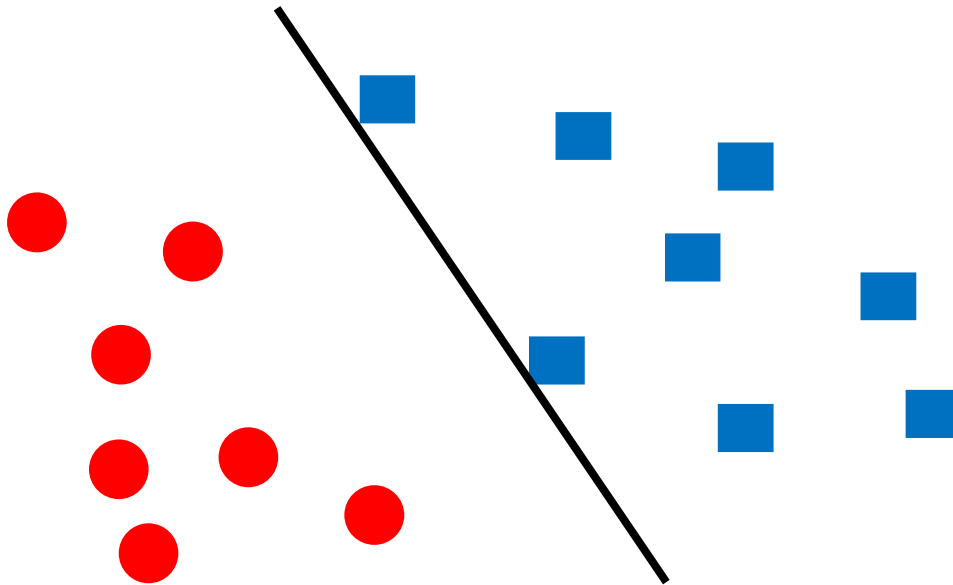
- Perceptrons exhibit various limitations in their ability to classify data
  - There could be many hyperplanes, which are not all equally good



**Another acceptable hyperplane, but the new example will now be classified as a **circle** (although it seems closer to **squares**!)**

# Limitations of Perceptron

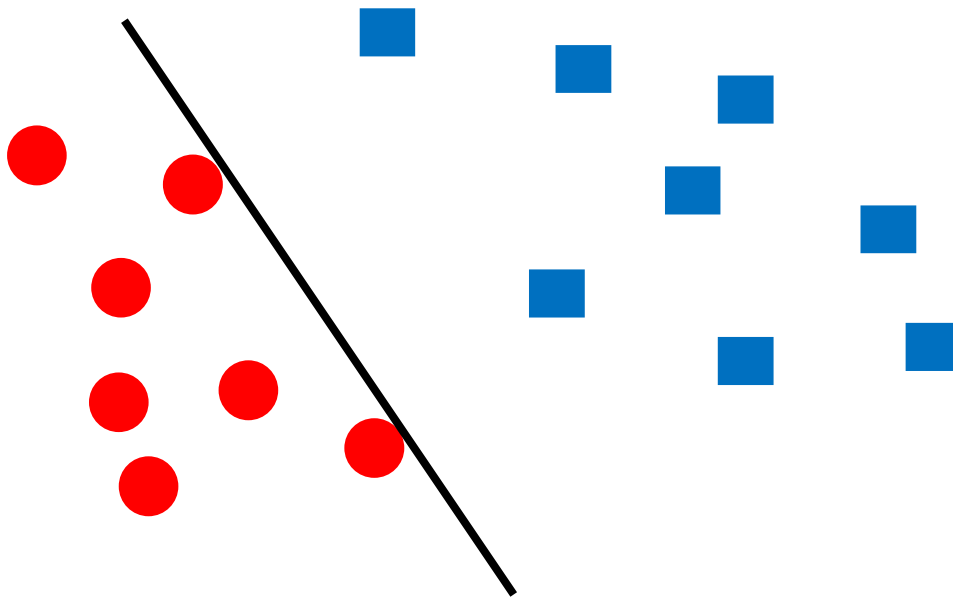
- Perceptrons exhibit various limitations in their ability to classify data
  - another problem is that perceptrons usually stop as soon as there are no misclassified examples



This hyperplane *just* managed to accommodate the two **squares** it touches before stopped

# Limitations of Perceptron

- Perceptrons exhibit various limitations in their ability to classify data
  - another problem is that perceptrons usually stop as soon as there are no misclassified examples

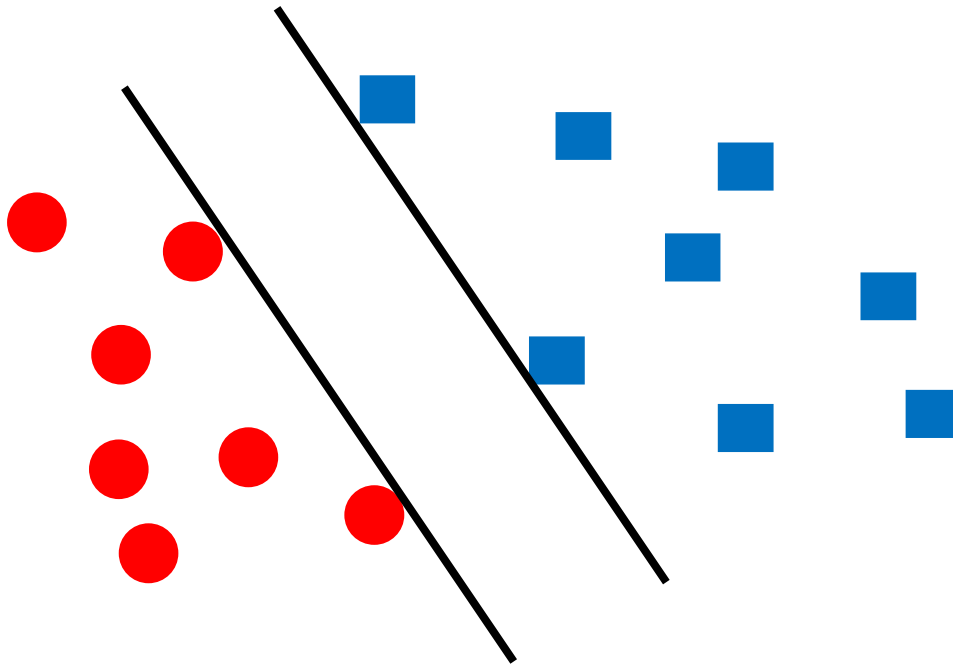


This hyperplane also *just* managed to accommodate the two **circles** it touches before stopped



# Limitations of Perceptron

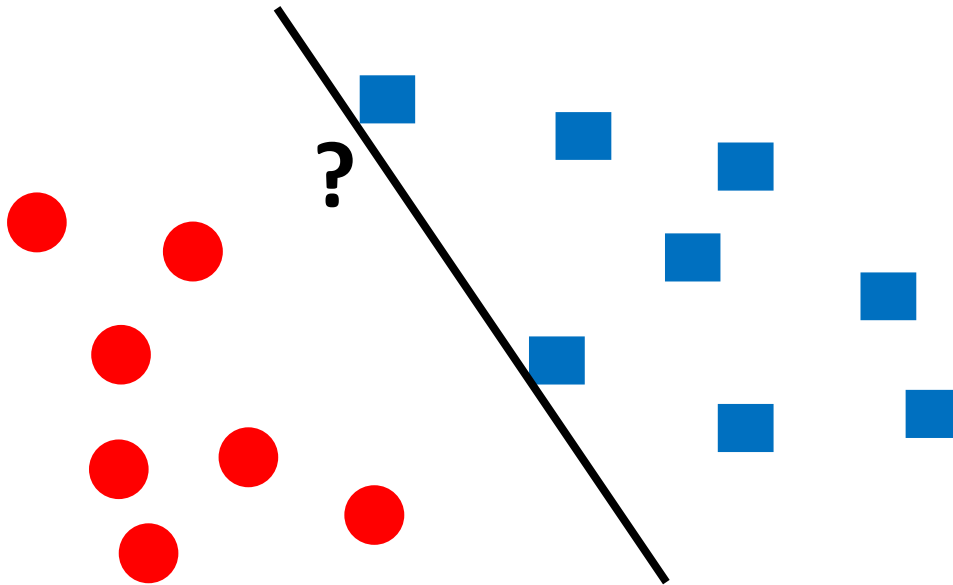
- Perceptrons exhibit various limitations in their ability to classify data
  - another problem is that perceptrons usually stop as soon as there are no misclassified examples



**If either of these hyperplanes represents the final weight vector, the weights will be biased toward one of the classes**

# Limitations of Perceptron

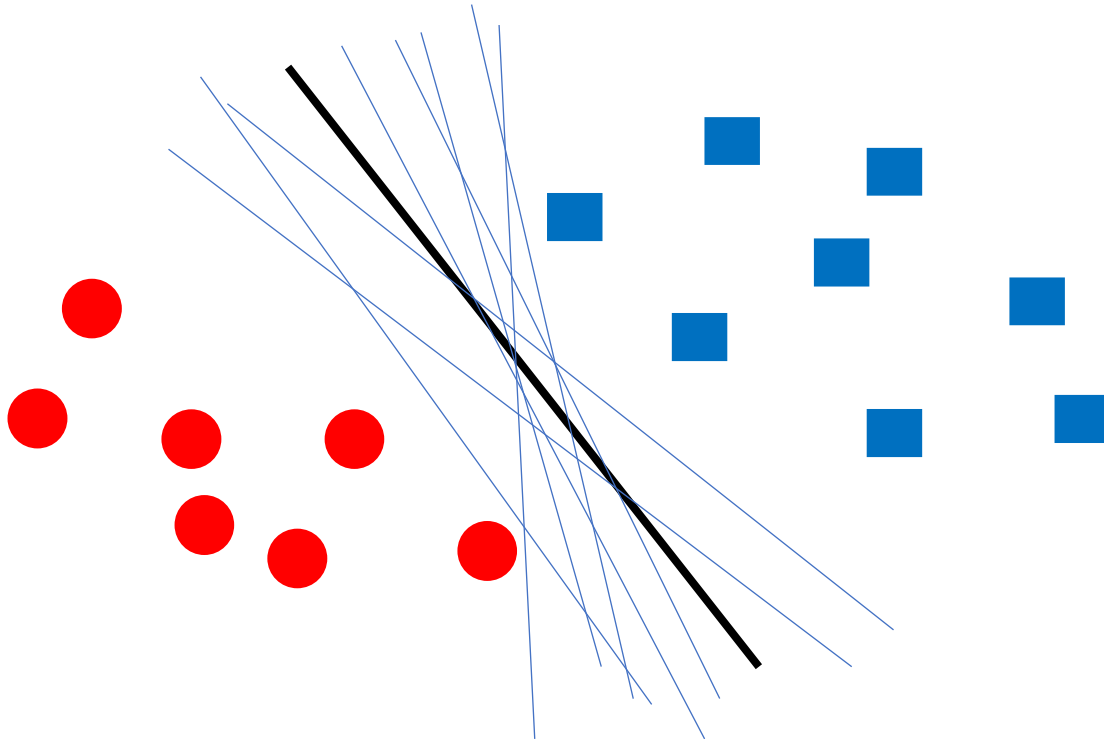
- Perceptrons exhibit various limitations in their ability to classify data
  - another problem is that perceptrons usually stop as soon as there are no misclassified examples



**For example, if this hyperplane is the one that the perceptron chooses, the example indicated by “?” will be classified as a **circle****

# Limitations of Perceptron

- Perceptrons exhibit various limitations in their ability to classify data

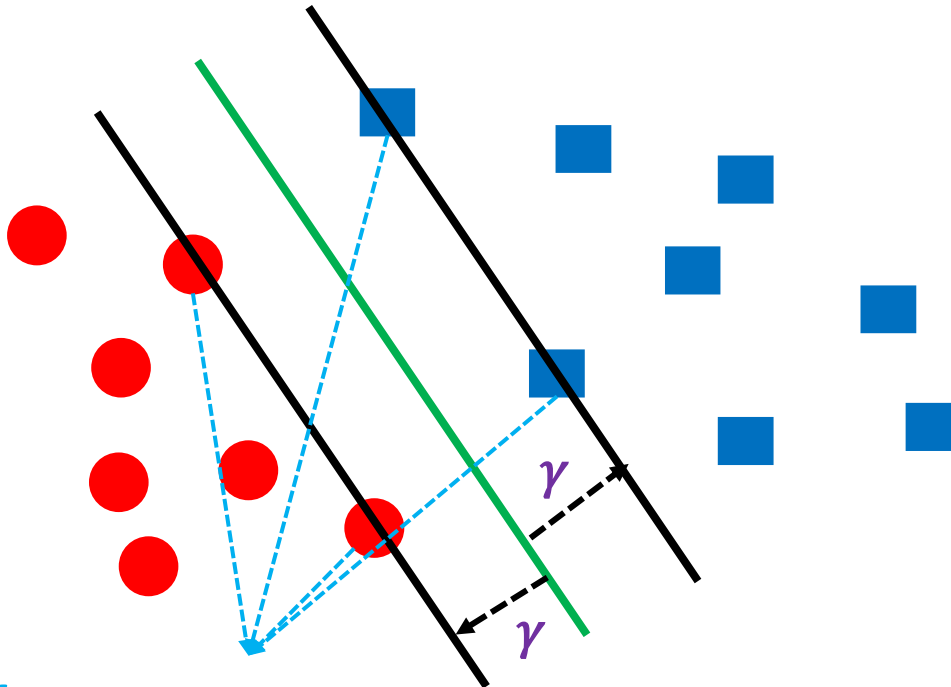


Any of these would be fine..

