

# Deep Learning & Neural Networks

**Dr. Shahira Ahmed**

**DS710**

**Lecture 4**





# Deep Learning & Neural Networks DS710

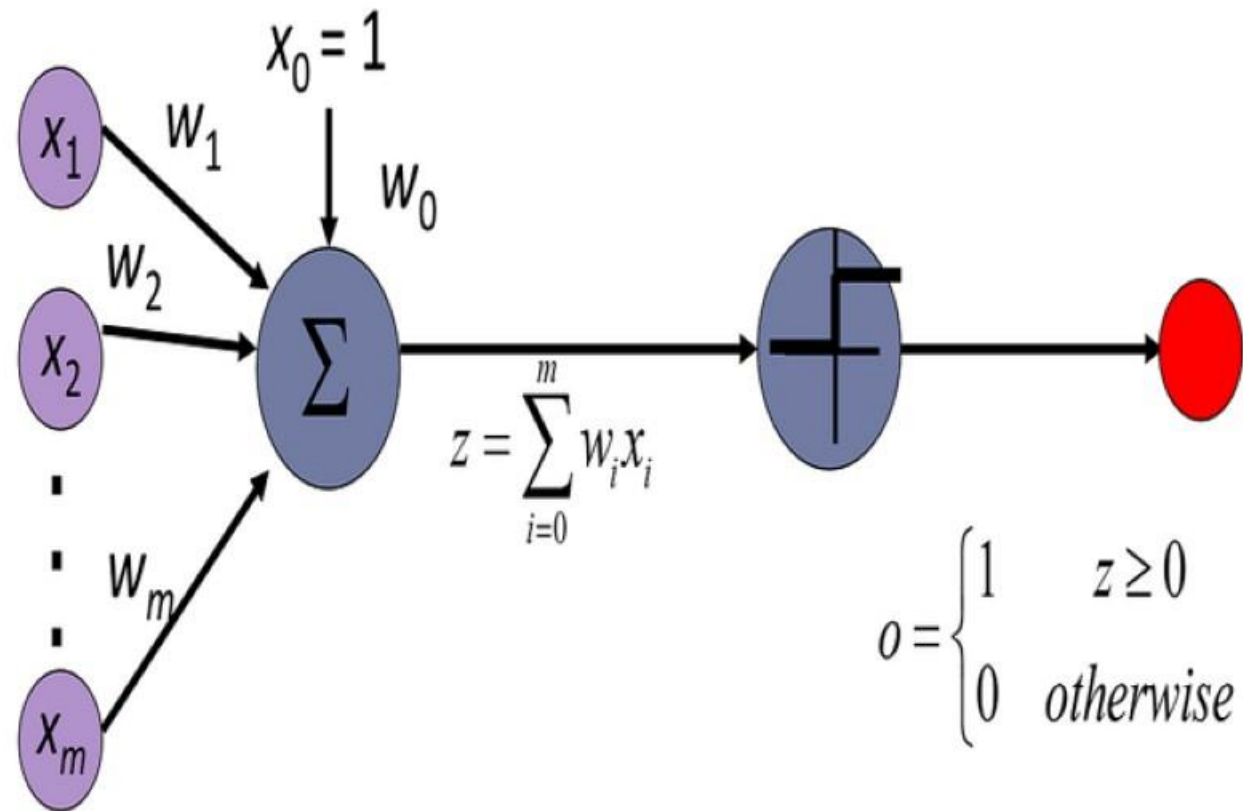
**Lecture 3**  
**Training the Network-Part(1)**

**Dr. Shahira Ahmed**

[www.midocean.ae](http://www.midocean.ae)

- Single Layer Perceptron : Review
- Perceptron Learning algorithm
- Perceptron Learning algorithm: Example

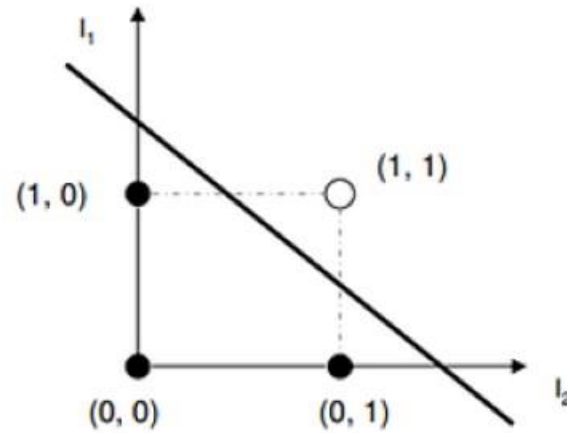
# Single Layer Perceptron : Review



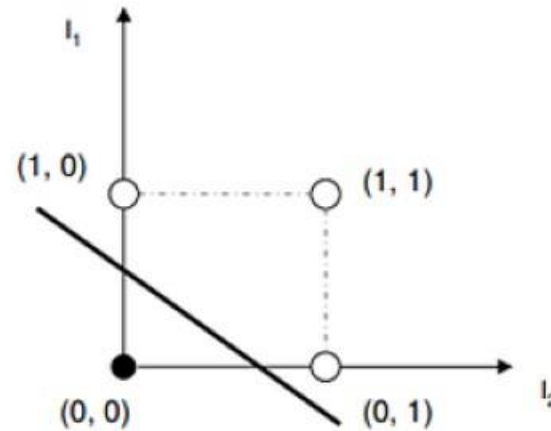
- A perceptron model's output can only be a binary number i.e. "0" or "1" because of activation function. → Classification
- The perceptron model can only be used to categorize the input vectors' linearly separable sets. If input vectors are non-linear, they can't be properly classified.

# Limitation of Perceptron

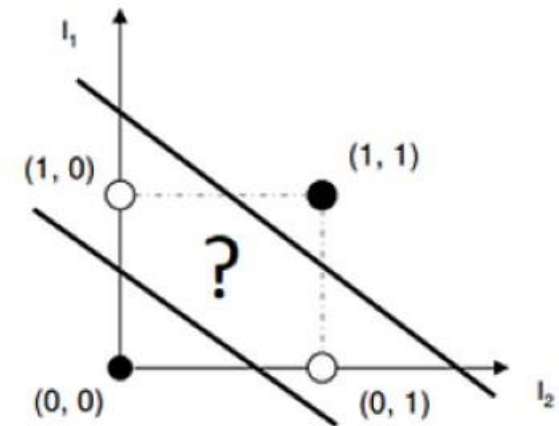
| AND   |       |     |
|-------|-------|-----|
| $I_1$ | $I_2$ | out |
| 0     | 0     | 0   |
| 0     | 1     | 0   |
| 1     | 0     | 0   |
| 1     | 1     | 1   |



| OR    |       |     |
|-------|-------|-----|
| $I_1$ | $I_2$ | out |
| 0     | 0     | 0   |
| 0     | 1     | 1   |
| 1     | 0     | 1   |
| 1     | 1     | 1   |

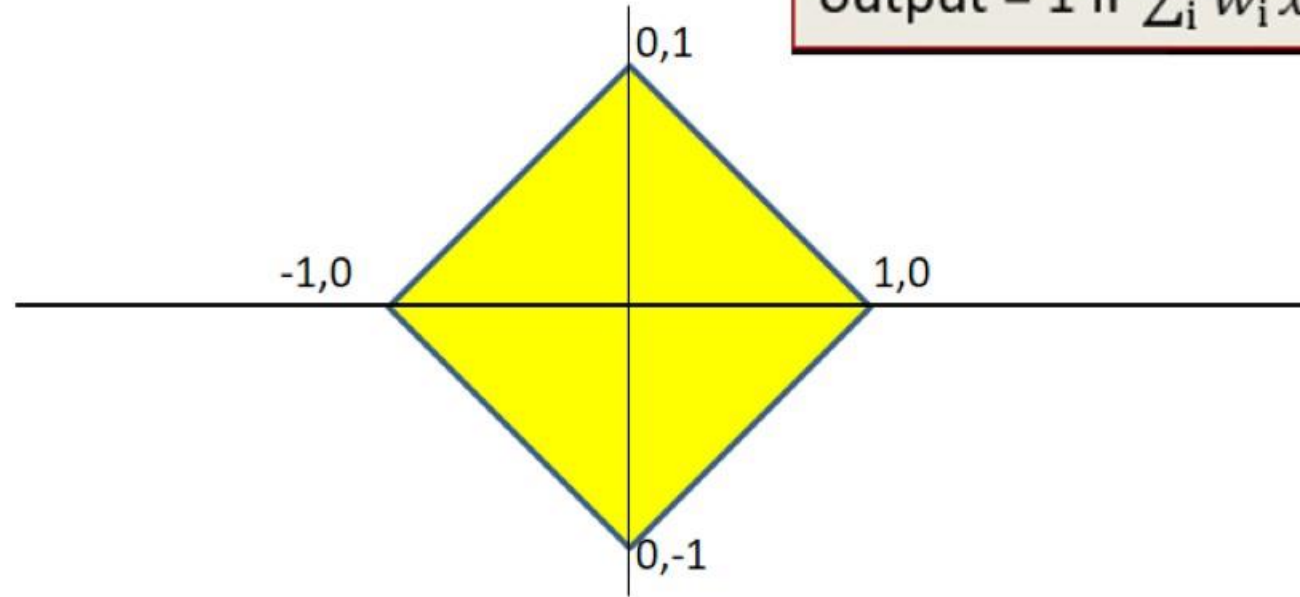


| XOR   |       |     |
|-------|-------|-----|
| $I_1$ | $I_2$ | out |
| 0     | 0     | 0   |
| 0     | 1     | 1   |
| 1     | 0     | 1   |
| 1     | 1     | 0   |



