Minia University
Faculty of Computers & information

Artificial Neural Networks and Deep Learning

Slides By:

A. T. Sarah Osama Talaat

✉ E-mail: SarahOsama.fci@gmail.com

Slides were prepared based on set of references mentioned in the last slide

Lectures, FCI, Mina University

Agenda

- ❑Revision
- ❑Mathematical Basics
  - ❑Derivatives
- ❑Supervised Learning Network Paradigms
  - ❑Learning rate
  - ❑Perceptron
  - ❑The delta rule as a gradient based learning strategy for SLPs
  - ❑Multi-layer perceptron

Let’s Start

Revision
Offline or online learning?

❑ Offline learning:

- In this learning, a **set** of training samples is presented, then the weights are changed, the total error is calculated by means of an error function operation or simply accumulated. Offline training procedures are also called **batch training procedures** since a batch of results is corrected all at once. Such a training section of a whole batch of training samples including the related change in weight values is called **epoch**.

❑ Definition 4.2 (Offline learning).

- Several training patterns are entered into the network at once, the errors are accumulated and it learns for all patterns at the same time.

Revision
Offline or online learning?

❑ Online learning:

- In this type of learning, after **every** sample presented the weights are changed.

❑ Definition 4.3 (Online learning):

- The network learns directly from the errors of each training sample..

Revision
Delta rule for learning

❑ Definition 4.21 (Delta rule *or* windrow-Hoff rule):

- Continuously modifying the strengths of the input connections to reduce the difference (the delta) between the **desired output (i.e. teaching input)** value and the actual output of a processing element. Derivative of the transfer function is used.

Revision

Delta rule for learning

Delta rule for learning versions

Offline delta rule

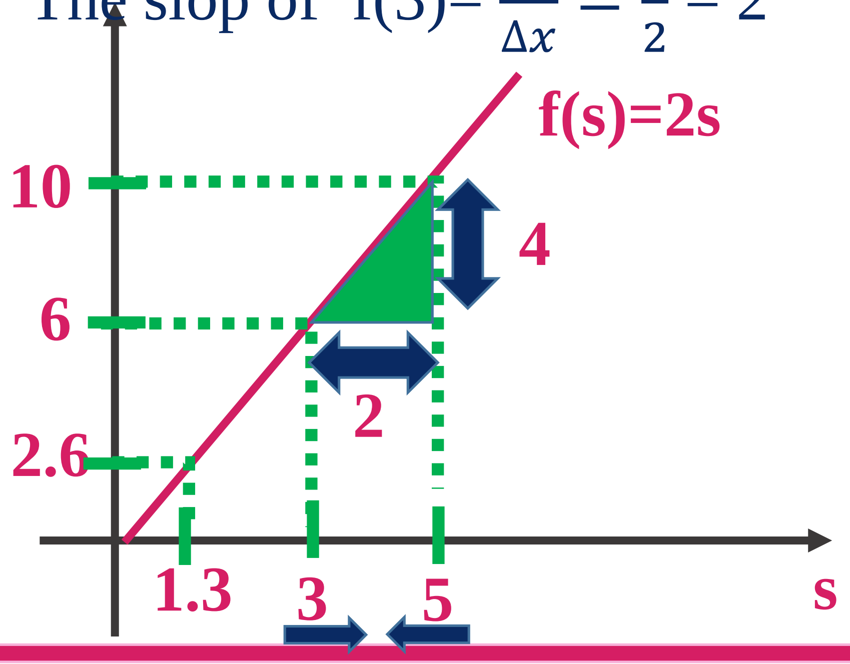
Online Delta rule

Mathematical Basics

Derivatives

❑ Intuition about derivatives

- Assume that, we have a following function f(s)=2s
- The slope (derivative) of a function is given by $= \frac{\Delta y}{\Delta x}$
- The slop of f(3)= $\frac{\Delta y}{\Delta x} = \frac{4}{2} = 2$