..but which is best?

# Support Vector Machines

- Linear Support Vector Machines (Linear SVM)
  - Maximum margin linear classifier

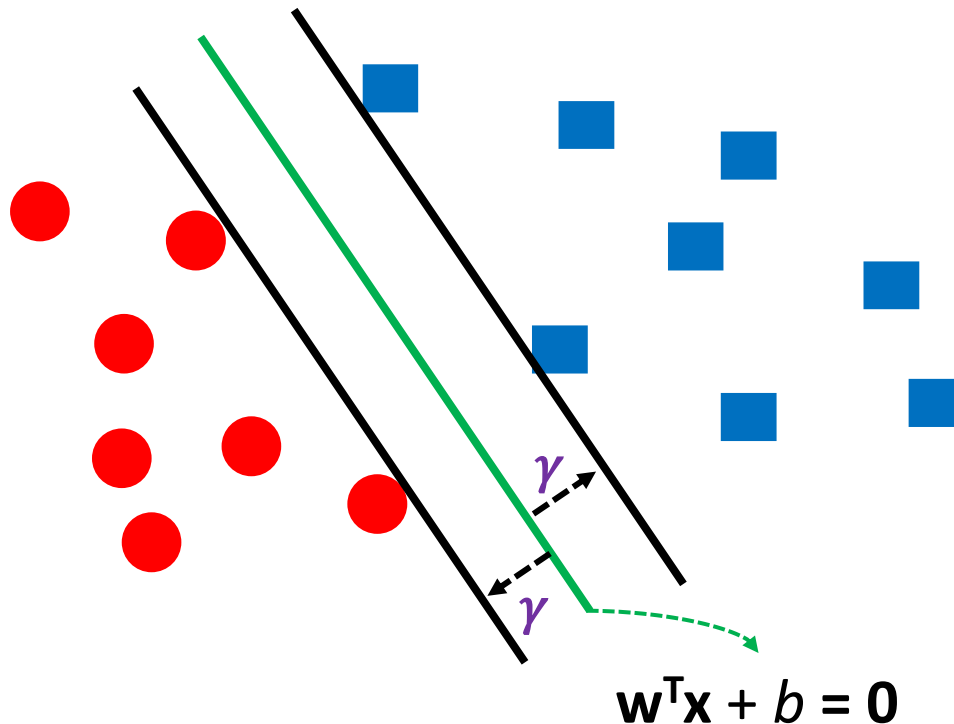


SVM selects one particular hyperplane (i.e. decision boundary – **the green line in the figure**) that not only separates the examples into two classes, but does so in a way that maximizes the **margin**: which is measured by the distance between the hyperplane and the closest examples of the training set

**Support Vectors:** subset of the data which defines the position of the separator

# The Objective of SVM

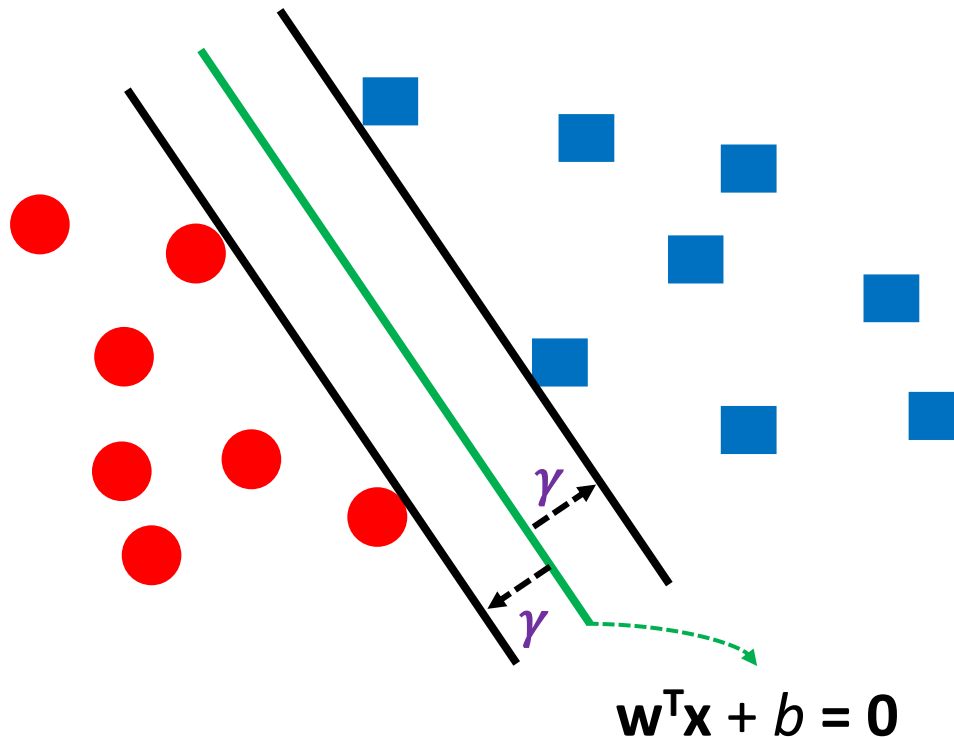
- The objective of an SVM is to select a hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$  that maximizes the distance,  $\gamma$ , between the hyperplane and examples in the training set



Intuitively, we are more certain of the class of examples that are far from the separating hyperplane than we are of examples near to that hyperplane

# The Objective of SVM

- The objective of an SVM is to select a hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$  that maximizes the distance,  $\gamma$ , between the hyperplane and examples in the training set



Thus, it is desirable that all the training examples be as far from the hyperplane as possible  
*(but on the correct side of that hyperplane, of course!)*

# Finding the Decision Boundary

- Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ 
  - $\{x^1, x^2, \dots, x^n\}$ : our data set
  - $y^i \in \{1, -1\}$ : the class label
- Classifier:  $f(\mathbf{x}^i) = \text{sign}(\mathbf{w}^T \mathbf{x}^i + b)$ 
  - $\mathbf{w}$ : *weight vector*
  - $b$ : *bias*
- The SVM decision boundary aims to
  - classify all points correctly
  - maximize the margin (by varying weight vector  $\mathbf{w}$  and bias  $b$ )
- The decision boundary should be as far away from the data of both classes as possible

# Computing the margin width

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow

$$\mathbf{w}^T \mathbf{x}^i + b \geq 1 \text{ if } y^i = +1$$

$$\mathbf{w}^T \mathbf{x}^i + b \leq -1 \text{ if } y^i = -1$$

- For support vectors, the inequality becomes an equality
- The margin is:  $2\gamma = \frac{2}{\|\mathbf{w}\|}$
- Maximizing the margin is equivalent to minimizing  $\|\mathbf{w}\|$

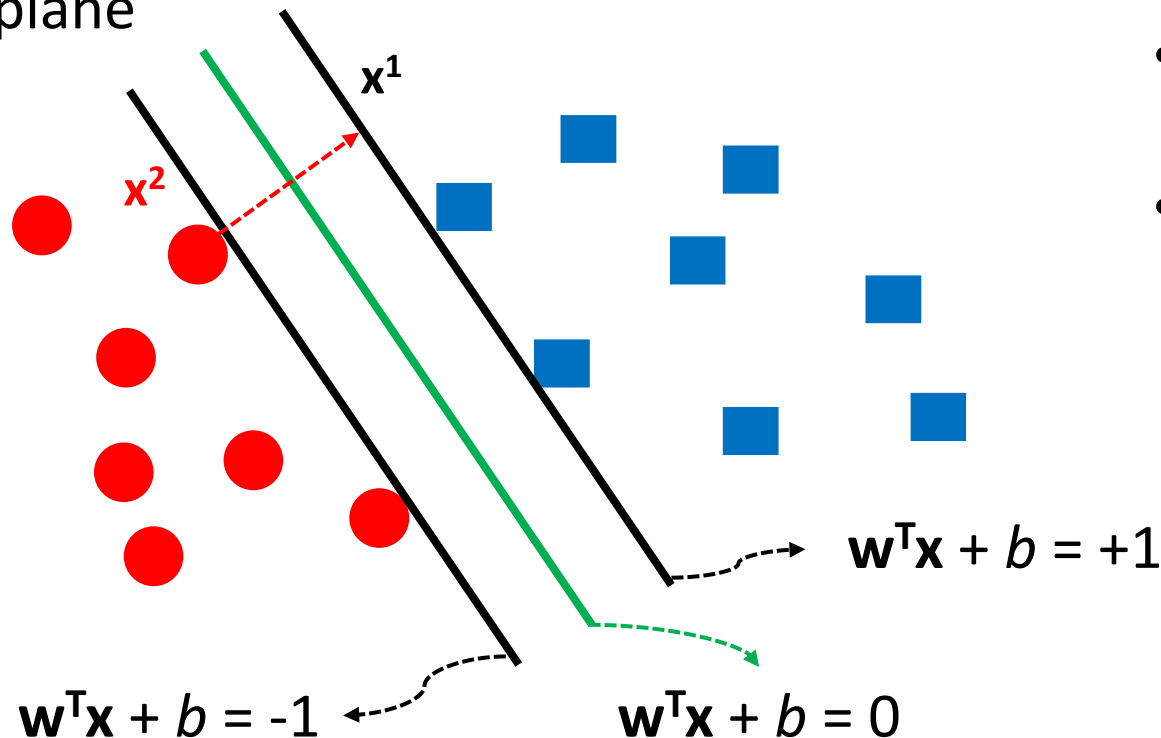


# Computing the margin width

- How do we compute  $\gamma$  in terms of  $\mathbf{w}$  and  $b$ ?
- Claim: The vector  $\mathbf{w}$  is perpendicular (orthogonal) to the hyperplane  $\mathbf{w} \cdot \mathbf{x} + b = 0$ 
  - If  $P$  and  $Q$  are in the plane with equation  $\mathbf{w} \cdot \mathbf{x} + b = 0$ , then  $\mathbf{w} \cdot P = -b$  and  $\mathbf{w} \cdot Q = -b$ , so  $\mathbf{w} \cdot (Q - P) = 0$ .
  - This means that the vector  $\mathbf{w}$  is orthogonal to any vector  $PQ$  between points  $P$  and  $Q$  of the plane.