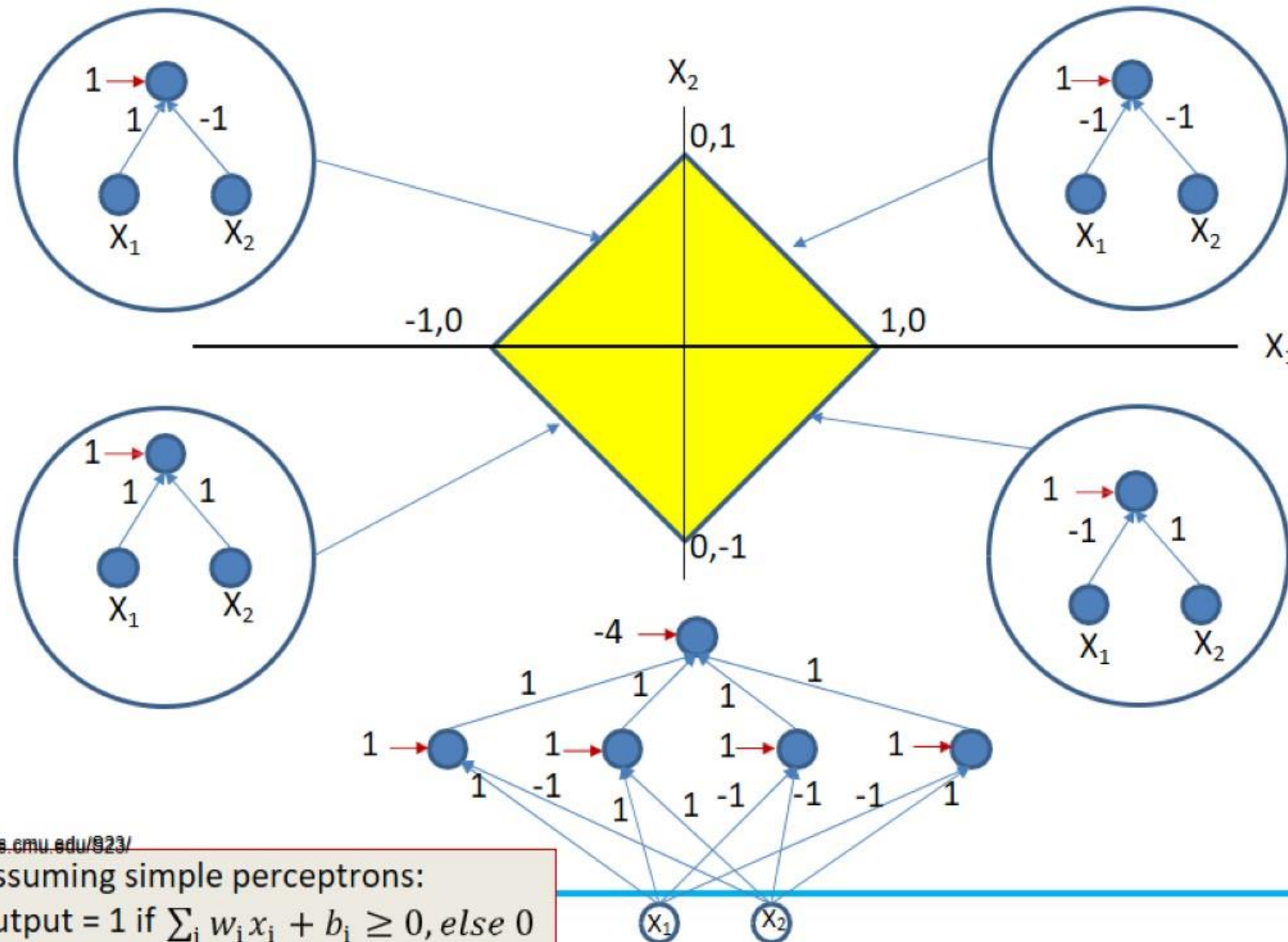
# Option 1: Construct by hand

Assuming simple perceptrons:  
output = 1 if  $\sum_i w_i x_i + b_i \geq 0$ , else 0



- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

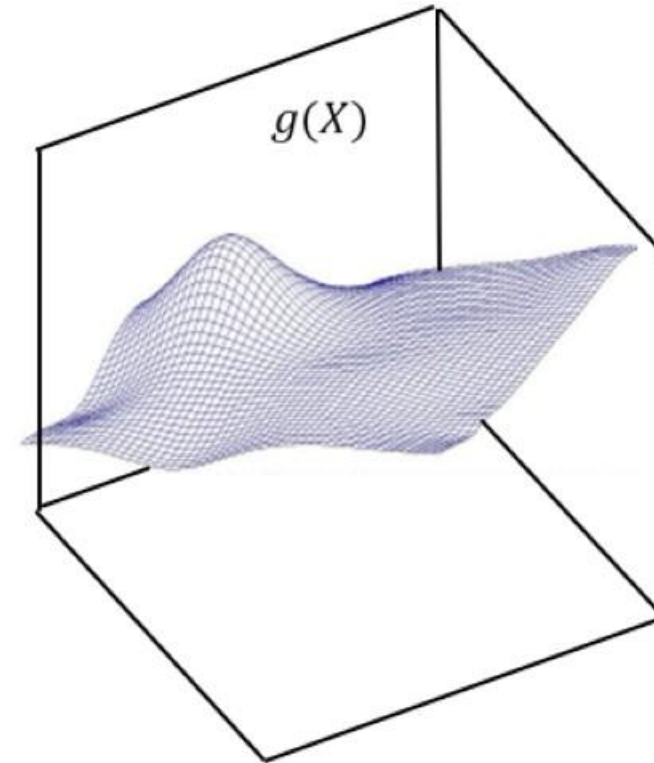
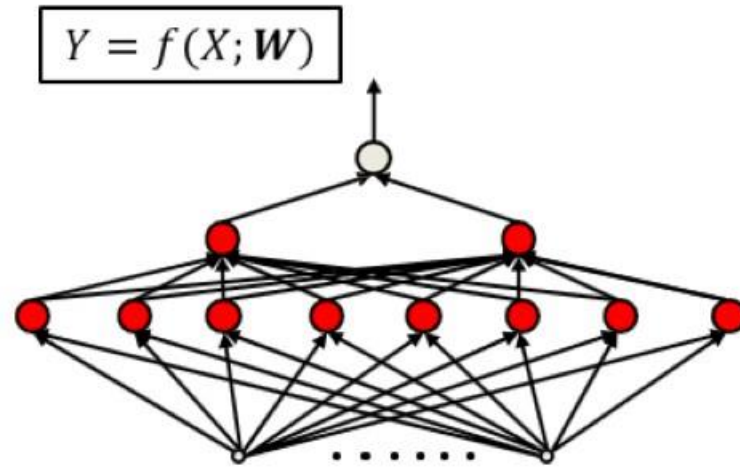
# Option 1: Construct by hand



<https://deeplearning.cs.cmu.edu/923/>

Assuming simple perceptrons:  
output = 1 if  $\sum_i w_i x_i + b_i \geq 0$ , else 0

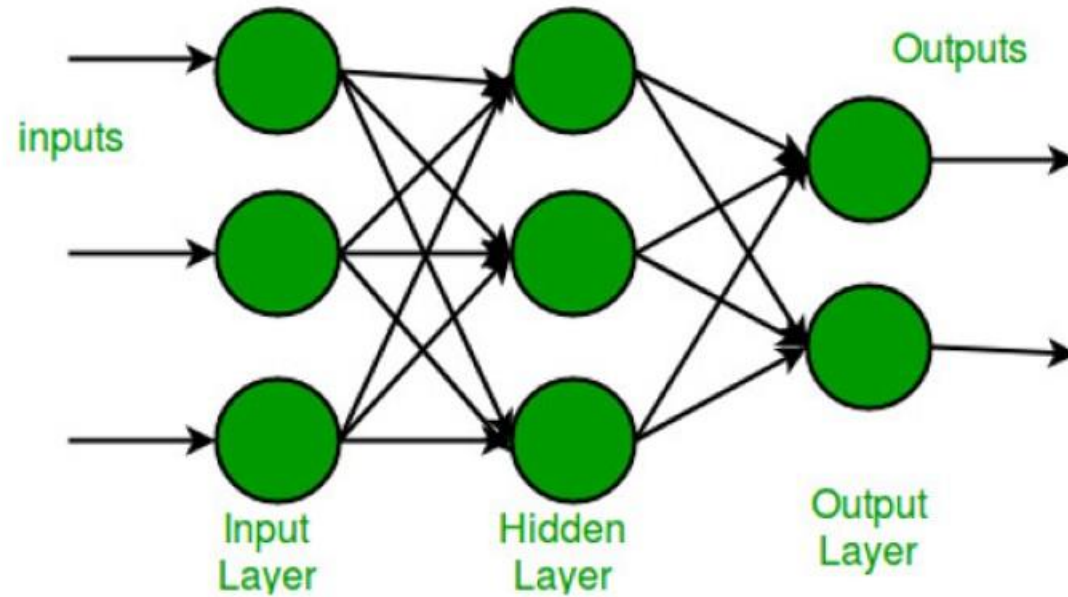
## Option 2: Automatic estimation



- More generally, *given* the function  $g(X)$  to model, we can *derive* the parameters of the network to model it, through computation

# How to learn a network?

- What is the purpose of learning in a multi-layer perceptron (MLP)?
  - To train the model and adjust its weights and biases