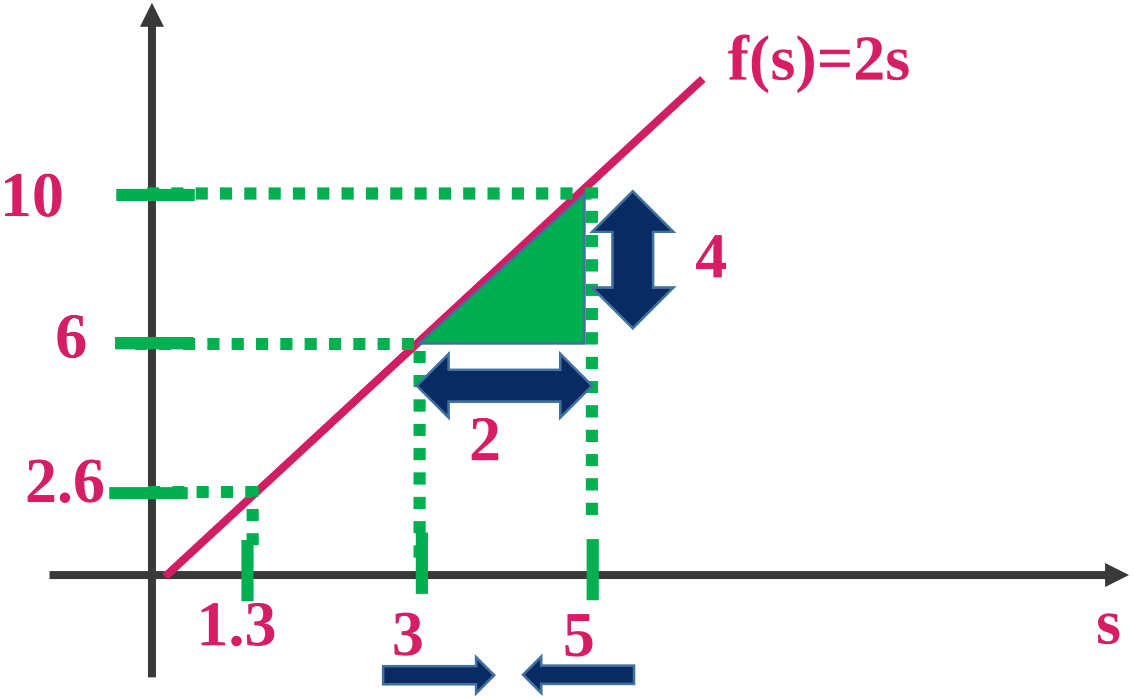
s	f(s)
3	2×3=6
5	2× 5=10
1.3	2× 1. 3=2.6
0	2×0=0

Mathematical Basics

Derivatives

❑ Intuition about derivatives:

- The slope (derivative) of a function f(s) is given by $\frac{\Delta y}{\Delta x}$
- Also, the slop is equal to = $\frac{df(s)}{ds} = \frac{d}{ds}f(s)$
- If s=3 the slope = $\frac{df(3)}{d3}$= $\frac{4}{2}$ $= 2$