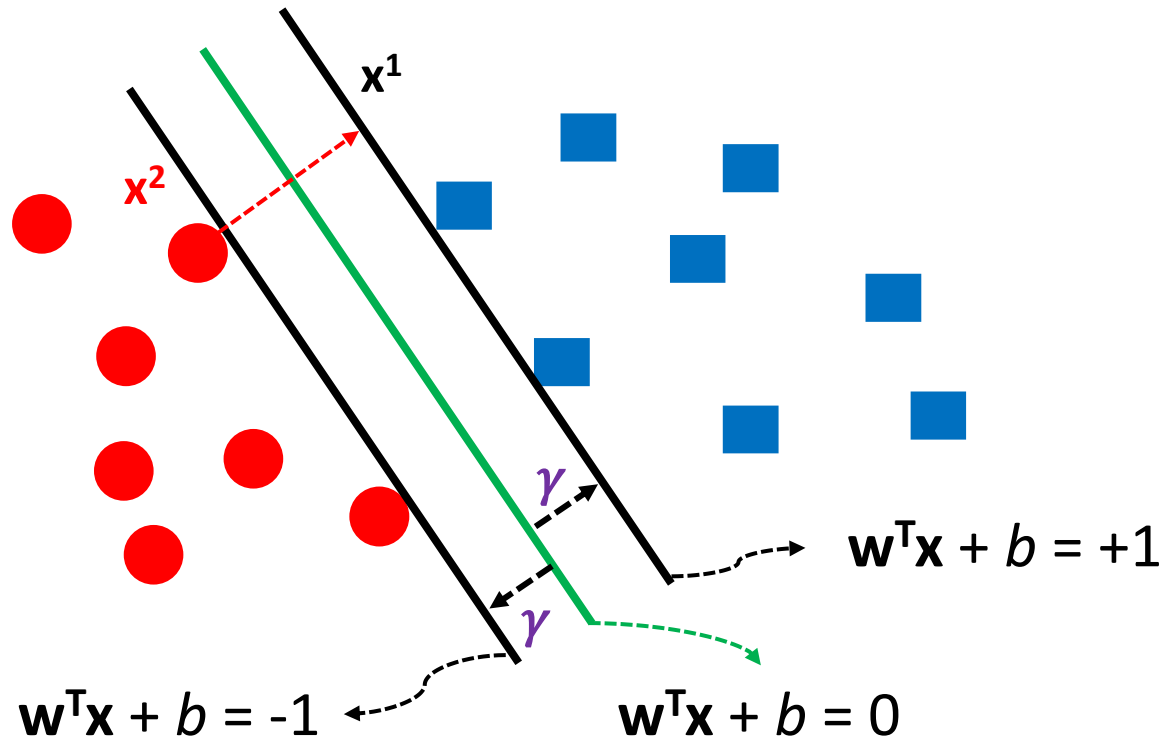
# Computing the margin width

- Consider one of the support vectors, (say,  $x^2$  on the black line) and let  $x^1$  be the projection of  $x^2$  to the upper hyperplane



- $x^1 = x^2 + \lambda w$  for some value of  $\lambda$ 
  - The line from  $x^2$  to  $x^1$  is perpendicular to the planes
  - So to get from  $x^2$  to  $x^1$  travel some distance in direction  $w$

# Computing the margin width



**What we know:**

$$w^T x^1 + b = +1$$

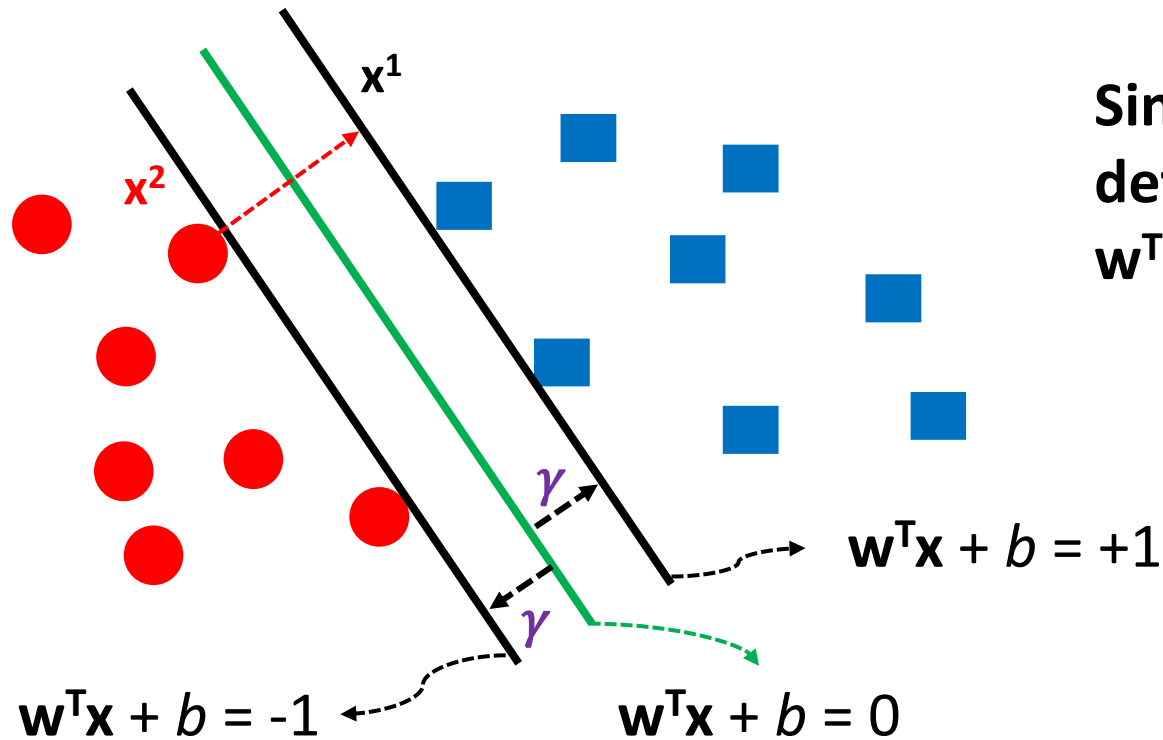
$$w^T x^2 + b = -1$$

$$x^1 = x^2 + \lambda w$$

$$\|x^1 - x^2\| = 2\gamma$$

It's now easy to get  $\gamma$   
in terms of  $w$  and  $b$

# Computing the margin width



What we know:

$$w^T x^1 + b = +1$$

$$w^T x^2 + b = -1$$

$$x^1 = x^2 + \lambda w$$

$$|x^1 - x^2| = 2\gamma$$

Since  $x^1$  is on the hyperplane defined by  $w \cdot x + b = +1$ , we know that  $w^T x^1 + b = 1$ . If we substitute for  $x^1$ :

$$w^T x^1 + b = 1$$

$\Rightarrow$

$$w^T (x^2 + \lambda w) + b = 1$$

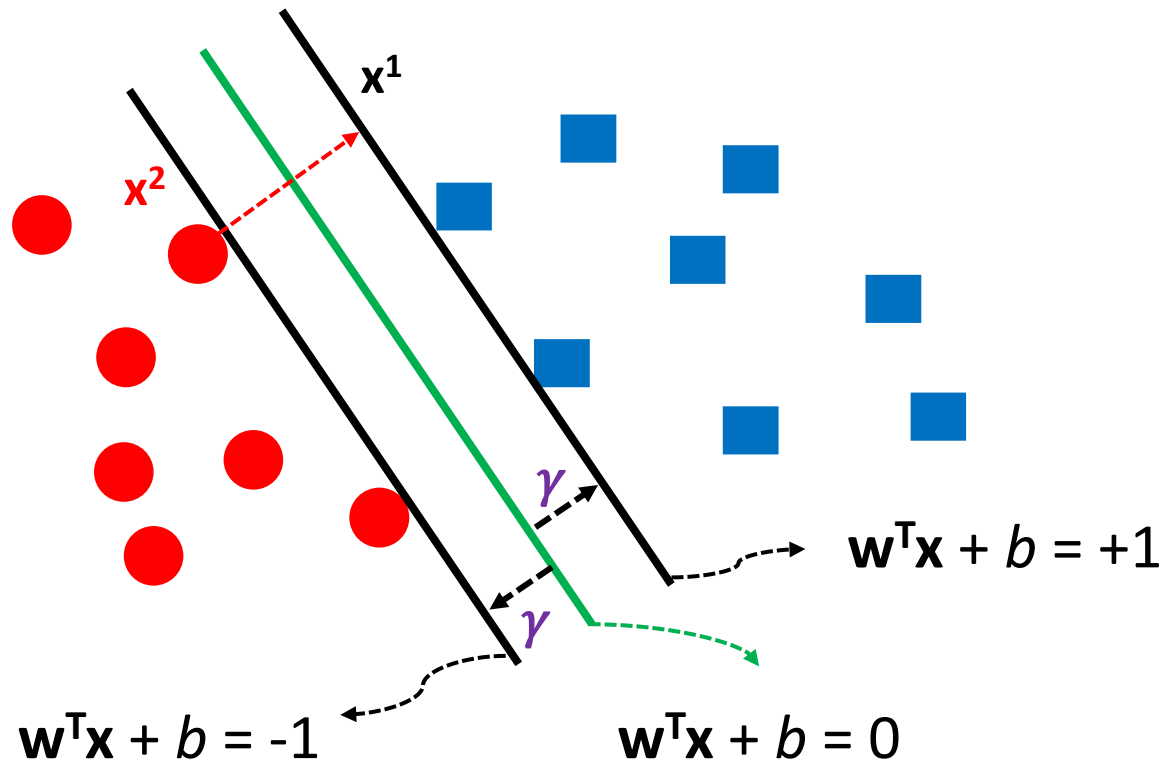
$\Rightarrow$

$$-1 + \lambda w^T w = 1$$

$\Rightarrow$

$$\lambda = 2 / w^T w$$

# Computing the margin width



What we know:

$$w^T x^1 + b = +1$$

$$w^T x^2 + b = -1$$

$$x^1 = x^2 + \lambda w$$

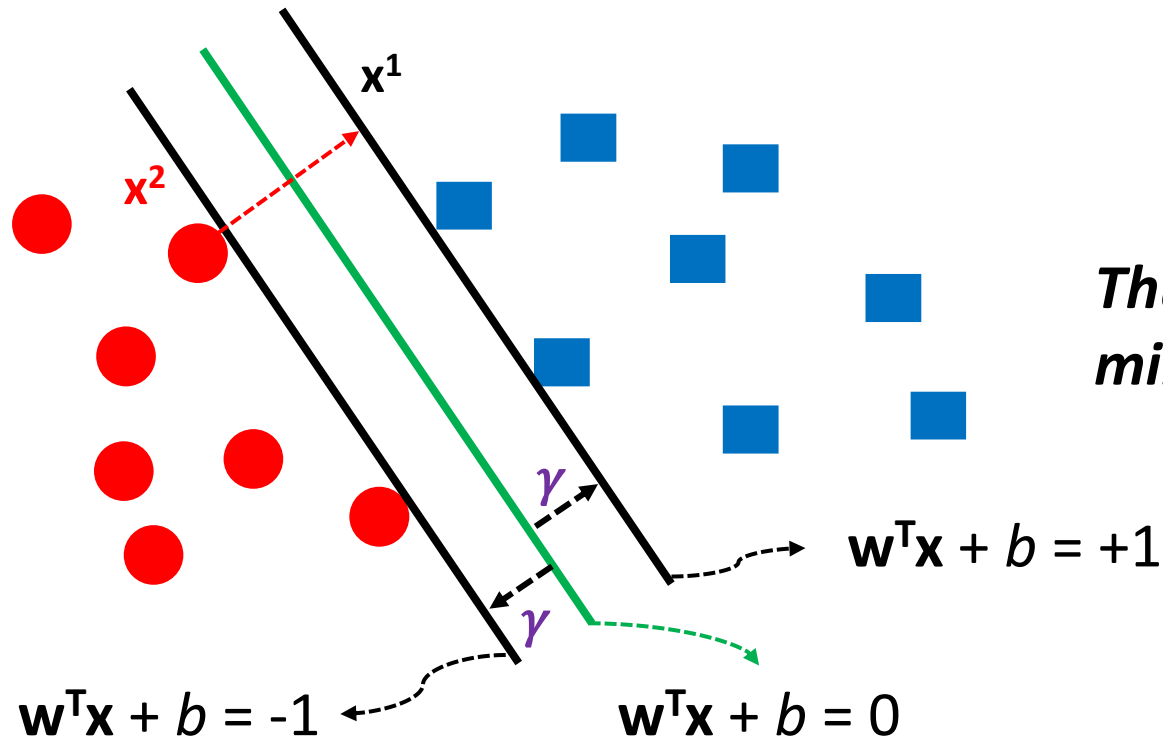
$$|x^1 - x^2| = 2\gamma$$

$$\begin{aligned} 2\gamma &= \|x^1 - x^2\| \\ &= \|\lambda w\| \\ &= \lambda \|w\| \end{aligned}$$