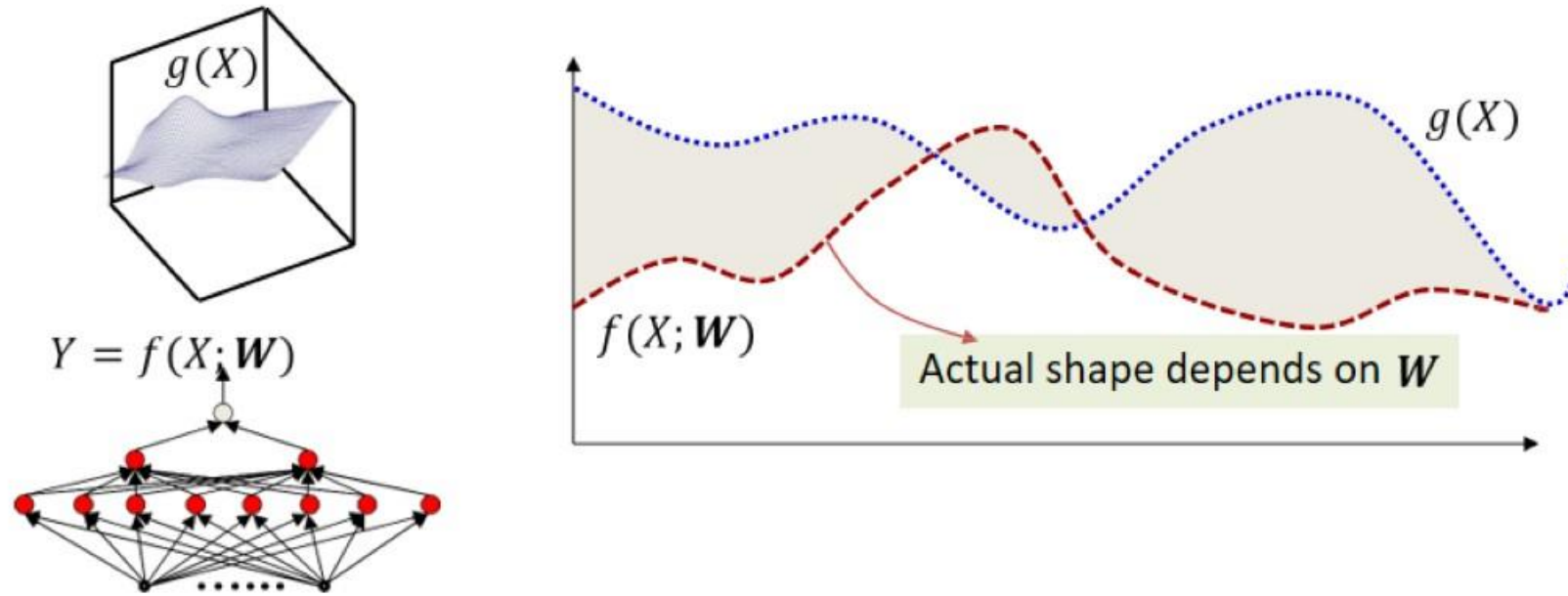
# How to learn a network?

- The input layer is the first layer of the neural network and its purpose is to **receive and process the input data**. The input layer receives data in the form of vectors (arrays of numbers) and typically includes one node (neuron) for each feature in the input data
- The output layer is the last layer of the neural network and its purpose is to **produce the network's prediction or output**.

## Hidden layers

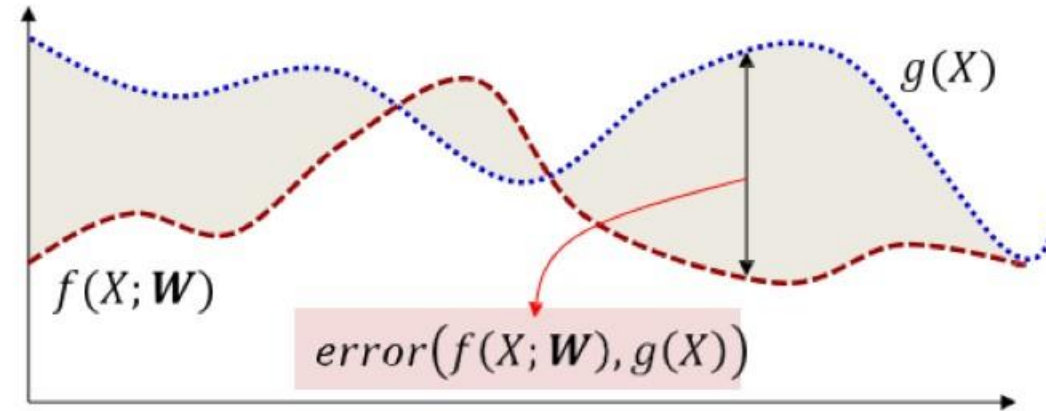
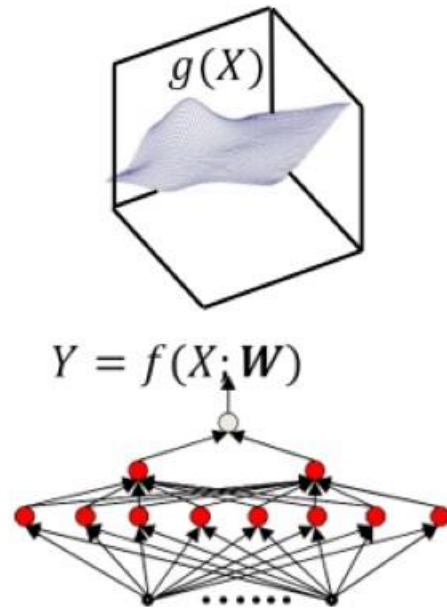
- learn complex representations
- Hidden layers allow for the function of a neural network to be broken down into specific transformations of the data.
- Each hidden layer function is specialized to produce a defined output.
- a hidden layer functions that are used to identify human eyes and ears may be used in conjunction with subsequent layers to identify faces in images. While the functions to identify eyes alone are not enough to independently recognize objects, they can function jointly within a neural network.

# How to learn a network?



- Solution: Estimate parameters to minimize the error between the target function  $g(X)$  and the network function  $f(X, \mathbf{W})$ 
  - Find the parameter  $\mathbf{W}$  that minimizes the shaded area

# How to learn a network?



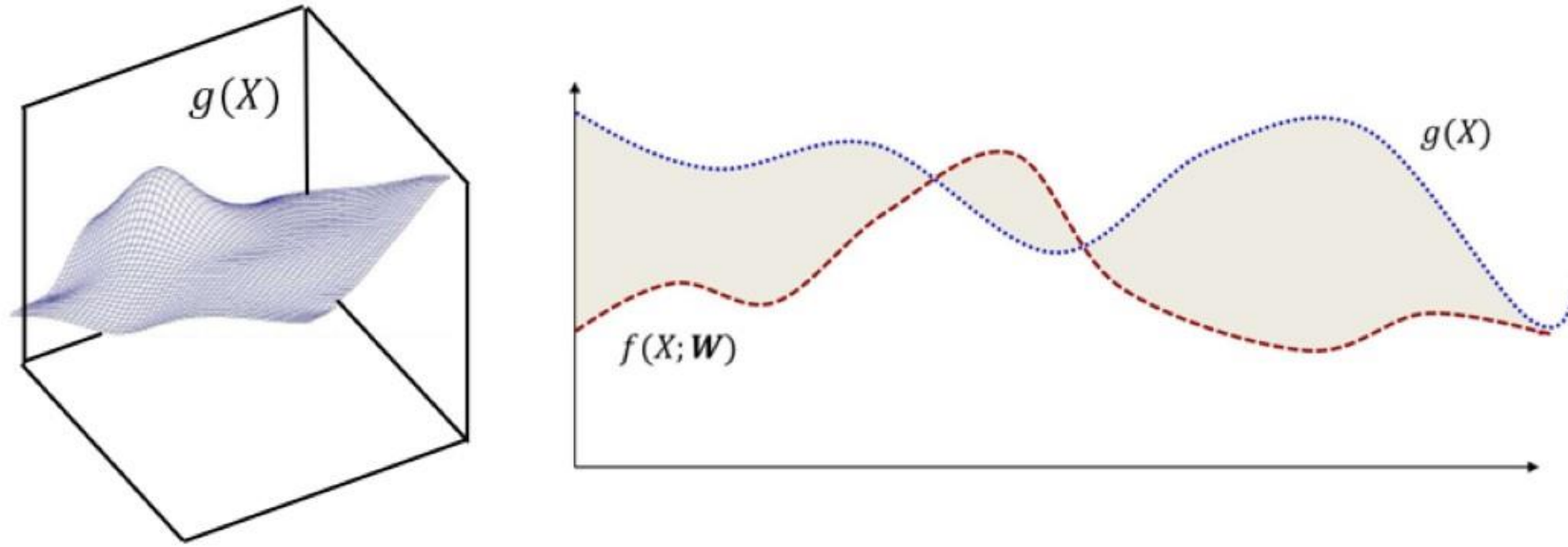
- The shaded area

$$totalerr(\mathbf{W}) = \int_{-\infty}^{\infty} error(f(X; \mathbf{W}), g(X)) dX$$

- The optimal  $\mathbf{W}$

$$\widehat{\mathbf{W}} = \operatorname{argmin}_{\mathbf{W}} totalerr(\mathbf{W})$$

# Problem: $g(X)$



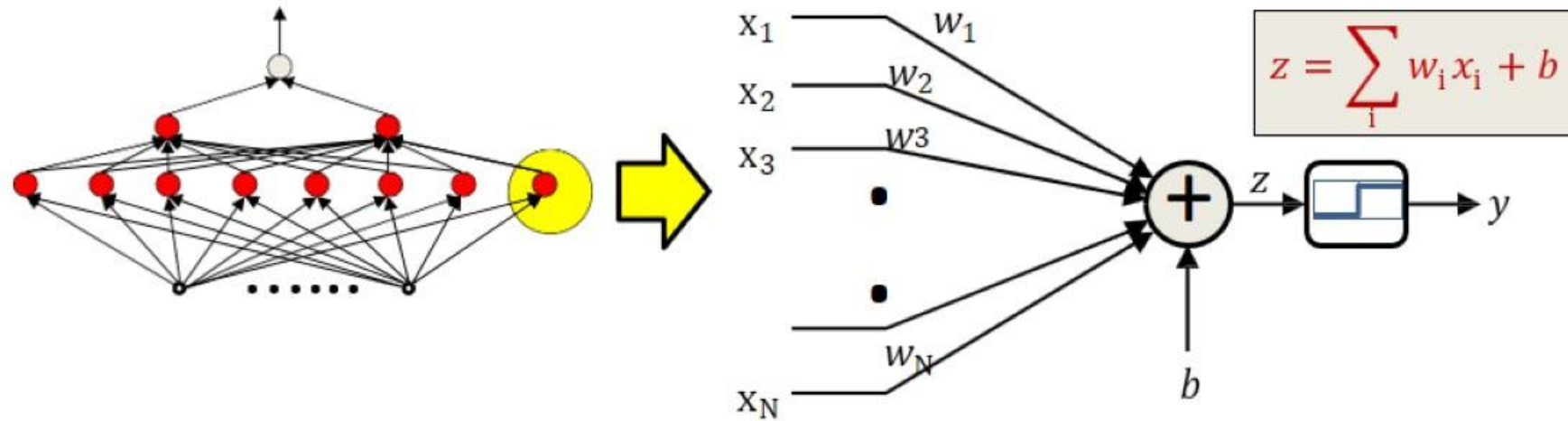
- Function  $g(X)$  must be fully specified in order to compute