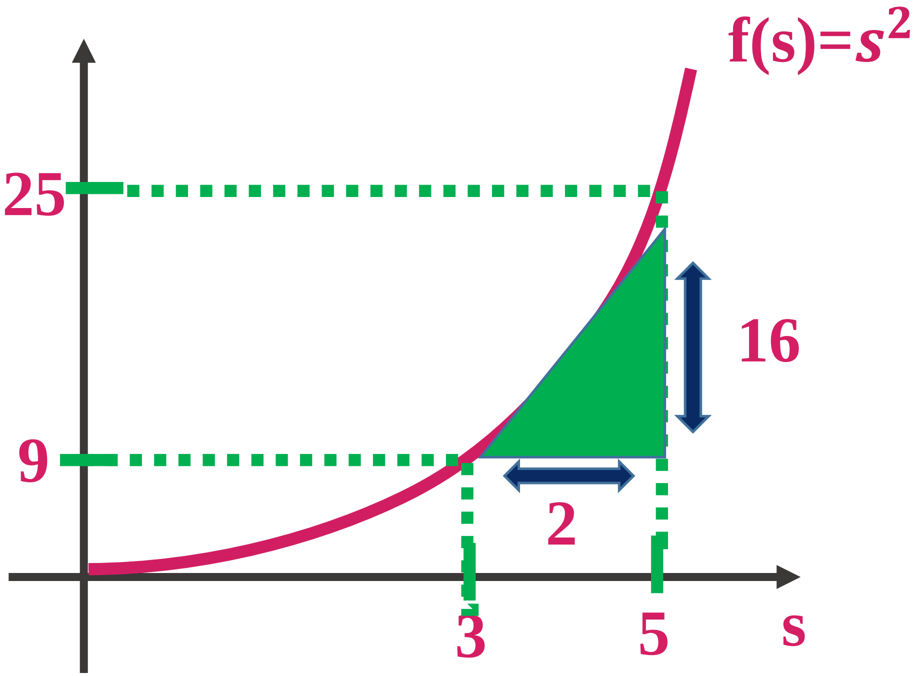


Mathematical Basics

Derivatives

❑ Derivatives examples

- Assume that, we have a following function f(s)=s^2



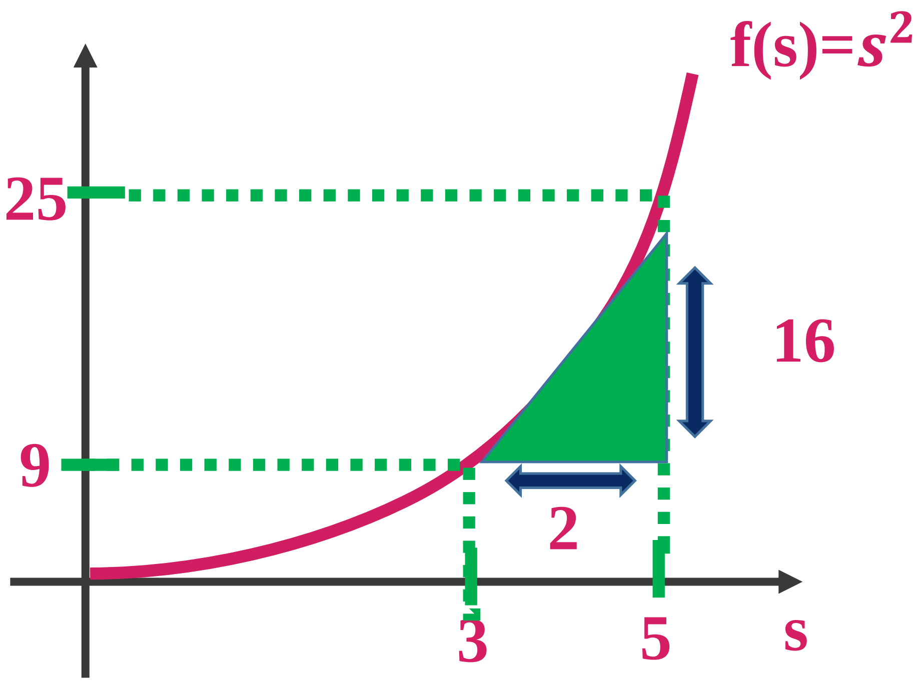
s	f(s)
3	3×3=9
5	5× 5=25

Mathematical Basics

Derivatives

❑ Derivatives examples

- The slope (derivative) of a function f(s) at s= 3 =$\frac{df(3)}{d3} = 9$
- $\frac{df(s)}{ds} = 9 \; where \; s = 3$
- $\frac{df(s)}{ds} = 25 \; where \; s = 5$
- $\frac{d}{ds}\mathrm{f(s)} = \frac{d}{ds}s^2 = 2\mathrm{s}$



Mathematical Basics

Derivatives

❑ Derivatives examples

- **When we have a following function f(s)=$\boldsymbol{s^2}$ the derivative will be $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)} = \boldsymbol{2s}$**
- $where\ \ s = 2.001\ \ the$ f(s) ≈ 4.004 and $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)}$= 4.002
- $where\ \ s = 5\ \ the$ f(s) $= 25$ and $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)}$= 10
- **When we have a following function f(s)=$\boldsymbol{s^3}$ the derivative will be $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)} = \boldsymbol{3s^2}$**
- $where\ \ s = 2.001\ \ the$ f(s) ≈ 8.012 and $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)} \approx 12.012$
- $where\ \ s = 5\ \ the$ f(s) $= 125$ and $\frac{\boldsymbol{d}}{\boldsymbol{ds}}\boldsymbol{f(s)}$=7

Supervised Learning Network Paradigms

❑ Supervised Learning procedures:

- **Entering** the input pattern (activation of input neurons),
- **Forward propagation** of the input by the network, generation of the output,
- **Comparing** the output with the desired output (***teaching input***), *provides an **error vector*** (difference vector),
- **Corrections** of the network are calculated based on the error vector,
- **Corrections** are applied.

Supervised Learning Network Paradigms

The perceptron

- The term **perceptron** in the most of the time is used to describe a **feedforward network with shortcut connections**.
- This network has a layer of **scanner neurons** (retina) with statically weighted connections to the following layer and is called **input layer** (figure 5.1 on the next slide); but the weights of all other layers are allowed to be changed. All neurons subordinate to the retina are pattern detectors.