Because  $\lambda = 2 / w^T w$   
 $\Rightarrow$

$$2\gamma = \frac{2\|w\|}{w^T w} = \frac{2}{\|w\|}$$

# Computing the margin width



$$\gamma = 1 / \|w\|$$

*Thus, maximizing  $\gamma$  is the same as minimizing  $\|w\|$*

# The Objective of SVM

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $\mathbf{w}$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} \mathbf{w}^T \mathbf{x}^i + b \geq 1 & \text{if } y^i = +1 \\ \mathbf{w}^T \mathbf{x}^i + b \leq -1 & \text{if } y^i = -1 \end{cases}$$

**But, why would this constraint serve in materializing *large margin* classification?**

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $\mathbf{w}$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} \mathbf{w}^T \mathbf{x}^i + b \geq 1 & \text{if } y^i = +1 \\ \mathbf{w}^T \mathbf{x}^i + b \leq -1 & \text{if } y^i = -1 \end{cases}$$

**For illustrative purposes, let us assume only two features (i.e.,  $\mathbf{x}^i = [x_1^i, x_2^i]$  and  $\mathbf{w} = [w_1, w_2]$ ) and  $b = 0$**

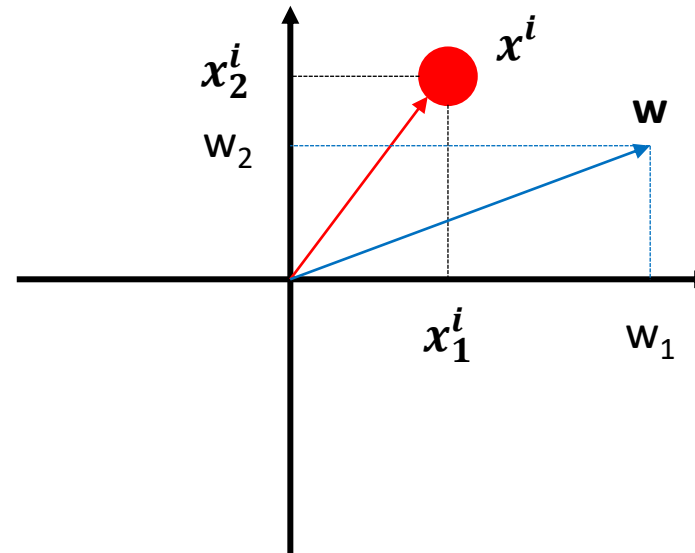


# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} w^T x^i + b \geq 1 & \text{if } y^i = +1 \\ w^T x^i + b \leq -1 & \text{if } y^i = -1 \end{cases}$$

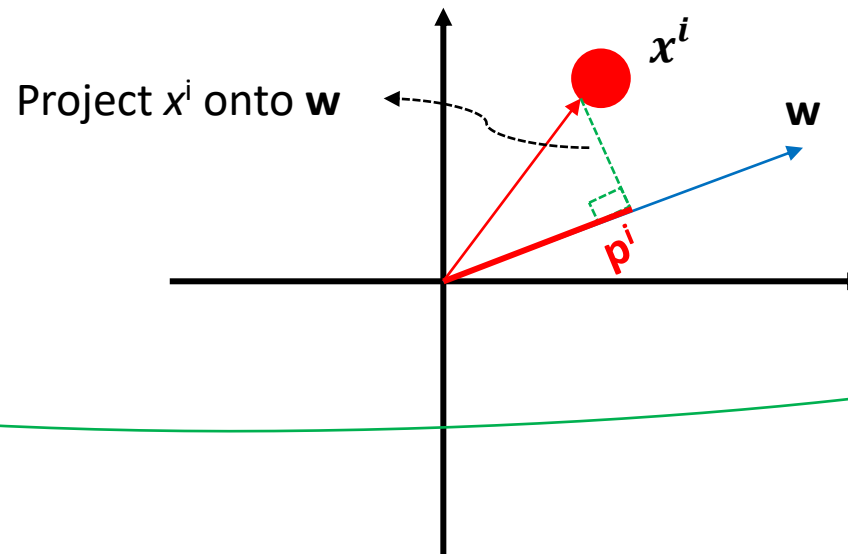


# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} w^T x^i + b \geq 1 & \text{if } y^i = +1 \\ w^T x^i + b \leq -1 & \text{if } y^i = -1 \end{cases}$$



What is the inner product of  $w$  and  $x^i$  (i.e.,  $w^T x^i$ )?

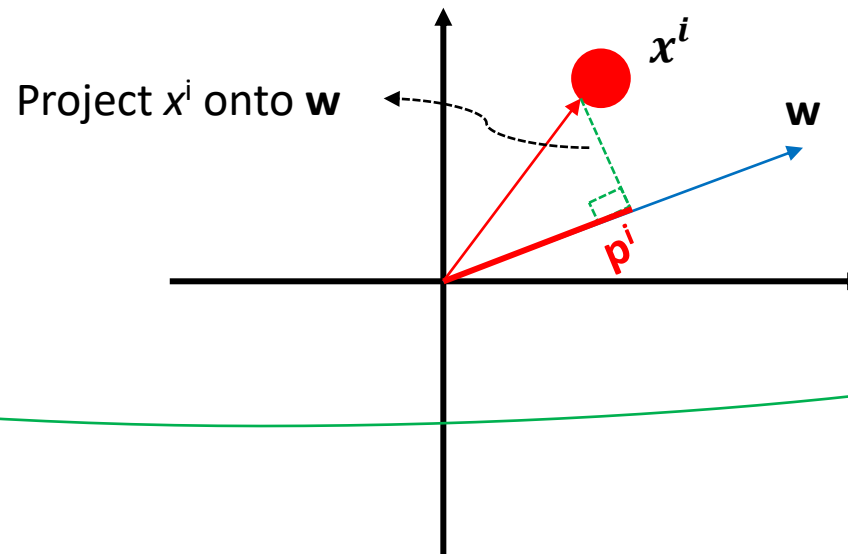
$$\begin{aligned} & p^i \cdot \|w\| \\ &= w_1 \cdot x_1^i + w_2 \cdot x_2^i \end{aligned}$$

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i \cdot \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i \cdot \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



What is the inner product of  $w$  and  $x^i$  (i.e.,  $w^T x^i$ )?

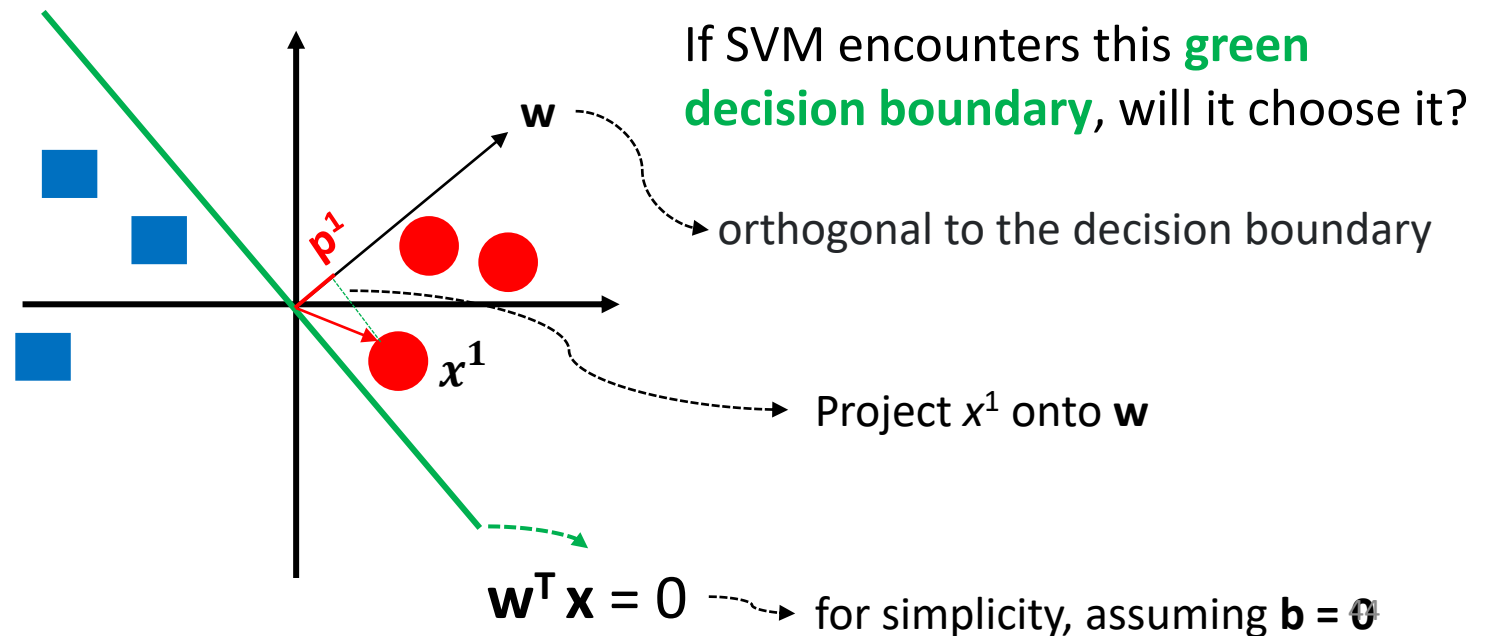
$$\begin{aligned} & p^i \cdot \|w\| \\ &= w_1 \cdot x_1^i + w_2 \cdot x_2^i \end{aligned}$$

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i. \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i. \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$

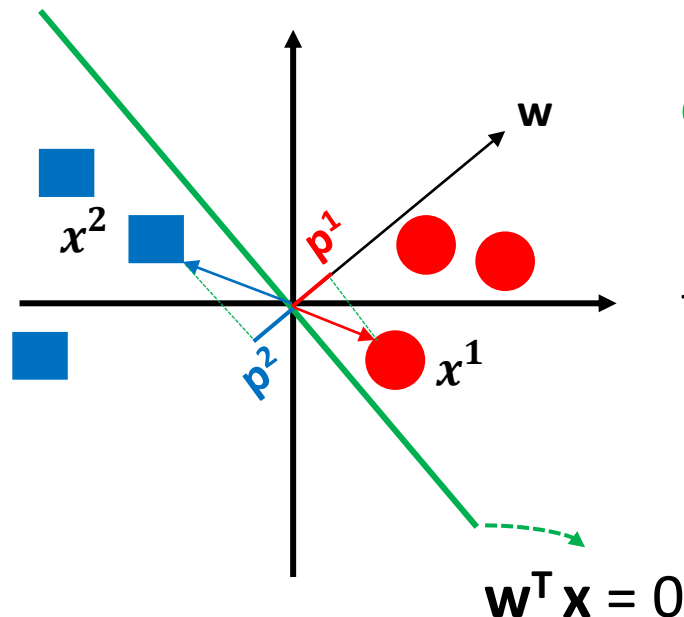


# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i \cdot \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i \cdot \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

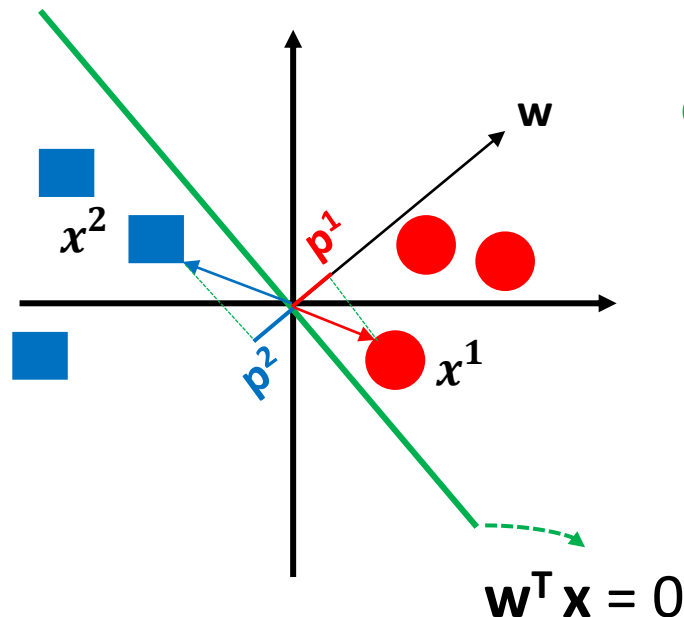
$p^1$  is positive and very small, hence, for  $p^1 \cdot \|w\|$  to be  $\geq 1$ ,  $\|w\|$  has to be very large!