$$\int_{-\infty}^{\infty} \text{error}(f(X; \mathbf{W}), g(X)) dX$$

– Known *everywhere*, i.e. for every input  $X$

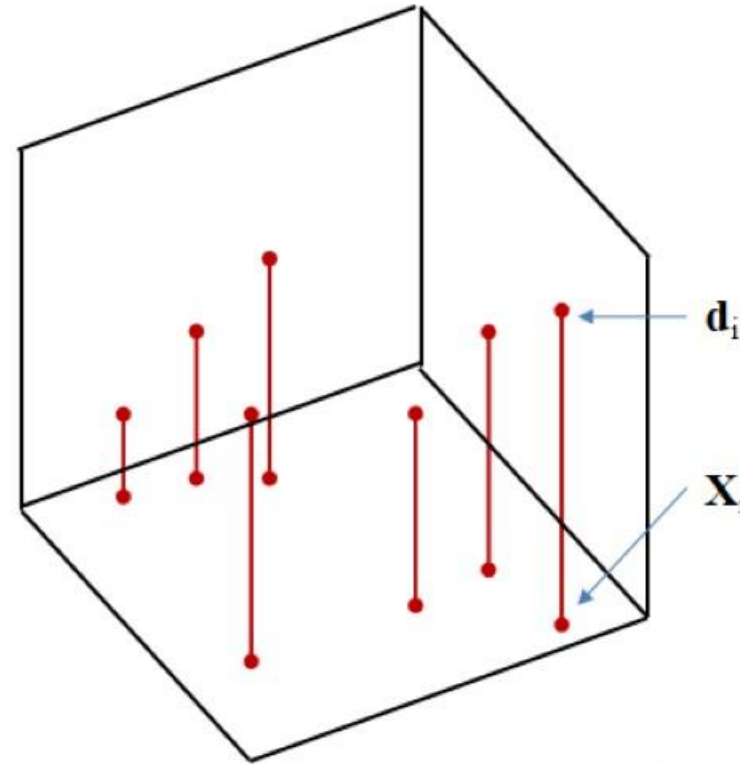
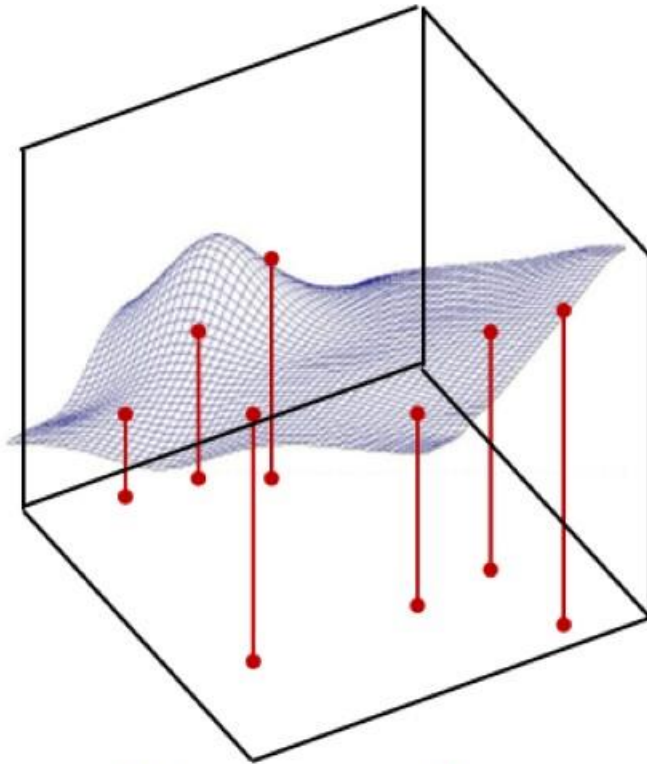
- **In practice we will not have such specification**

# History: The original MLP



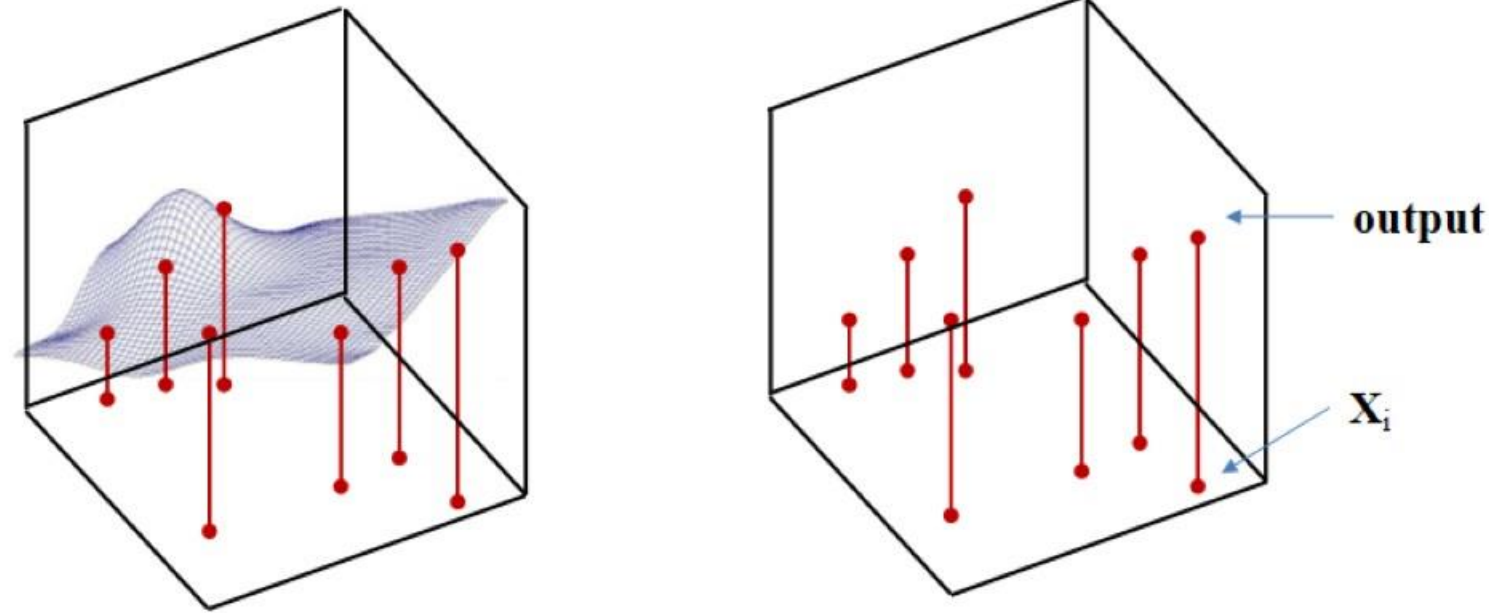
- The original MLP as proposed by Rosenblatt/Minsky: a network of threshold units
  - But how do you train it?
    - Given only “training” instances of input-output pairs

# Drawing samples



- We must **learn** the *entire* function from these few examples
  - The “training” samples

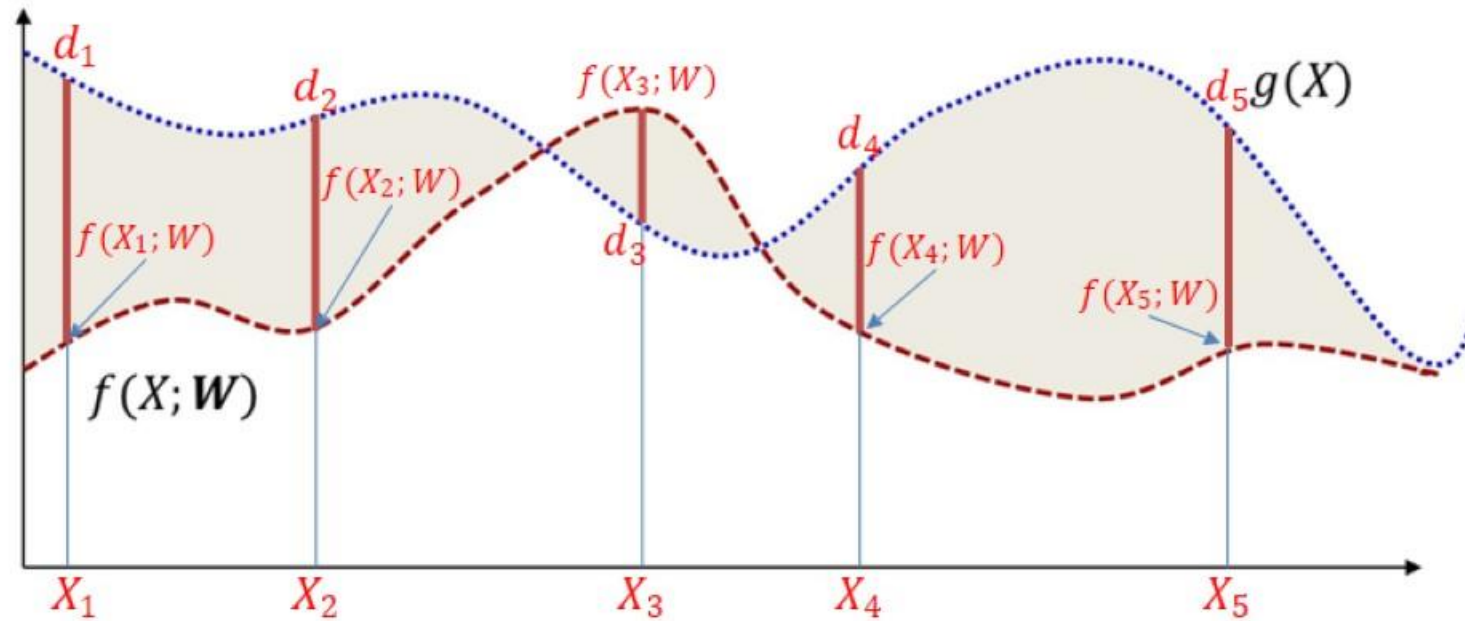
# Sampling the function



Sample – Basically, get input-output pairs for a number of samples of input

- Many samples input + output
- Very easy to do in most problems: just gather training data
  - E.g. set of images and their class labels –
  - E.g. speech recordings and their transcription

# The *Empirical* error(Training Error)



- The *empirical estimate* of the error is the *average* error over the training samples

$$\text{EmpiricalError}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N \text{error}(f(X_i; \mathbf{W}), d_i)$$

- Estimate network parameters to minimize this average error instead

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \text{EmpiricalError}(\mathbf{W})$$