Supervised Learning Network Paradigms

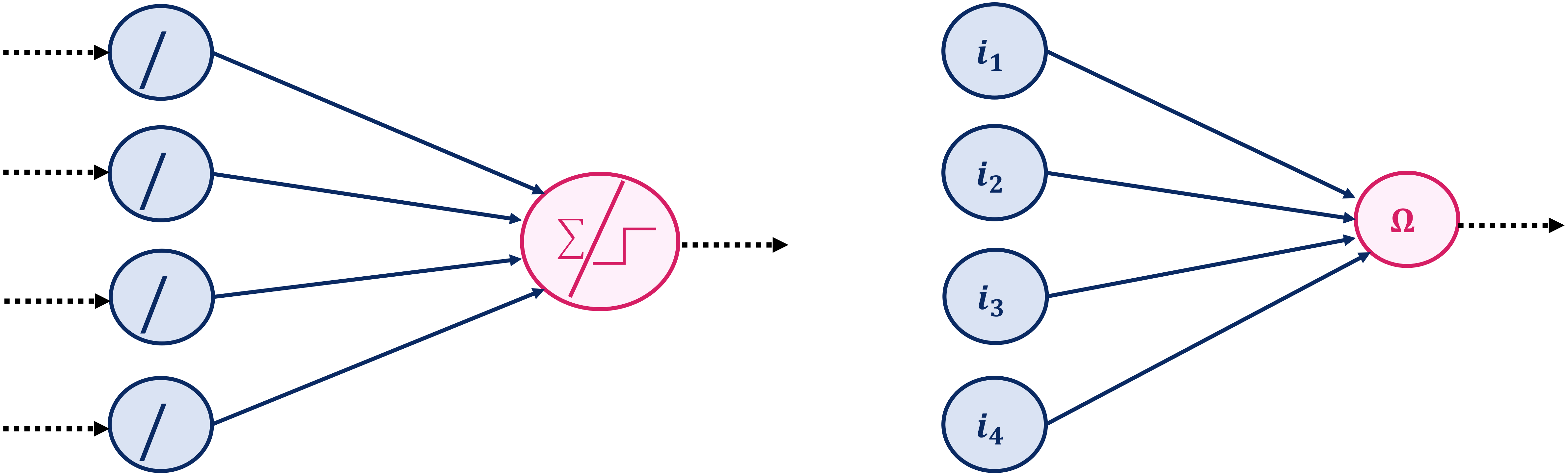
The perceptron

❑ Definition 5.2 (Perceptron):

- The perceptron (figure 5.1) is a **feedforward network** containing a layer of **scanner neurons** (a retina that is used only for data **acquisition** and which has fixed-weighted connections with the first neuron layer (input layer)). The fixed-weight layer is followed by at least one trainable weight layer. **One neuron layer** is completely linked with the following layer. The first layer of the perceptron consists of the input neurons. corresponding to our convention. **The fixed-weight layer will no longer be taken into account in this course.**

Supervised Learning Network Paradigms

The perceptron

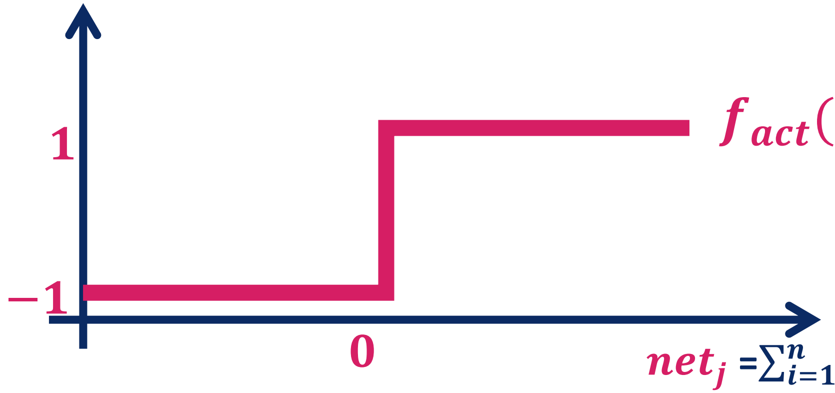


Figure(5.1): Illustration of perceptron with indicated (left side) and without indicated(right side) fixed-weight layer using the defined designs of the functional descriptions for neurons.

Supervised Learning Network Paradigms

The perceptron

- Here we initially use a binary perceptron with every output neuron having exactly two possible output values (e.g. {0, 1} or {−1, 1}). Thus, a binary threshold function is used as activation function, depending on the threshold value Θ of the output neuron.



$$f_{act}(net_j) = \left\{ \begin{matrix} value\,, net_j > \Theta \\ x, \qquad net_j \leq \Theta \end{matrix} \right\},$$

$$net_j = \sum_{i=1}^{n} w_i \, x_i + b$$

Figure(3.4): Illustration of binary threshold function

❑ Algorithm 5.1: Perceptron Learning(tr,y)

Input: tr set of training patterns,

Output: y

```
While $\exists \in tr\ \&error\ \ too\ large$ do
        Input $p$ into the network and calculate the output
        For all output neurons $\Omega$ do
                If $y_\Omega = \hat{y}_\Omega$ then
                        Output is good enough and no correction of weights
                Else
                    If $\hat{y}_\Omega = 0$ then
                        For all input neurons $i$ do
                            $w_{i,\Omega} := w_{i,\Omega} + o_i \{increase\ weight\ twords\ \Omega\ by o_i\}$
                        End for
                    End IF
                If $\hat{y}_\Omega = 1$ then
                    For all input neurons $i$ do
                        $w_{i,\Omega} := w_{i,\Omega} - o_i \{decrease\ weight\ twords\ \Omega\ by o_i\}$
                    End for
                End If
            End If
        End For
End While
```