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i. \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i. \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

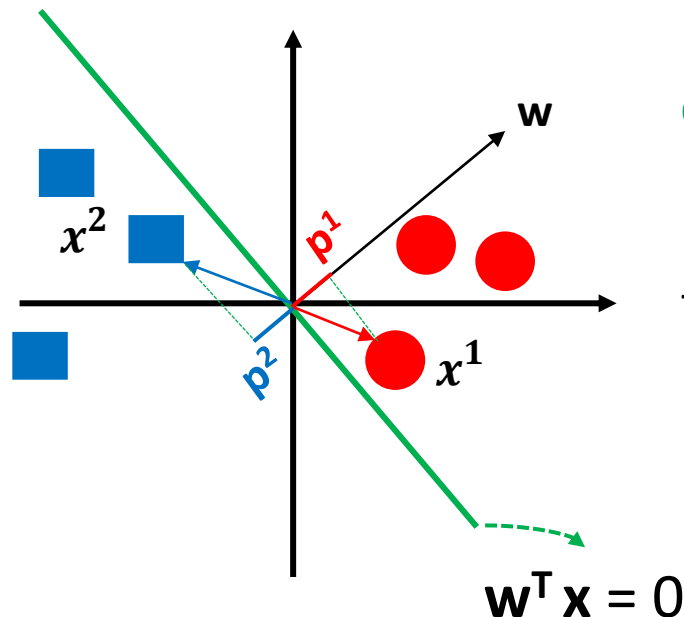
But, the optimization objective is to minimize  $\|w\|$ , hence, SVM will not prefer this decision boundary

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i \cdot \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i \cdot \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

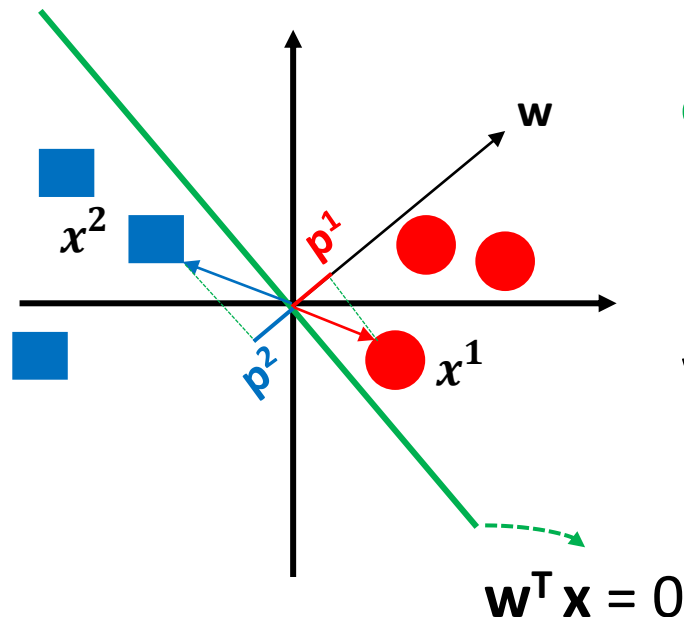
$p^2$  is negative and very small, hence, for  $p^2 \cdot \|w\|$  to be  $\leq -1$ ,  $\|w\|$  has to be very large!

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i. \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i. \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

But, the optimization objective is to minimize  $\|w\|$ , thus, again, SVM will not prefer this decision boundary

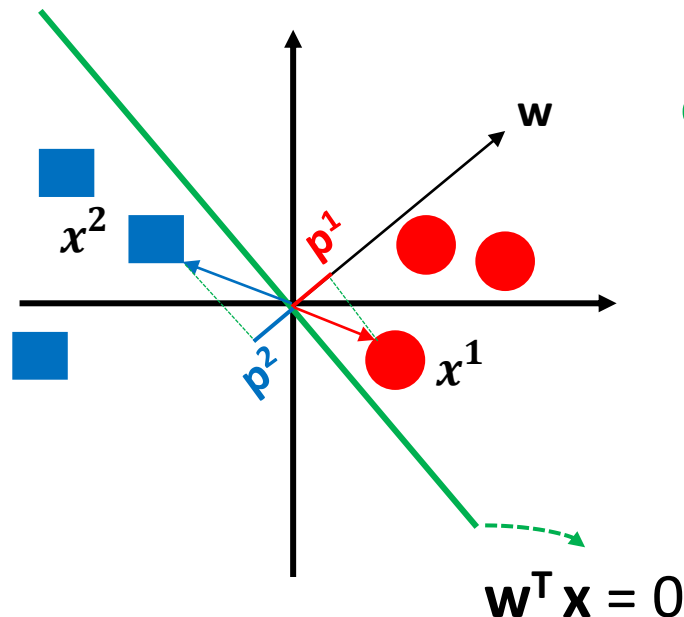


# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i. \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i. \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

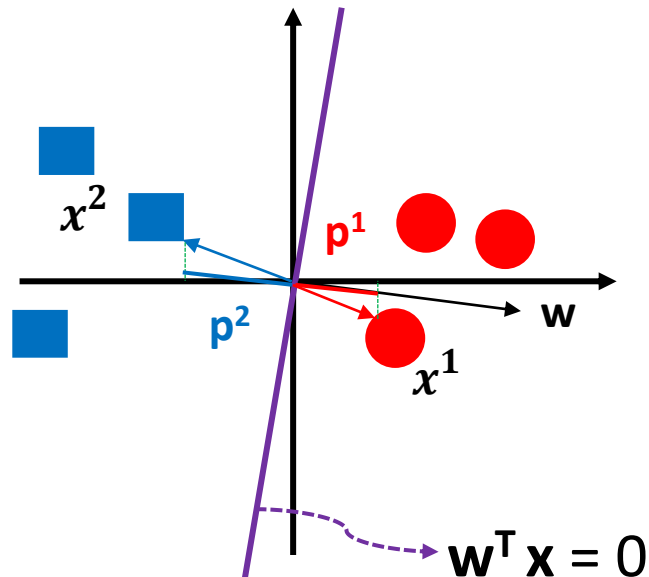
**NO**

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i \cdot \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i \cdot \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

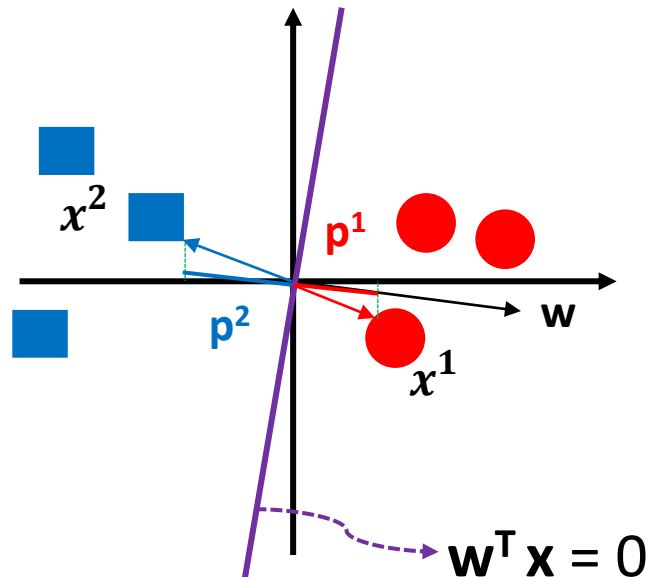
$p^1$  is positive and bigger now, hence, for  $p^1 \cdot \|w\|$  to be  $\geq 1$ ,  $\|w\|$  can be smaller, aligning better with the optimization objective

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i \cdot \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i \cdot \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

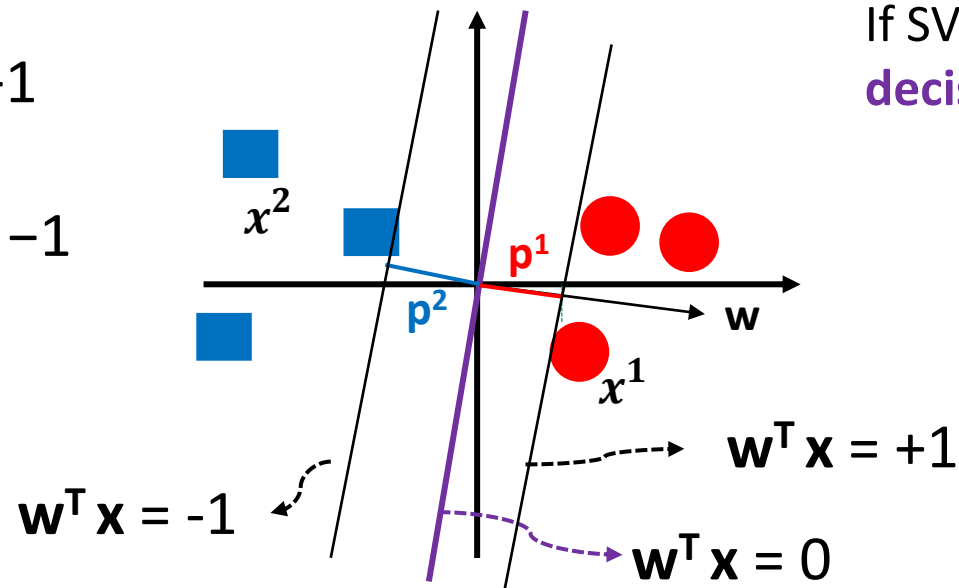
$p^2$  is negative and bigger now, hence, for  $p^2 \cdot \|w\|$  to be  $\leq -1$ ,  $\|w\|$  can be smaller, aligning better with the optimization objective

# The Objective of SVM - Illustration

- More formally, the goal of SVM can be stated as follows:

Given a training set  $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$ , minimize  $\|w\|$  (by varying  $w$  and  $b$ ) subject to the constraint that for all  $i = 1, 2, \dots, n$ ,

$$\begin{cases} p^i. \|w\| \geq 1 & \text{if } y^i = +1 \\ p^i. \|w\| \leq -1 & \text{if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

YES

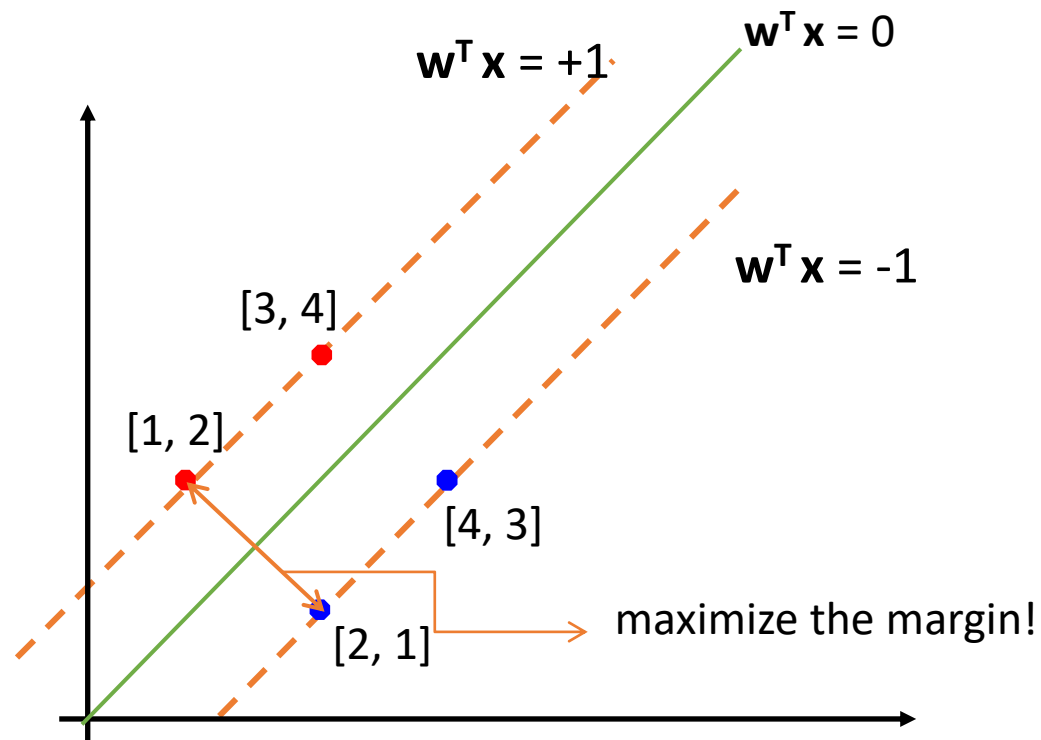
# Example

- Consider the following training examples, assuming  $\mathbf{w} = [w_1, w_2]$
- How to solve for  $\mathbf{w}$  and  $b$ ?
- Our goal is to maximize the margin subject to the constraints  $y^i(\mathbf{w}^T \mathbf{x}^i + b) \geq 1$ , which can be derived from the training examples

$\mathbf{x}$	$y$	Constraints
[1, 2]	+1	$u + 2v + b \geq 1$
[2, 1]	-1	$2u + v + b \leq -1$
[3, 4]	+1	$3u + 4v + b \geq 1$
[4, 3]	-1	$4u + 3v + b \leq -1$

# Example

- Consider the following training examples, assuming  $\mathbf{w} = [w_1, w_2]$
- it is easy to find that  $b = 0$  and  $\mathbf{w} = [-1, +1]$  using geometric interpretation.