# Learning the function from training samples

- Find the network parameters that “fit” the training points exactly
- hopefully the resulting function is also correct where we don't have training samples

# Learning the function from training samples

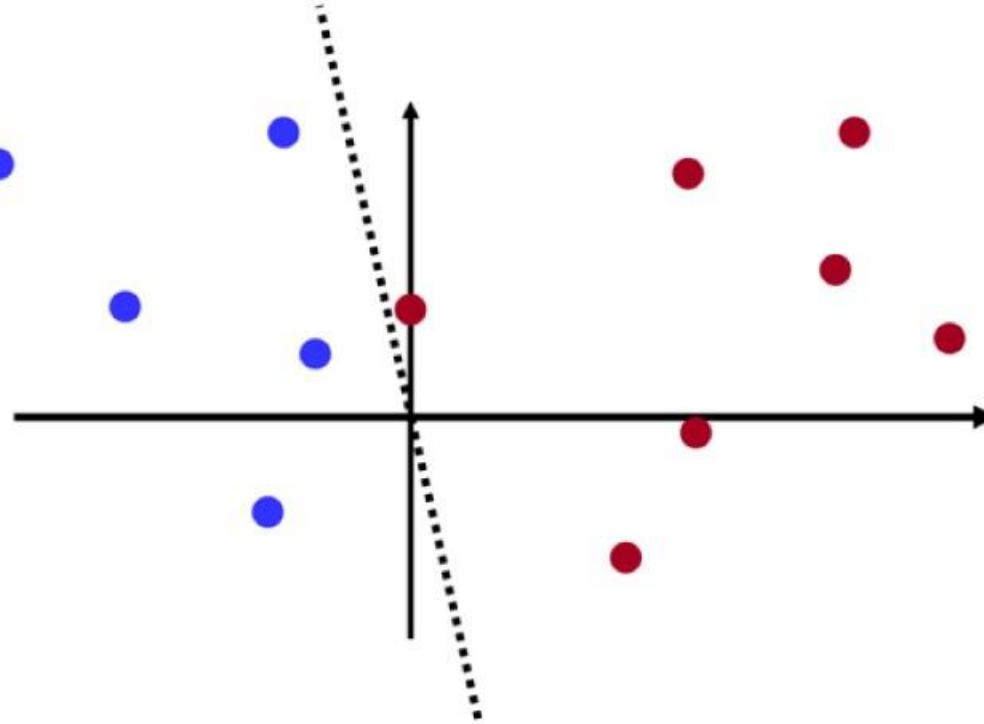
- Find the network parameters that “fit” the training points exactly
- hopefully the resulting function is also correct where we don't have training samples
- This Phase is called the training Phase
- The aim of the training phase is to minimize empirical error(Training Error).
- The test phase: model use new data.

# The *Empirical Classification* error

- Actual = predicted. Classifier is Correct
- Actual  $\neq$  predicted. Classifier is incorrect
- Empirical error = average(Classification Error)
- • Learning the classifier: Minimizing the count of misclassifications

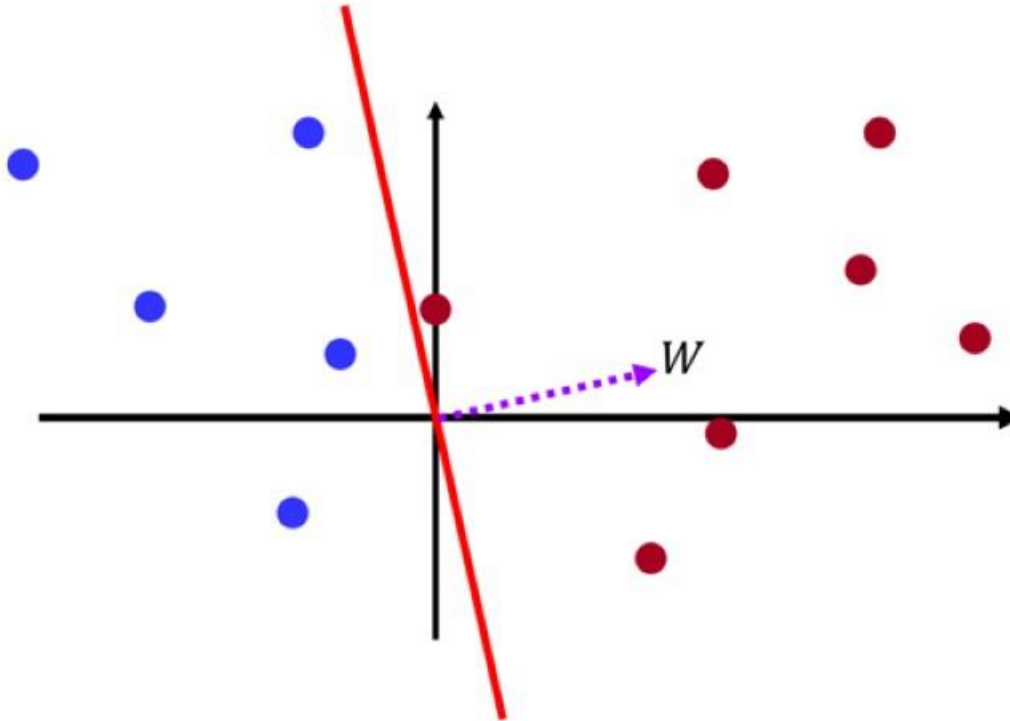
# The Original Perceptron Problem

- Find the hyperplane that perfectly separates the two groups of points
- $\sum_{i=0}^m w_i x_i = 0$



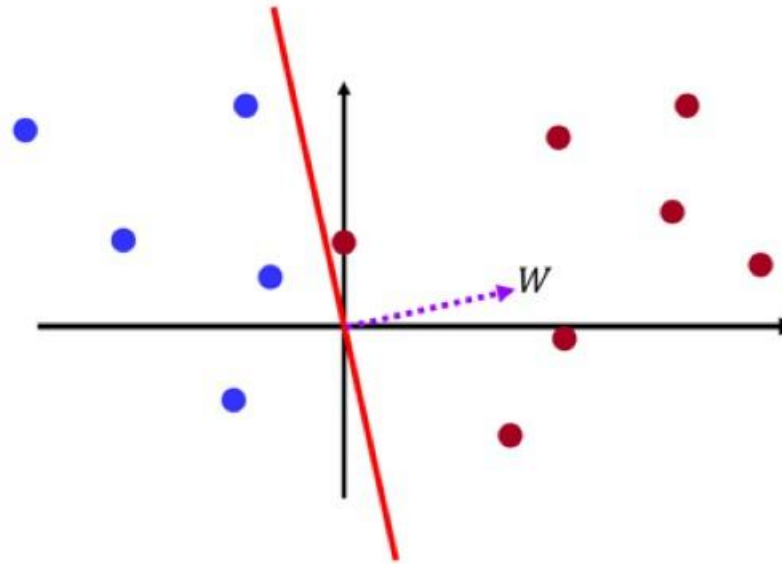
# The Original Perceptron Problem

- Find the hyperplane that perfectly separates the two groups of points
- $\sum_{i=0}^m w_i x_i = 0$
- $W^T x = 0$



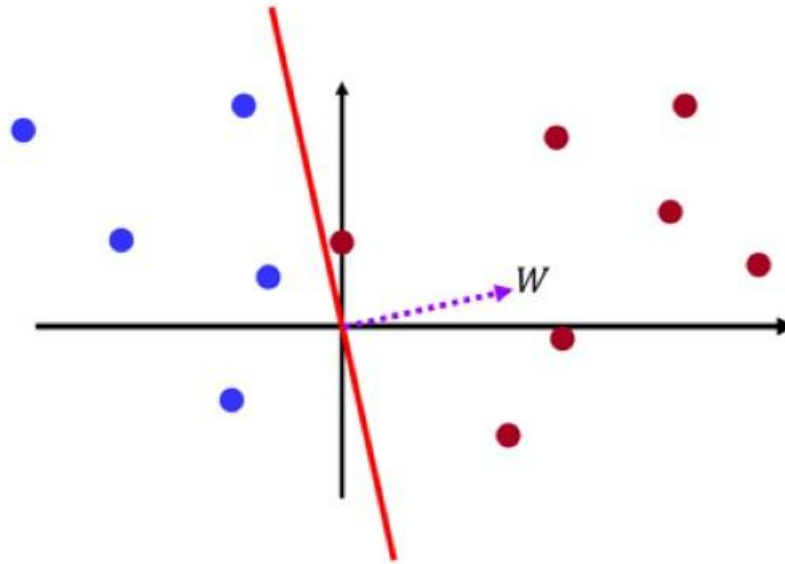
# The Original Perceptron Problem

- $W^T x=0$  is the hyperplane comprising all  $X$ s orthogonal to vector  $W$
- Learning the perceptron = finding the weight vector  $W$  for the separating hyperplane
- $W$  points in the direction of the positive class



# The Original Perceptron Problem

- $W^T x = 0$  is the hyperplane comprising all  $X$ s orthogonal to vector  $W$
- Learning the perceptron = finding the weight vector  $W$  for the separating hyperplane
- $W$  points in the direction of the positive class
- $W^T x$  is positive for all red dots and negative for all blue ones