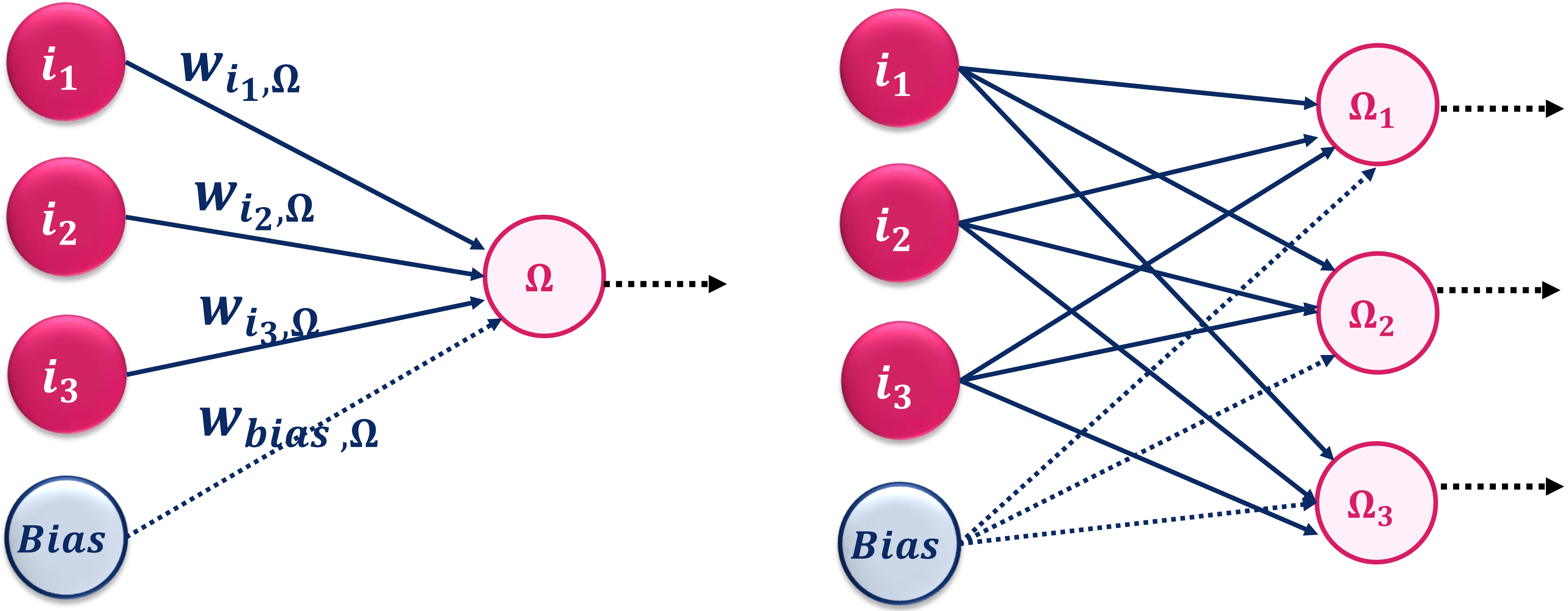
Supervised Learning Network Paradigms

Single Layer perceptron

- **The singlelayer perceptron provides only one trainable weight layer:**
  - Here, connections with **trainable weights** go from the input layer to an output neuron Ω, which returns the information whether the pattern entered at the **input neurons was recognized or not**. Thus, a singlelayer perception has **only one level of trainable weights** (Figure 5.2).

Supervised Learning Network Paradigms

Single Layer perceptron



Figure(5.2): Left side: illustration of a single layer perceptron with four input neurons and one output neuron. Right side: illustration of a single layer perceptron with four input neurons and three output neuron.

Supervised Learning Network Paradigms

Single Layer perceptron

❑ Definition 4.22 (Single Layer Perceptron):

- A **single layer** perceptron (**SLP**) is a perceptron having **only one layer of variable weights** and **one layer of output neurons Ω**. The technical view of an SLP is shown in figure 5.2.

Supervised Learning Network Paradigms

Single Layer perceptron

❑ More about Single Layer Perceptron:

- The **delta rule** only applies for Single Layer perceptron (SLP), since the formula is always related to the **teaching input**, and there is no teaching input for the inner processing layers of neurons.
- A SLP is only capable of representing **linearly separable data**.
- Today, used as a synonym for a **single-layer, feed-forward network.**

Supervised Learning Network Paradigms

Single Layer perceptron

❑ More about Single Layer Perceptron:

- There are many versions of SLP one of these versions is **Classical Single Layer Perceptron (also called Single layer perceptron using delta rule)** which used the delta rule for learning.
- **Classical Single Layer Perceptron is used** the **delta rule as** a gradient based learning strategy for SLPs

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Gradient Descent input selection:

- The objective function measures the mismatch between the desired output for given inputs and the actual network output on the same inputs. We leave the step size (η) and stopping threshold unspecified. We have the following selections:
  - **Parameter space** P =$\mathbb{R}^n = \{(w_0, w_1, .., w_n) | each\, w_i\; is\; a\; real\; number\}$.
  - **Objective function**= $\frac{1}{2}(y - \hat{y})^2$ where **y** is the desired(i.e. actual) output and