# Decision Tree for Malware Detection

- Malware (malicious software): any intrusive software developed by cybercriminals to steal data and damage or destroy computers and computer systems
- Examples of common malware include
  - virus
  - trojan
  - botnet
  - downloader
  - rootkit
  - ransomware
  - APT
  - spyware
  - zero day

# Common Types of Malware

- **Virus**

- A self-replicating program that reproduces its code by attaching copies into other executable codes, designed to disrupt a system's ability to operate

- **Trojan**

- Executable that appears as legitimate and harmless, but once it is launched, it executes malicious instructions in the background

- **Botnet**

- Malware that has the goal of compromising as many possible hosts of a network, in order to put their computational capacity at the service of the attacker

- **Downloader**

- Malware that downloads malicious libraries or portions of code from the network and executes them on victim hosts
- Malicious code that exists only to download other malicious code



# Common Types of Malware

- **Rootkit**

- Malware that compromises the hosts at the operating system level and often comes in the form of device drivers, making the various countermeasures (e.g. antiviruses installed on the endpoints) ineffective

- **Ransomware**

- Malware that proceeds to encrypt files stored inside the host machines, asking for a ransom from the victim to obtain the decryption key which is used for recovering the original files

- **APT**

- APT (Advanced Persistent Threat) is a form of tailored attack that exploits specific vulnerabilities on the victimized hosts

# Common Types of Malware

- **Spyware**

- Spyware is malicious software that runs secretly on a computer and reports back to a remote user. Rather than simply disrupting a device's operations, often used to steal financial or personal information
- A specific type of spyware is a keylogger, which records keystrokes to reveal passwords and personal information

- **Zero day**

- Malware that exploits vulnerabilities not yet disclosed to the community of researchers and analysts, whose characteristics and impacts in terms of security are not yet known, and therefore go undetected by antivirus software

# Malware Detection Strategies

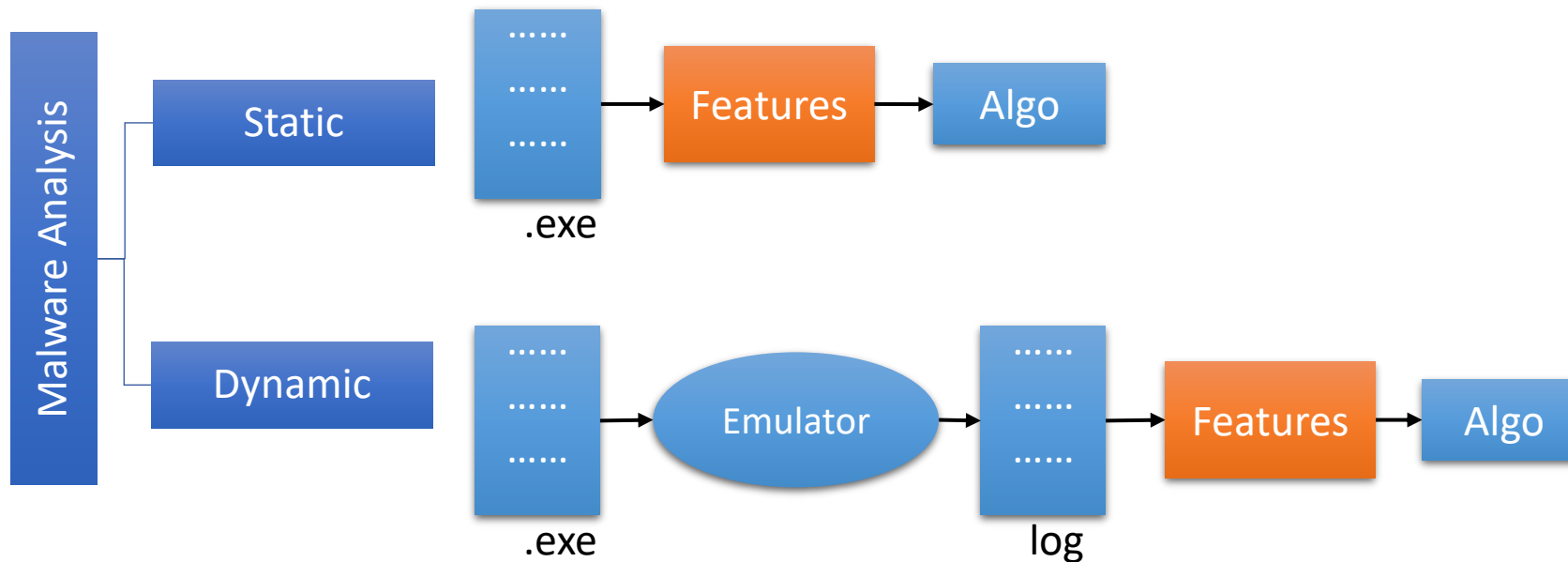
- Detection activities that can be easily automated
  - File hash calculation: To identify known threats already present in the knowledge base
  - System monitoring: To identify anomalous behavior of both the hardware and the operating system
    - such as an unusual increase in CPU cycles
    - a particularly heavy disk writing activity
    - changes to the registry keys
    - the creation of new and unsolicited processes in the system ...
  - Network monitoring: To identify anomalous connections established by host machines to remote destinations

# Malware Analysis Techniques

- Two fundamental approaches to malware analysis
  - Static and dynamic
- Static (code) analysis: analyzing the code or structure of a program to determine its function without running it
- Dynamic (behavioral) analysis: running the malware and observing its behavior on the system
  - Before running the malware safely, a sandbox environment must be set up that will allow to study the running malware without damaging to the system or network.
- Usually fulfilled by security specialists

# Steps for building a ML-based Malware detector

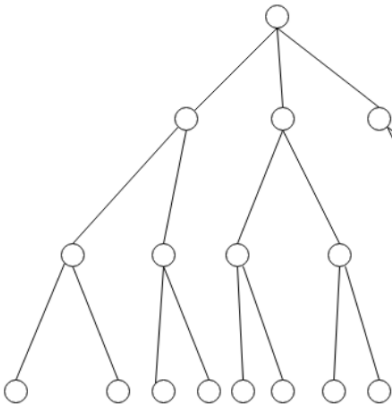
- After analyzing the malware, we can extract the features of the binary files (whether legitimate or suspect), and store them in a dataset of artifacts with which to train our algorithms

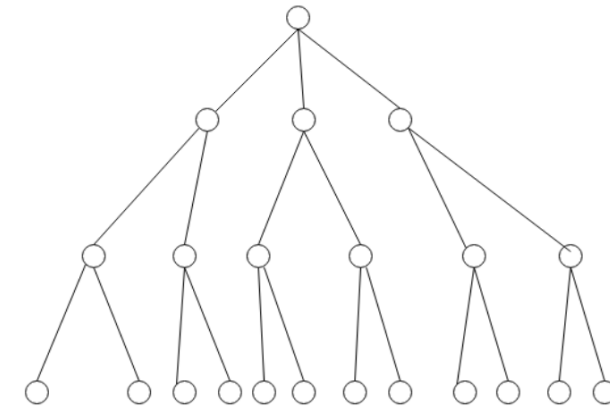


# Steps for building a ML-based Malware detector

- Collect
  - Examples of malware and benignware
  - Used to train the model
- Extract
  - Features from each training example to represent the data
  - This step also includes research to design good features that will help the machine learning model make accurate inferences – domain experts
- Train and Test
  - Machine learning model to detect malware using the features
  - Our focus: malicious or non-malicious

# Decision tree Malware Detectors

- Now, we will deal with the classification problems solved by **decision trees** in the context of detecting malware threats
    - supervised learning method
    - classify the data into malicious and non-malicious categories
  - Model the learning process based on a sequence of **if-then-else** decisions
    - => a tree structure
  - Non-linear classifier
    - decision boundaries are not reducible to straight lines or hyperplanes in the space
- 



# Next

- Decision Tree Overview
  - ID3 is a simple decision tree learning algorithm developed by Ross Quinlan
  - The basic idea: construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node
  - To select the attribute that is most useful for classifying a given sets, we introduce a metric---information gain
- Information Gain
- Build a Decision Tree



# Decision Tree Example

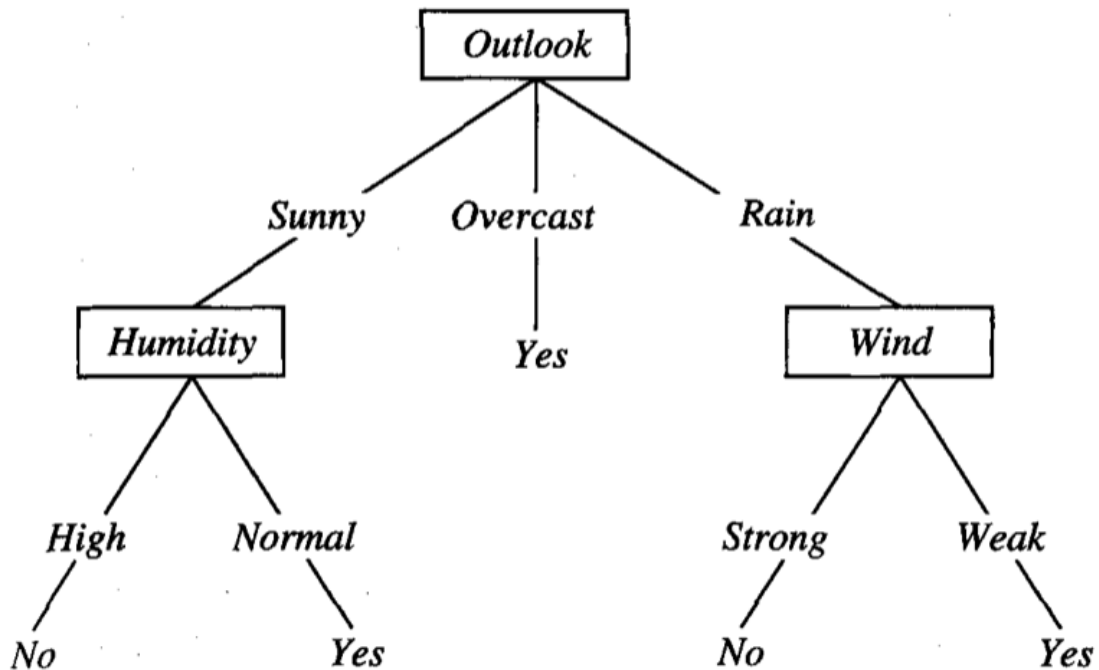
- Here is an example for decision tree: play tennis.
- Outlook, temperature, humidity and wind are the attributes:  $X$
- PlayTennis is the label:  $Y$
- **Decision Tree**
  - based on the attributes  $X$
  - make decisions on the label  $Y$

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

For ease of understanding, we will follow an example from Quinlan's ID3

# Decision Tree Example

- A possible decision tree for the data:

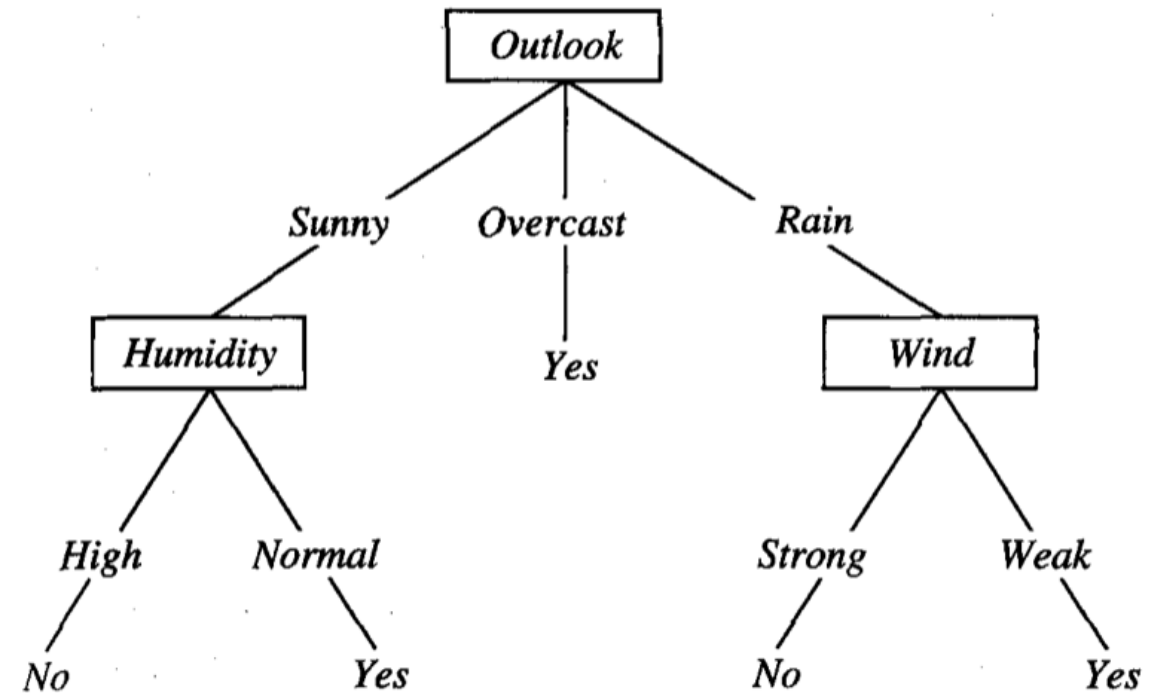


- Each internal node:
  - tests one attribute  $X_i$  (such as outlook, humidity, wind...)
- Each branch from a node:
  - selects one value for  $X_i$
- Each leaf node:
  - prediction of  $Y$  (yes or no)

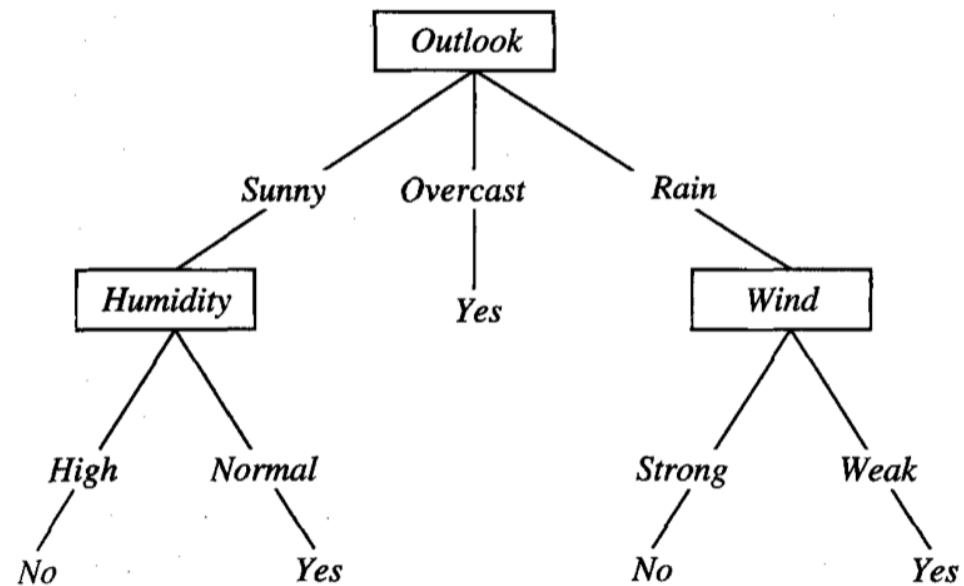
# Decision Tree Example

- What is the prediction for:
- outlook=sunny, temperature=mild, humidity=high, wind=weak ?
- Example gets sorted down the leftmost branch of this decision tree and classified as a negative instance (i.e., the tree predicts that PlayTennis is no).

A possible decision tree for the data:



- Decision trees classify examples by sorting top down.
- An example is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example.
- This process is then repeated for the subtree rooted at the new node



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

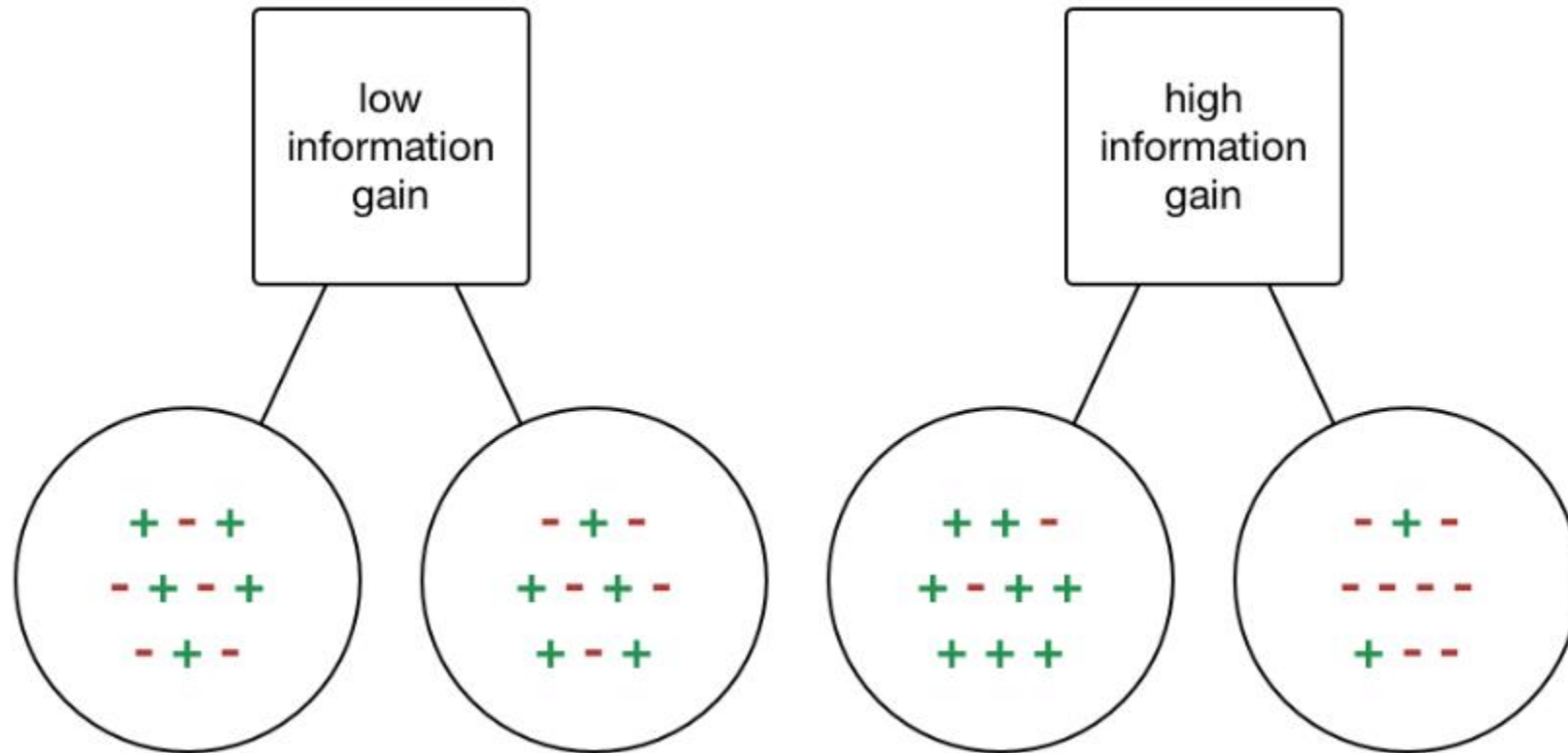
# Information Gain

- To build a good decision tree
  - We need to find good splits
  - i.e.,  $X_1$  or  $X_2$
  - select which attribute to test at each node in the tree
- Increase certainty about classification (yes or no) after split
- To measure classification certainty, we introduce information gain

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

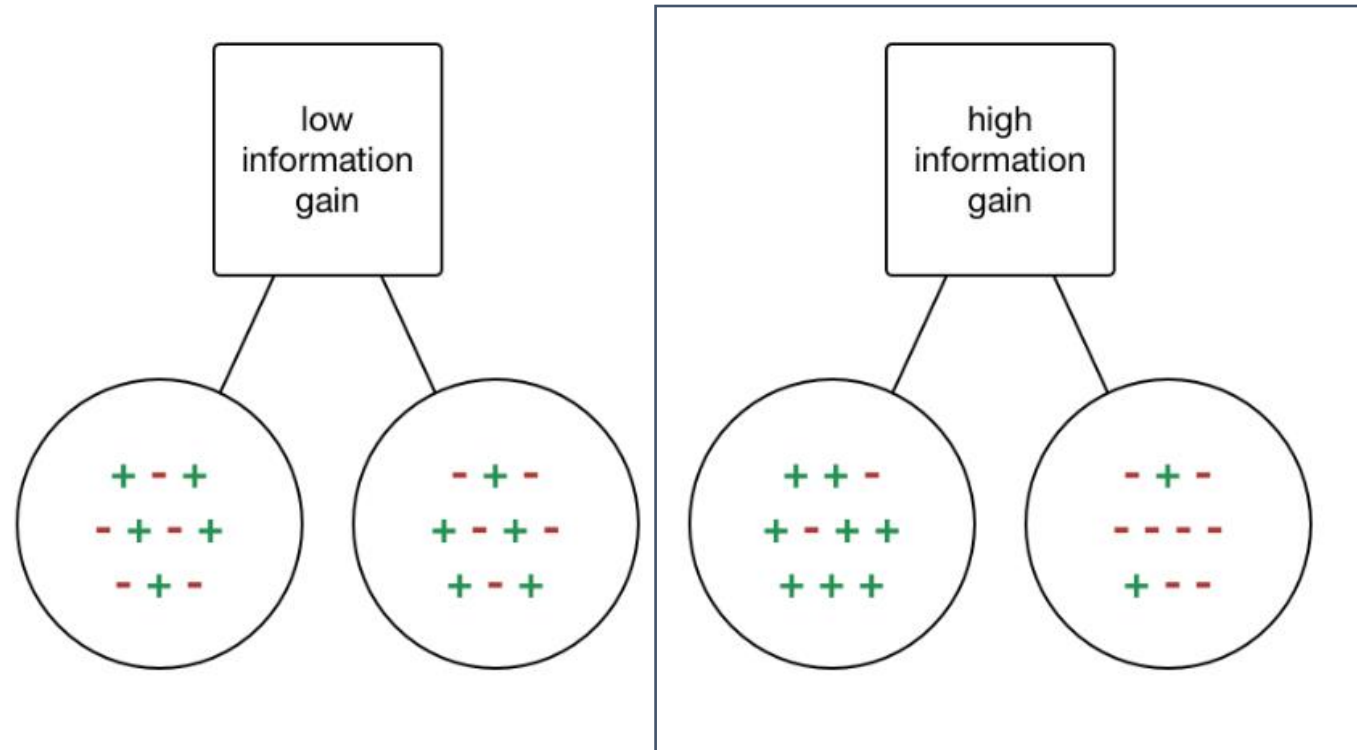
# Information Gain

- Which split is more informative? Why?