# The Original Perceptron Problem

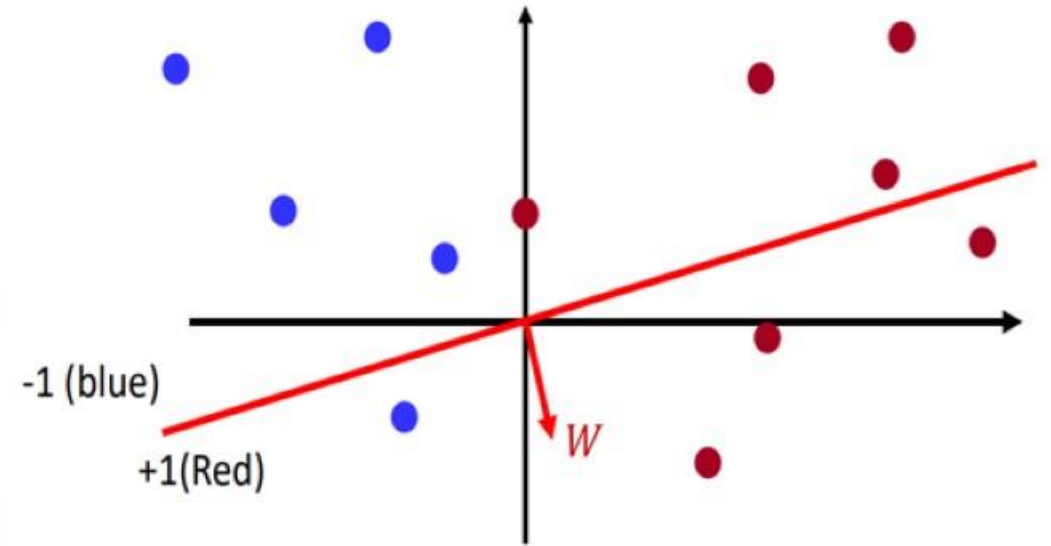
- The popular solution, originally proposed by Rosenblatt
- Cycle through the training instances
- Only update on **misclassified** instances
  - If instance misclassified:
    - If instance is positive class (positive misclassified as negative)  $W = W + x_i$
    - If instance is negative class (negative misclassified as positive)  $W = W - x_i$

# The Original Perceptron Problem

- The popular solution, originally proposed by Rosenblatt
- Cycle through the training instances
- Only update on **misclassified** instances
  - If instance misclassified:
    - If instance is positive class (positive misclassified as negative)  $W = W + x_i$
    - If instance is negative class (negative misclassified as positive)  $W = W - x_i$

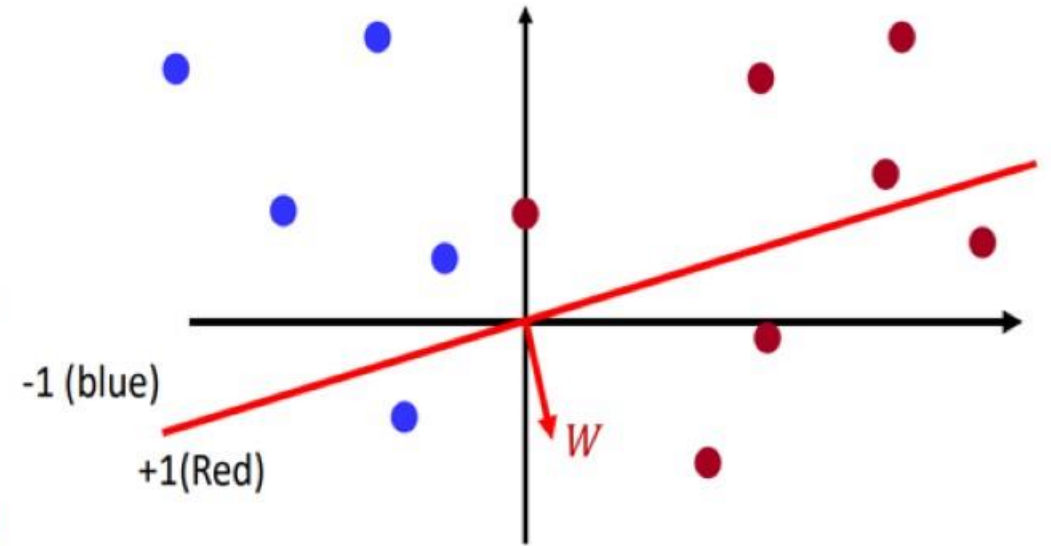
# The Original Perceptron Problem

- Initialize: Randomly initialize the hyperplane (Randomly choose values for  $W$ )
- Classification rule
  - Vectors on the same side of the hyperplane as will be assigned +1 class
  - Vectors on the other side will be assigned -1
- The random initial plane will make **mistakes**

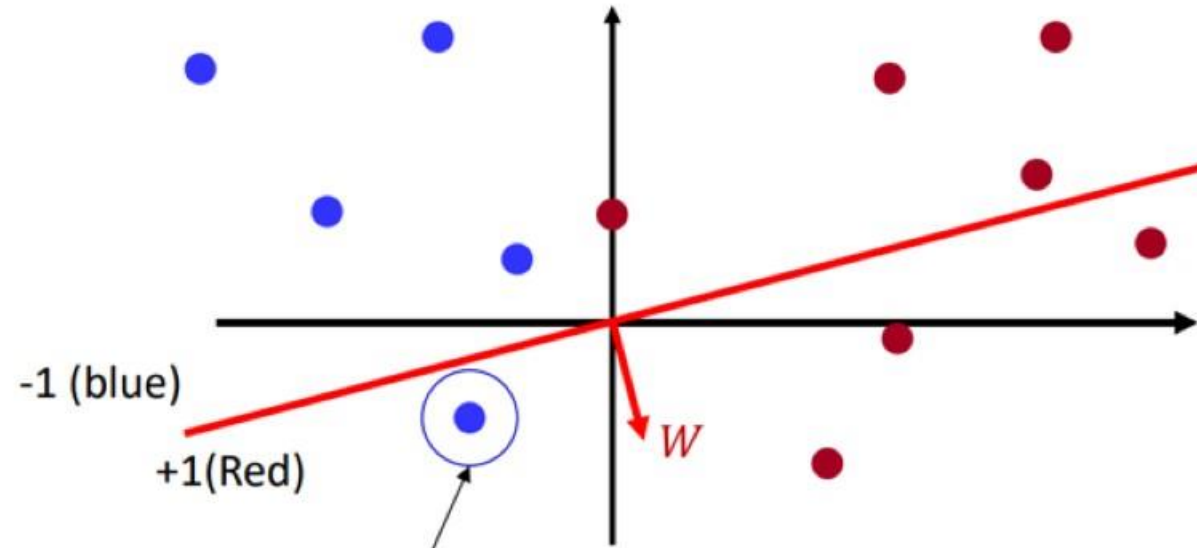


# The Original Perceptron Problem

- Initialize: Randomly initialize the hyperplane (Randomly choose values for  $W$ )
- Classification rule
  - Vectors on the same side of the hyperplane as will be assigned +1 class
  - Vectors on the other side will be assigned -1
- The random initial plane will make **mistakes**



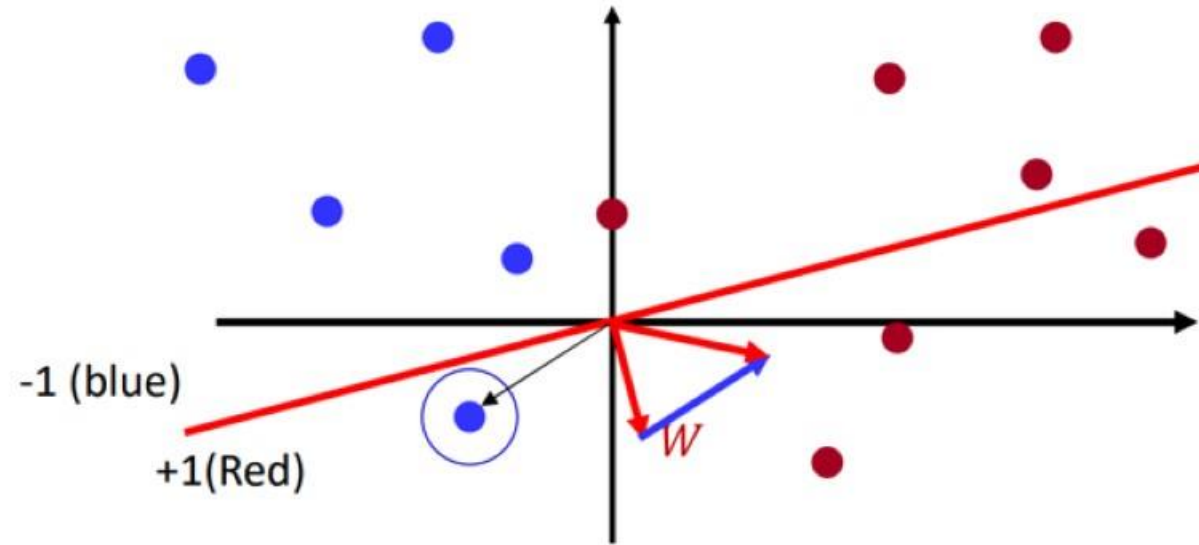
# The Original Perceptron Problem



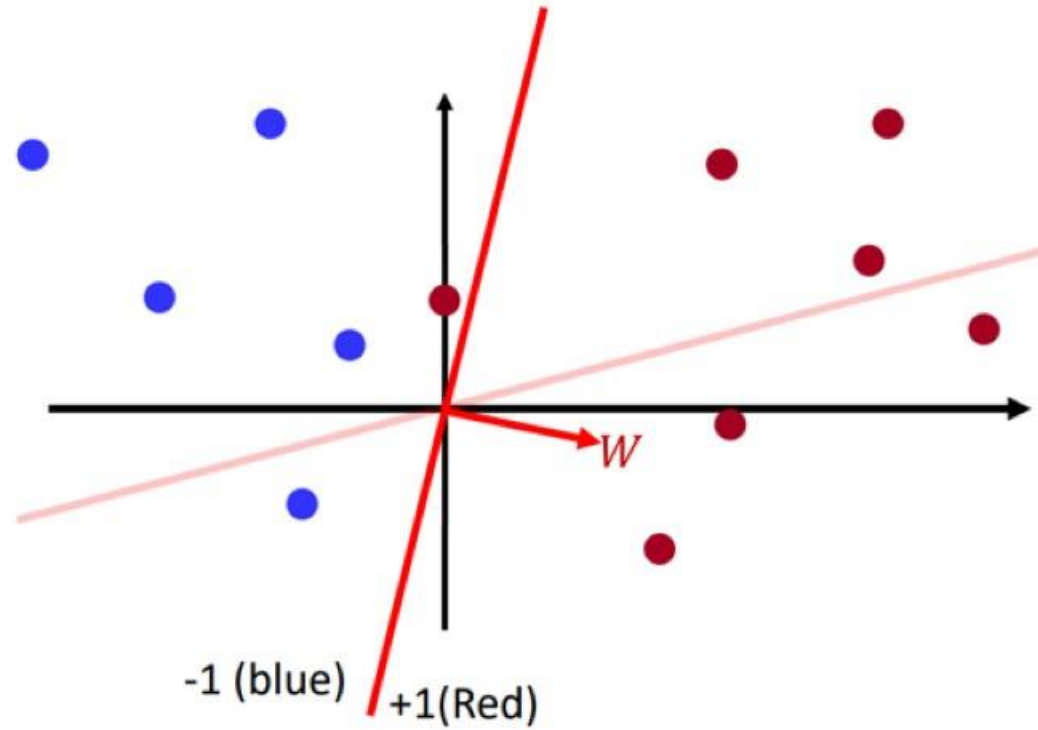
Misclassified negative instance

# The Original Perceptron Problem

If instance is negative class (negative misclassified as positive)  $W = W - x_i$