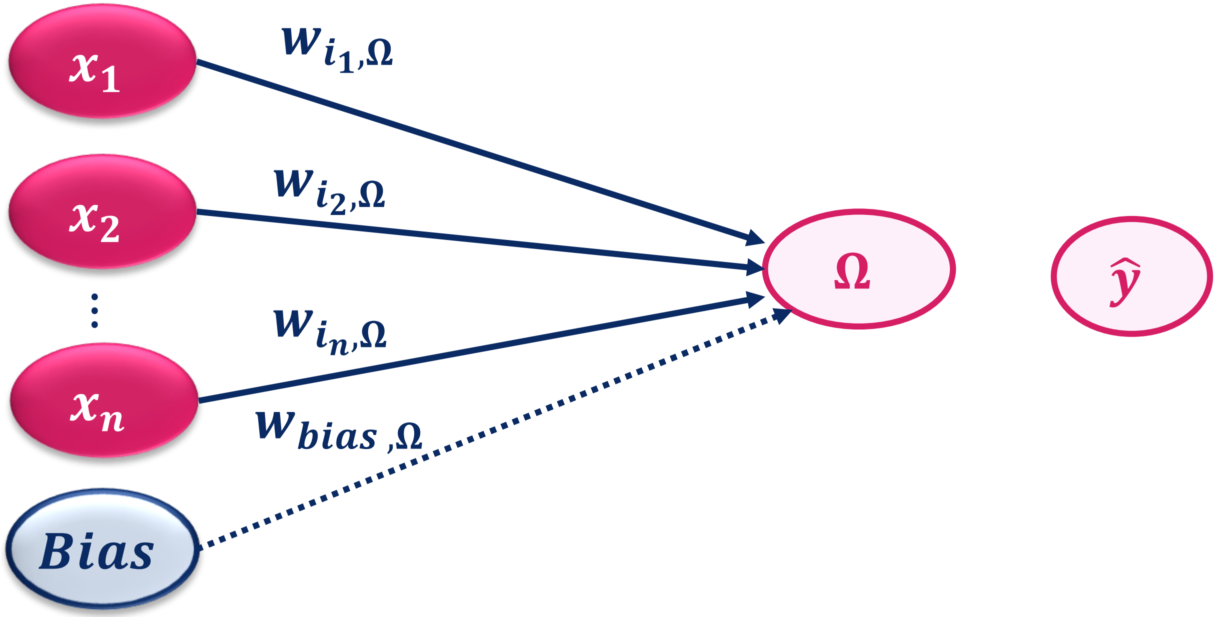
    $\hat{y}$ is the predicted output which is calculated by the $\hat{y}$ =$a(\sum_{i=0}^{n} w_i x_i)$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Training Single Layer Perceptron using Gradient Descent:

- Let's see how to apply the gradient descent search strategy to the machine learning task of training a single layer neural network.



Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Training Single Layer Perceptron using Gradient Descent:

- Now, we deviate from our **binary threshold value as activation function of algorithm 5.1** because at least for applying the **gradient descent (backpropagation of error)** we need, as you now and you will see that again, a **differentiable or even a semi-linear activation function**. **For the now following delta rule it is not always necessary but useful.** Compared with the other perceptron learning algorithm**, the delta rule has the advantage to be suitable for non-binary activation functions and, being far away from the learning target, to automatically learn faster.**

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Training Single Layer Perceptron using Gradient Descent:

- As already shown in lecture 4, the gradient descent algorithm calculates the gradient of an arbitrary but a finite-dimensional function i.e. cost function (here: of the **error function Err(W)**) and **moves down against the direction of the gradient until a minimum is reached.**
- **Err(W)** is defined on the set of all weights (which means Err(W) **represents cost function**) which we here regard as the vector **W**.

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Training Single Layer Perceptron using Gradient Descent:

- So we try to decrease or to minimize the error by simply tweaking the weights – thus **one receives information about how to change the weights** (the change in all weights is referred to as ΔW) by calculating **the gradient ∇Err(W)** of the error function **Err(W)**:

$$\boxed{\Delta W = -\eta\, \nabla Err(W)} \quad = \quad \boxed{\Delta W = -\eta\, \nabla C(W)}$$

$$=$$

$$\boxed{\Delta w_{i,\Omega} = -\eta \frac{\partial\, Err(W)}{\partial w_{i,\Omega}}} \quad = \quad \boxed{\Delta w_{i,\Omega} = -\eta \frac{\partial\, C(W)}{\partial w_{i,\Omega}}}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ **Define the loss function (error function for each instance i.e. Loss function):**