# Information Gain

- The split on the right is more informative, since after the split the distribution of data becomes more **discriminative**.
  - i.e., more similar labels in one partition.
- We use **entropy** and **information gain** to evaluate the uncertainty and goodness of a split.



# Entropy

- Entropy can be used as a measure of uncertainty in information theory.
- Entropy  $H(Y)$  of a random variable  $Y$  with  $n$  different possible values:

$$H(Y) = - \sum_{i=1}^n P(y_i) \log_2 P(y_i)$$

where  $P(y_i)$  is the proportion of values in  $Y$  that belong to the  $i$ -th class

- Entropy is 0 if all values of  $Y$  belong to the same class.



# Entropy example

- If Y is a **binary** variable as shown in the table.
- Y has **two** values: T (True) and F (False).
- In the following equation:

$$H(Y) = - \sum_{i=1}^n P(y_i) \log_2 P(y_i)$$

- We have  **$n = 2$** .

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Entropy example

- There are
  - 5 instances with  $Y = T$
  - and 3 instances with  $Y = F$
- Therefore,  $P(Y = T) = 5/8$  and  $P(Y = F) = 3/8$
- We can calculate its entropy as:

$$H(Y) = -\frac{5}{8} \log_2 \frac{5}{8} - \frac{3}{8} \log_2 \frac{3}{8} \approx 0.954$$

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Entropy after split

- Entropy  $H(Y)$  of a random variable  $Y$  with  $n$  different possible values:

$$H(Y) = - \sum_{i=1}^n P(y_i) \log_2 P(y_i)$$

- Conditional entropy  $H(Y|X = x_j)$  of  $Y$  given  $X = x_j$ :

$$H(Y|X = x_j) = - \sum_{i=1}^n P(y_i|X = x_j) \log_2 P(y_i|X = x_j)$$

- The above equation: split the data according to the value of  $X$ , the entropy of random variable  $Y$  with  $X = x_j$ .

- Expected entropy  $H(Y|X)$  after split:

$$H(Y|X) = \sum_{j=1}^k P(x_j) H(Y|X = x_j)$$

- The above equation:  $X$  has  $k$  possible values,  $P(x_j)$  is the probability that  $X = x_j$ .

# Entropy after split example

- Suppose we split the data according to the value of  $X_1$ .
- $X_1$  has **two** possible values: T and F.
- We can compute the conditional entropy for  $X_1 = T$  and  $X_1 = F$ .

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Entropy after split example

- Find the conditional entropy for  $X_1 = T$
- After the split, when  $X_1 = T$ , all corresponding  $Y = T$ .
- Therefore:
  - $P(Y = T|X_1 = T) = 1$  and  $P(Y = F|X_1 = T) = 0$
- We have the following entropy when  $X_1 = T$ :

$$\begin{aligned} H(Y|X_1 = T) &= -\sum_{i=1}^n P(y_i|X_1 = T) \log_2 P(y_i|X_1 = T) \\ &= -1 * \log_2(1) - 0 * \log_2(0) = 0 \end{aligned}$$

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Entropy after split example

- Find the conditional entropy for  $X_1 = F$
- After the split, when  $X_1 = F$ , **one** corresponding  $Y = T$ , and **three** corresponding  $Y = F$ .
- Therefore,
  - $P(Y = T|X_1 = F) = 1/4$  and  $P(Y = F|X_1 = F) = 3/4$
- We have the following entropy when  $X_1 = F$ :

$$\begin{aligned} H(Y|X_1 = F) &= -\sum_{i=1}^n P(y_i|X_1 = F) \log_2 P(y_i|X_1 = F) \\ &= -\frac{1}{4} * \log_2 \frac{1}{4} - \frac{3}{4} * \log_2 \frac{3}{4} = 0.811 \end{aligned}$$

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Entropy after split example

- After the above computations, we then find the expected entropy.
- There are **four** instances with  $X_1 = T$ , and **four** instances with  $X_1 = F$
- Therefore, we have  $P(X_1 = T) = P(X_1 = F) = 1/2$
- We have the following expected entropy for  $X_1$  :

$$H(Y|X_1) = \frac{1}{2} * 0 + \frac{1}{2} * 0.811 = 0.406$$

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

# Information Gain

- Information gain  $I(Y, X)$  is defined as the expected reduction in entropy of target variable  $Y$  after split over variable  $X$

$$I(Y, X) = H(Y) - H(Y|X)$$



# Information Gain example

- Recall that we have the entropy of  $Y$ ,  $H(Y) = 0.954$ , and expected entropy  $H(Y|X_1) = 0.406$ .
- We can easily compute the information gain for data in the table after split over variable  $X_1$

$$I(Y, X_1) = H(Y) - H(Y|X_1) = 0.954 - 0.406 = 0.548$$

Exercise: What is the information gain  $I(Y, X_2)$ ?

$X_1$	$X_2$	$Y$
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Next - Build a decision tree

# What is a good decision tree

- When we learn a decision tree, we want to determine which attribute in a given dataset of training examples is the most useful for discriminating between the classes to be learned
- We use **information gain** which tells us how important a given attribute is to decide the order of attributes for splitting in a decision tree.
- We introduce **ID3** (Iterative Dichotomiser 3) invented by Ross Quinlan for learning decision tree.

# Overview of ID3

- ID3 employs a **top-down** manner to construct a decision tree.
- ID3 builds the decision tree from the root node, and at each step, it makes the decision of node to be split based on the information gain.

# ID3 Algorithm

- 1. Base entropy is computed based on the distribution of label  $Y$ .
- 2. Each attribute is evaluated to get the information gain which determines how well it classifies the training data.
- 3. The best attribute is selected and used as the root node of the tree.
- 4. A descendant of the root node is created for each possible value of this attribute.
- 5. The previous processes are repeated on the remaining attributes to find the best attribute based on the information gain as the descendant node.
- 6. Finally, we form a decision tree.

# ID3 example

- We use the Play Tennis data as an example to build a decision tree using ID3.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# ID3 example

- In the given data, there are 14 days in total, including 9 positive and 5 negative labels.
- Therefore, the base entropy is:

$$H(Y) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14}$$
$$= 0.940$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# ID3 example

- We show an example information gain computation for the attribute **Outlook**.
- Outlook has three values: Sunny, Overcast and Rain.

$$P(sunny) = \frac{5}{14}$$
$$P(overcast) = \frac{4}{14}$$
$$P(rain) = \frac{5}{14}$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



# ID3 example

- The conditional entropy given **Outlook=sunny** can be computed as:

$$H(Y|sunny) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$

Because when outlook=sunny, there are 2 positive and 3 negative labels.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# ID3 example

- The conditional entropy given **Outlook=overcast** can be computed as:

$$H(Y|overcast) = -\frac{0}{4} \log_2 \frac{0}{4} - \frac{4}{4} \log_2 \frac{4}{4} = 0$$

Because when outlook=overcast, there are 4 positive and 0 negative labels.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# ID3 example

- The conditional entropy given **Outlook=rain** can be computed as:

$$H(Y|rain) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$

Because when outlook=overcast, there are 3 positive and 2 negative labels.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# ID3 example

- The expected entropy of attribute **Outlook** is:

$$\begin{aligned} H(Y|outlook) &= P(sunny) * H(Y|sunny) + P(overcast) * \\ &\quad H(Y|overcast) + P(rain) * H(Y|rain) \\ &= \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.694 \end{aligned}$$

$$(\text{recall that: } P(sunny) = \frac{5}{14}, P(overcast) = \frac{4}{14}, P(rain) = \frac{5}{14})$$

# ID3 example

- The information gain after split over **outlook** is:

$$\begin{aligned} I(Y, outlook) &= H(Y) - H(Y|outlook) \\ &= 0.940 - 0.694 \\ &= 0.246 \end{aligned}$$

# ID3 example

- Similarly, we can compute the information gain for other attributes

Temperature, Wind and Humidity:

$$I(Y, temperature) = 0.029$$

$$I(Y, wind) = 0.048$$

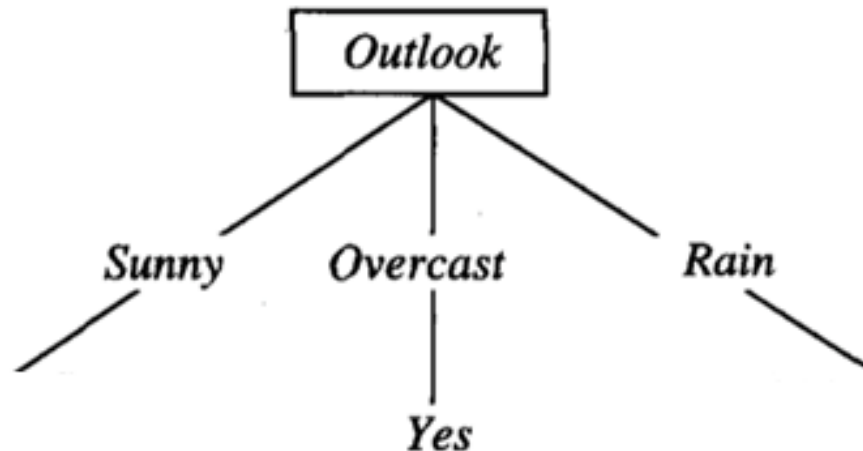
$$I(Y, humidity) = 0.152$$

- And from the previous computation:

$$I(Y, outlook) = 0.246$$

# ID3 example

- Among all attributes, **outlook** has **the most** information gain.
- We make outlook the root node of the decision tree and its values as the branches.
- Because when outlook=overcast, the labels are all positive, we have a pure leaf node in this tree.



# ID3 example

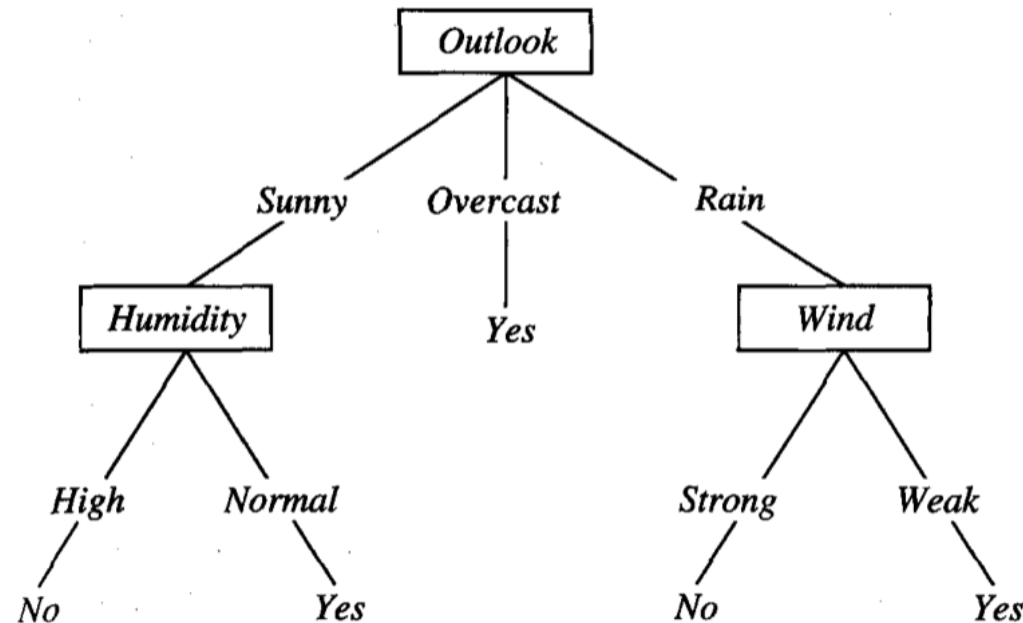
- For each branch of the root node, we repeat the computation of the information gain, and find the best node as the child node.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



# ID3 example

- For sunny branch, we have humidity as the child node.
- For rain branch, we have wind as the child node.
- For each branch, we can find a label as the leaf node. Therefore, we build our decision tree:



# References

- *Support Vector Machines, Andrew W. Moore, CMU School of Computer Science*
- *Introduction to Information Retrieval: support vector machines*