# The Original Perceptron Problem

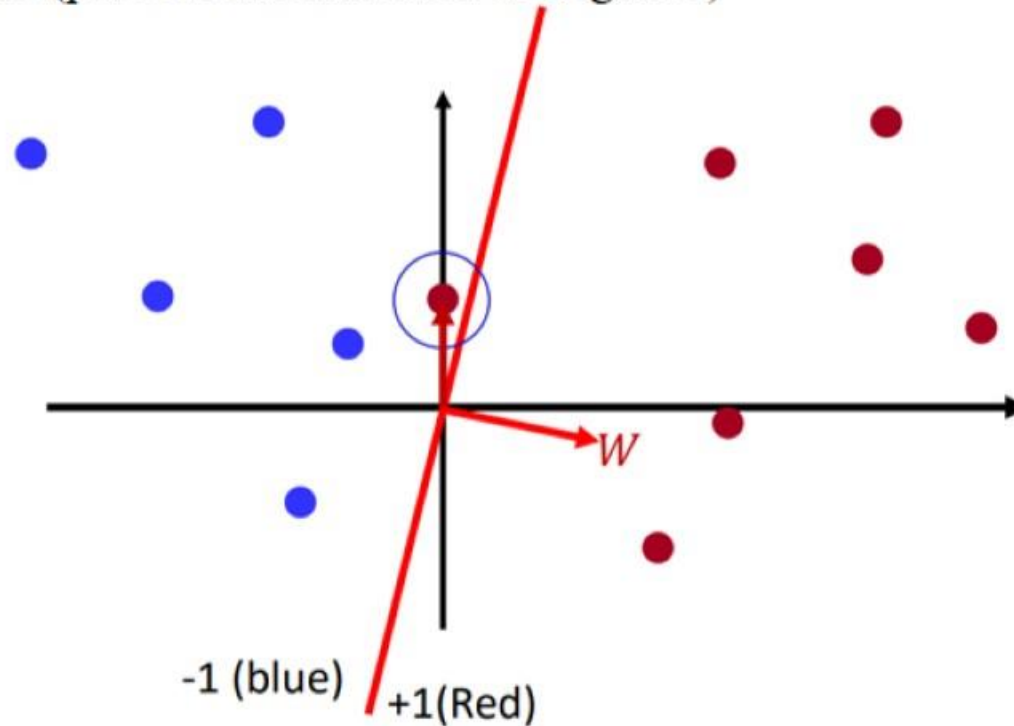


# The Original Perceptron Problem

Misclassified positive instance

If instance is positive class (positive misclassified as negative)

$$W = W + x_i$$



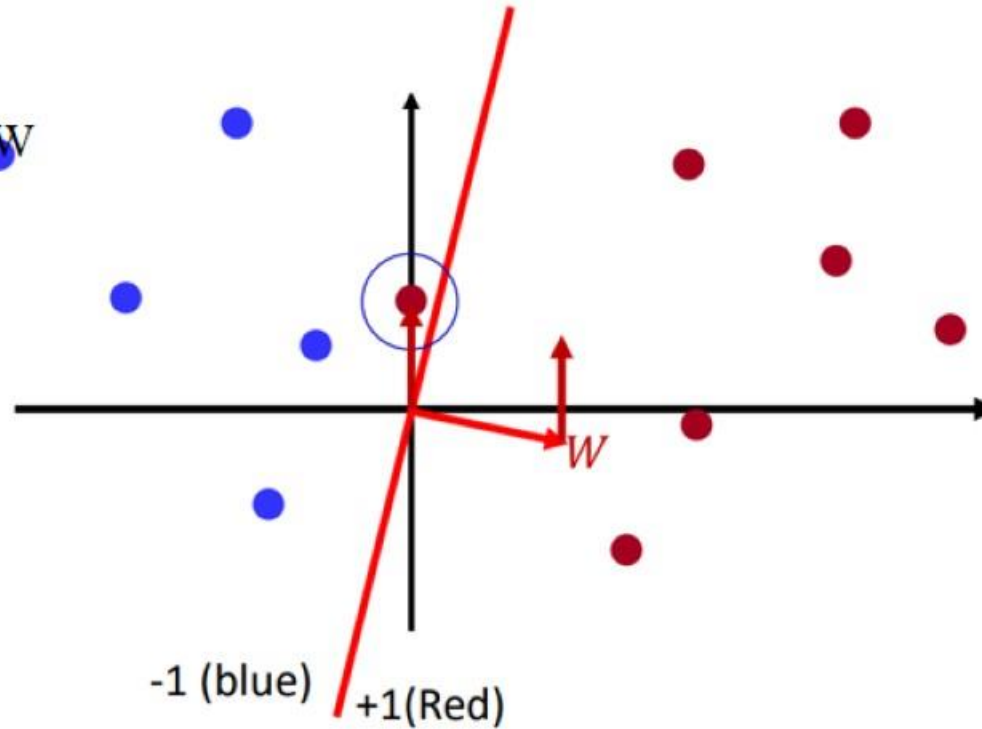
# The Original Perceptron Problem

Misclassified positive instance

If instance is positive class (positive misclassified as negative)

$$W = W + x_i$$

Misclassified positive instance, add it to  $W$



# The Original Perceptron Problem

Misclassified positive instance

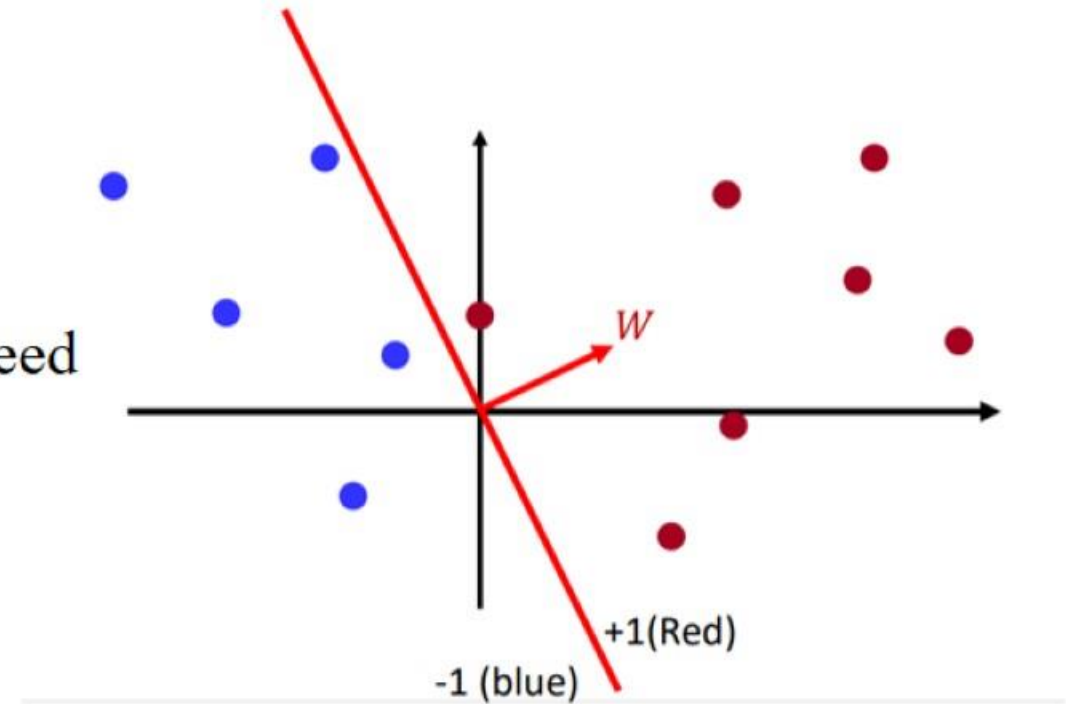
If instance is positive class (positive misclassified as negative)

$$W = W + x_i$$

Misclassified positive instance, add it to W

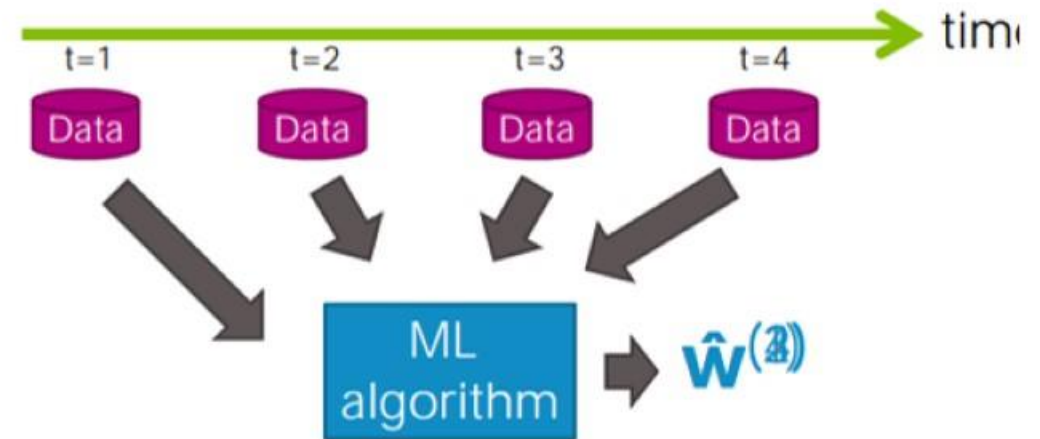
# The Original Perceptron Problem

- The new decision boundary Perfect classification, no more updates, we are done.
- All points are correctly classified.
- If the classes are linearly separable, guaranteed to converge in a finite number of steps



# Batch vs online learning

- The perceptron algorithm is an **online learning**, which means → It can update its weights after each training example



# Perceptron Learning: Simple Example

Input : Set of examples, each example is a pair an **input vector** and a **label** (0 or 1).