- **Assume that we used the offline learning.**
- **Now, we need to define our specific error function $Err_{\mathrm{p}}(W)$ (i.e. $loss\ function\ L(W)$).**
- It is not good if many results are far away from the desired ones; the error function should then provide large values – on the other hand, it is similarly bad if many results are close to the desired ones but there exists an extremely far outlying result. Here we are going to calculate the **squared difference of the components of the teaching input (i.e. actual value) y and the output vector $\hat{y}$, given the pattern p, and sum up these squares.**

$$l(W) = Err_{\mathrm{p}}(W) = \frac{1}{2}\sum_{\Omega \in O}(y_{p,\Omega} - \hat{y}_{p,\Omega})^2 \qquad \textbf{Eq.7.1}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Define the cost function:

- The **cost function** $Err\ (W)$ is defined as a summation of the **specific errors** $Err_{\mathrm{p}}(W)$ of all patterns p then yields the definition of the error **Err** and therefore the definition of the error function Err(W):

$$C = Err(W) \ = \sum_{\mathrm{p} \in P} Err_{p}(W)$$

$$C = Err\ (W) = \frac{1}{2} \sum_{\mathrm{p} \in p} \left(\sum_{\Omega \in O} (y_{p,\Omega} - \hat{y}_{p,\Omega}\ )^{2} \right)$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- Now we want to continue deriving the delta rule for **linear activation functions**.
- We have already discussed that we tweak the individual weights $\boldsymbol{w}_{\mathbf{i},\boldsymbol{\Omega}}$ a bit and see how the error Err(W) is changing – which corresponds to the derivative of the error function Err(W) according to the very same weight $\boldsymbol{w}_{\mathbf{i},\boldsymbol{\Omega}}$ .This derivative corresponds to the sum of the derivatives of all specific errors Err according to this weight (since the total error Err(W) results from the sum of the specific errors):

$$\Delta w_{\mathrm{i},\Omega} = -\eta \frac{\partial Err(W)}{\partial w_{\mathrm{i},\Omega}}$$

$$\Delta w_{\mathrm{i},\Omega} = \sum_{\mathrm{p}\in P} -\eta \frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}} \qquad \textbf{Eq 7.2}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- Remember that the output of the NN is $\hat{y} = a(b + \sum_{i=0}^{n} x_i \cdot w_i)$

$$C = Err\ (W)\ and\ L = Err_{\mathrm{p}}(W)$$

- We want to calculate the **derivatives** of **Eq 7.2** and due to the nested function we can apply the **chain rule to factorize** the derivatives $\frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}}$ in equation **Eq.7.2.**

$$\frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}} = \frac{\partial Err_p(W)}{\partial \hat{y}_{\mathrm{i},\Omega}} \cdot \frac{\partial \hat{y}_{p,\Omega}}{\partial w_{\mathrm{i},\Omega}} \qquad \textbf{Eq. 7.3}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- Let us take a look at the first multiplication factor of the equation **Eq. 7.3** which represents the derivative of the specific error $\boldsymbol{Err_p(W)}$with an output $\boldsymbol{\hat{y}_{p,\Omega}}$. The examination of $\boldsymbol{Err_p(W)}$ in equation **Eq.7.1** clearly shows that this change is exactly the difference between the **teaching input y and the output vector $\boldsymbol{\hat{y}}$** $(y_{p,\Omega} - \hat{y}_{p,\Omega})$. Thus we can replace one by the other. This difference is also called delta term **δ** (which is the reason for the name delta rule):

$$\frac{\partial Err_p(W)}{\partial w_{i,\Omega}} = -(y_{p,\Omega} - \hat{y}_{p,\Omega}) \cdot \frac{\partial \hat{y}_{p,\Omega}}{\partial w_{i,\Omega}}$$

$$= -\delta_{p,\Omega} \cdot \frac{\partial \hat{y}_{p,\Omega}}{\partial w_{i,\Omega}}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- The second multiplication factor of the equation **Eq. 7.3** which represents the derivative of the predicted output $\hat{y}_{p,\Omega}$ to the pattern **p** of the neuron **Ω** according to the weight $w_{\mathbf{i},\mathbf{\Omega}}$. **So how does $\hat{y}_{p,\Omega}$ change when the weight from i to Ω is changed? Due to the requirement at the beginning of the derivation, we only have a linear activation function a, therefore we can just as well look at the change of the network input when $w_{\mathbf{i},\mathbf{\Omega}}$ is changing:**

$$\frac{\partial Err_{p}(W)}{\partial w_{\mathbf{i},\mathbf{\Omega}}} = -\delta_{p,\mathbf{\Omega}} \cdot \frac{\partial \sum_{i=0}^{n}(o_{p,\mathbf{i}} \cdot w_{\mathbf{i},\mathbf{\Omega}})}{\partial w_{\mathbf{i},\mathbf{\Omega}}},$$

where $o_{p,\mathbf{i}}$ is the output of the predecessor neuron ***i*** .