- Initialize weights randomly
- Cycle (repeatedly) through examples and for each example do:
  - – If perceptron is 0 while it should be 1,
    - **add** the input vector to the weight vector
  - – If perceptron is 1 while it should be 0,
    - **subtract** the input vector to the weight vector
  - – Otherwise do **nothing**.

# Perceptron Learning: Simple Example

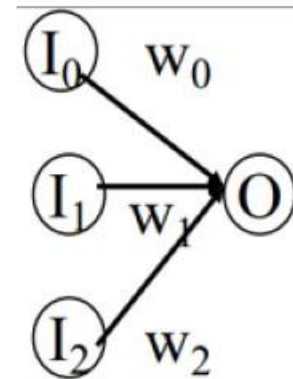
Our examples are:

| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |

# Perceptron Learning: Simple Example

- use a single perceptron with three inputs.
- start with all weights randomly
  - (we use 0 for simple calculation)  $\rightarrow [0 \ 0 \ 0]$
- Example 1  $x = [1 \ 0 \ 0]$  label=0
  - Perceptron ( $1*0 + 0*0 + 0*0 = 0$ , Class=0) output 0
  - correct classification, do nothing
- Example 2  $x = [1 \ 0 \ 0]$  label=1
  - Perceptron ( $1*0 + 0*0 + 1*0 = 0$ ) output 0
  - classifies it as 0, while it should be 1,
  - add input to weights  $W = [0 \ 0 \ 0] + [1 \ 0 \ 1] = [1 \ 0 \ 1]$

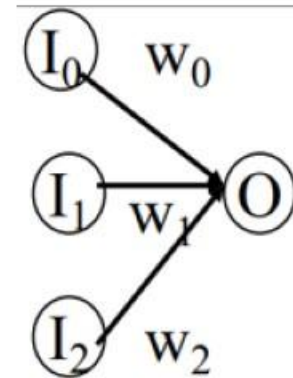
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- use a single perceptron with three inputs.
- $W=[1 \ 0 \ 1]$
- Example 3  $x= [1 \ 1 \ 0]$  label=1
  - Perceptron ( $1*1+ 1*0+ 0*1 =1$ , Class=1) output 1
  - correct classification, do nothing
- Example 4  $x= [1 \ 1 \ 1]$  label=1
  - Perceptron ( $1*1+ 1*0+ 1*1 = 1$ , Class=1) output 1
  - Correct classification, do nothing

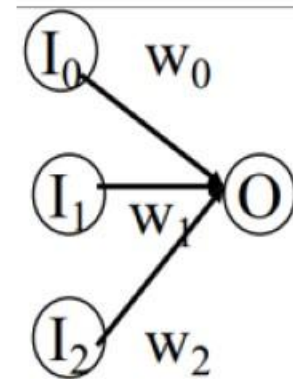
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- Epoch 2:
- $W=[1 \ 0 \ 1]$
- Example 1  $x=[1 \ 0 \ 0]$  label=0
  - Perceptron ( $1*0+ 0*0+ 0*1 =1$ , Class=0) output 1
  - classifies it as 1, while it should be 0
  - subtract input from weights =  $[1 \ 0 \ 1]-[1 \ 0 \ 0]=[0 \ 0 \ 1]$
- Example 2  $x=[1 \ 0 \ 1]$  label=1
  - Perceptron ( $1*0+ 0*0+ 1*1 > 0$ , class=1) output 1
  - do noting

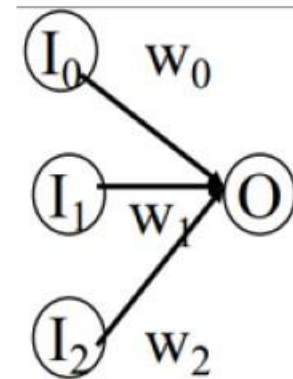
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- Epoch=2
- $W=[0 \ 0 \ 1]$
- Example 3  $x=[1 \ 1 \ 0]$  label=1
  - Perceptron ( $1*0+ 1*0+ 0*1 =0$ , Class=1) output 0
  - Add to weight= $[0 \ 0 \ 1]+[1 \ 1 \ 0]=[1 \ 1 \ 1]$
- Example 4  $x=[1 \ 1 \ 1]$  label=1
  - Perceptron ( $1*1+ 1*1+ 1*1 = 1$ , Class=1) output 1
  - Correct classification, do nothing

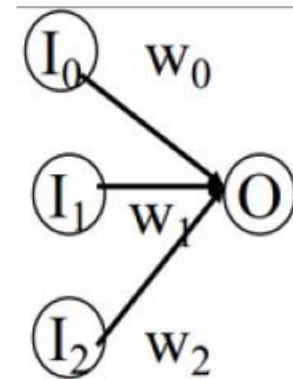
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- Epoch 3:
- $W=[1 \ 1 \ 1]$
- Example 1  $x=[1 \ 0 \ 0]$  label=0
  - Perceptron ( $1*1+ 0*1+ 0*1 =1$ , Class=0) output 1
  - classifies it as 1, while it should be 0
  - subtract input from weights =  $[1 \ 1 \ 1]-[1 \ 0 \ 0]=[0 \ 1 \ 1]$
- Example 2  $x=[1 \ 0 \ 1]$  label=1
  - Perceptron ( $1*0+ 0*1+ 1*1 > 0$ , class=1) output 1
  - do noting

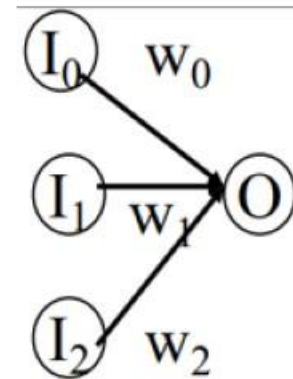
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- Epoch=3
- $W=[0 \ 1 \ 1]$
- Example 3  $x = [1 \ 1 \ 0]$  label=1
  - Perceptron ( $1*0 + 1*1 + 0*1 \geq 0$ , Class=1) output 1
  - Correct classification, do nothing
- Example 4  $x = [1 \ 1 \ 1]$  label=1
  - Perceptron ( $1*0 + 1*1 + 1*1 = 1$ , Class=1) output 1
  - Correct classification, do nothing

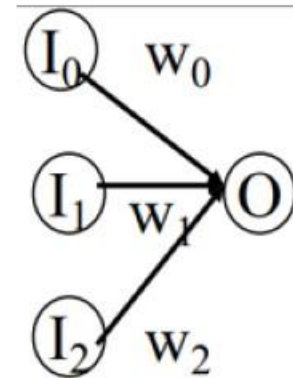
| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



# Perceptron Learning: Simple Example

- Epoch 4:
- $W=[0 \ 1 \ 1]$
- Example 1  $x=[1 \ 0 \ 0]$  label=0
  - Perceptron ( $0*1+ 0*1+ 0*1 =1$ , Class=0) output 1
  - do noting

| Sample | x0 | x1 | x2 | label |
|--------|----|----|----|-------|
| 1      | 1  | 0  | 0  | 0     |
| 2      | 1  | 0  | 1  | 1     |
| 3      | 1  | 1  | 0  | 1     |
| 4      | 1  | 1  | 1  | 1     |



| Epoch       | x0 | x1 | x2 | Desired<br>Target | w0 | w1 | w2 | Output | Error | New<br>w0 | New<br>w1 | New<br>w2 |
|-------------|----|----|----|-------------------|----|----|----|--------|-------|-----------|-----------|-----------|
| 1 example 1 | 1  | 0  | 0  | 0                 | 0  | 0  | 0  | 0      | 0     | 0         | 0         | 0         |
|             |    |    |    |                   |    |    |    |        |       |           |           |           |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 1  | 1  | 1      | 0     | 1      | 1      | 1      |



| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 1  | 1  | 1      | 0     | 1      | 1      | 1      |
| 3 example 1 | 1  | 0  | 0  | 0              | 1  | 1  | 1  | 1      | -1    | 0      | 1      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 1  | 1  | 1      | 0     | 1      | 1      | 1      |
| 3 example 1 | 1  | 0  | 0  | 0              | 1  | 1  | 1  | 1      | -1    | 0      | 1      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 1  | 1  | 1      | 0     | 1      | 1      | 1      |
| 3 example 1 | 1  | 0  | 0  | 0              | 1  | 1  | 1  | 1      | -1    | 0      | 1      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |

| Epoch       | x0 | x1 | x2 | Desired Target | w0 | w1 | w2 | Output | Error | New w0 | New w1 | New w2 |
|-------------|----|----|----|----------------|----|----|----|--------|-------|--------|--------|--------|
| 1 example 1 | 1  | 0  | 0  | 0              | 0  | 0  | 0  | 0      | 0     | 0      | 0      | 0      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 0  | 0      | 1     | 1      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 0  | 1  | 1      | 0     | 1      | 0      | 1      |
| 2 example 1 | 1  | 0  | 0  | 0              | 1  | 0  | 1  | 1      | -1    | 0      | 0      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 0  | 1  | 1      | 0     | 0      | 0      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 0  | 1  | 0      | 1     | 1      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 1  | 1  | 1  | 1      | 0     | 1      | 1      | 1      |
| 3 example 1 | 1  | 0  | 0  | 0              | 1  | 1  | 1  | 1      | -1    | 0      | 1      | 1      |
| example 2   | 1  | 0  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| example 3   | 1  | 1  | 0  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| example 4   | 1  | 1  | 1  | 1              | 0  | 1  | 1  | 1      | 0     | 0      | 1      | 1      |
| 4 example 1 | 1  | 0  | 0  | 0              | 0  | 1  | 1  | 0      | 0     | 0      | 1      | 1      |

# Perceptron Learning: Simple Example

- In more realistic cases the bias ( $W_0$ ) will not be 0.
- Also, in general, many more inputs (100 to 1000)
- Each time a dataset passes through an algorithm, it is said to have completed an **epoch**. Therefore, Epoch, in machine learning, refers to the one entire passing of training data through the algorithm.
- A machine learning model reaches **convergence** when it achieves a state during training in which loss settles to within an error range around the final value. In other words, a model converges when additional training will not improve the model.

- Single Layer Perceptron : Review
- Learning Problem
- Perceptron Learning algorithm
- Perceptron Learning algorithm: Example

*Thank  
you*