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- The resulting derivative $\frac{\partial \sum_{i=0}^{n}(o_{p,\mathrm{i}} \cdot w_{\mathrm{i},\Omega})}{\partial w_{\mathrm{i},\Omega}}$ can now be simplified: The function $\sum_{i=0}^{n}\left(o_{p,\mathrm{i}} \cdot w_{\mathrm{i},\Omega}\right)$ to be derivative consists of many summands, and only the summand $o_{p,\mathrm{i}} \cdot w_{\mathrm{i},\Omega}$contains the variable $w_{\mathrm{i},\Omega}$; according to which we derive. Thus, $\frac{\partial \sum_{i=0}^{n}(o_{p,\mathrm{i}} \cdot w_{\mathrm{i},\Omega})}{\partial w_{\mathrm{i},\Omega}} = o_{p,\mathrm{i}}$ and therefore:

$$\boxed{\frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}} = -\delta_{p,\Omega} \cdot o_{p,\mathrm{i}}}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Computation of the Gradient Descent by calculate derivatives of the cost function:

- As mentioned before, the modification of the weight and the derivative of the error are calculated by

$$\Delta w_{\mathrm{i},\Omega} = \sum_{\mathrm{p} \in P} -\eta \frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}}$$

$$\frac{\partial Err_p(W)}{\partial w_{\mathrm{i},\Omega}} = -\delta_{p,\Omega} \cdot o_{p,\mathrm{i}}$$

- Finally, the modified rule of a weight $w_{\mathrm{i},\Omega}$ is

$$\Delta w_{\mathrm{i},\Omega} = \eta \sum_{\mathrm{p} \in P} \delta_{p,\Omega} \cdot o_{p,\mathrm{i}}$$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Definition 4.21 (Offline version of Delta rule *or* windrow-Hoff rule):

- As mentioned before, we used the offline learning, so that the result of **derivatives is the offline Delta rule for learning.**

$$\Delta w_{\mathbf{i},\boldsymbol{\Omega}} = \eta \sum_{\mathrm{p}\in P} \delta_{p,\boldsymbol{\Omega}} \cdot o_{p,\mathbf{i}}$$

where $\delta_{p,\Omega}$corresponds to the difference between $\boldsymbol{t}_{\mathbf{p},\boldsymbol{\Omega}}$ ***and*** $\hat{y}_{\mathbf{p},\boldsymbol{\Omega}}$; $\boldsymbol{o}_{\boldsymbol{p},\mathbf{i}}$is the output of the predecessor neuron ***i*** . As we know the SLP has only one layer so that the predecessor neuron is $\boldsymbol{x}_{\boldsymbol{p},\mathbf{i}}$.

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ More about Offline version of Delta rule *or* windrow-Hoff rule:

- We should ask our self the question of **how to add the errors of all patterns and how to learn them after all patterns have been represented.** Although this approach is mathematically correct, **the implementation is far more time-consuming** and, partially needs a lot of computational effort during training.

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Definition 4.21 (Online version of Delta rule *or* windrow-Hoff rule):

- That leads us to the **online learning version of the delta rule which** simply omits the summation and learning is realized immediately after the presentation of each pattern, this also simplifies the notation (which is no longer necessarily related to a pattern p)

$$\Delta w_{\mathrm{i},\Omega} = \eta\, \delta_{p,\Omega} \cdot o_{p,\mathrm{i}}$$

where δ_{Ω} corresponds to the difference betweeny_{Ω} ***and*** $\hat{y}_{\mathrm{i}}$; $o_{p,\mathrm{i}}$ is the output of the predecessor neuron ***i***.

- **When we apply the Delta rule for SLP,** the predecessor neuron $o_{p,\mathrm{i}}$ is become x_{i} the reason behind that the SLP has only one layer so there is no predecessor neurons except the input neuron x_{i} .

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

- **Algorithm 5.2: Single Layer Perceptron using delta learning rule (tr,y)**

```
Input: tr  set of training patterns
Output: y
1.  Initialization Initialize all weights by small random values (e.g. 0,-1,1,..) and select a suitable learning rate $\eta$
2.  While stopping condition is false (i.e. $\exists\delta_i = 0$ in the current epoch)do
3.        Activation: Input $p$ into the network and calculate output o for neuron i
4.              $net_i = \sum_{i=0}^{n} w_i x_i$
5.        Activation: Apply the  activation function on the output
6.              $\hat{y}_i = o_i = a_{iy_i}(net_i)$
7.        Calculate the delta  $term$ $(\delta)$
8.           $\delta_i = y_i - \hat{y}_i,$      $\delta = \left\{\begin{matrix} 0, & means\ coorect\ output \\ +1, & we\ need\ to\ increase\ weight\ twords\ \Omega\ by o_i \\ -1, & we\ need\ to\ decrese\ weight\ twords\ \Omega\ by o_i \end{matrix}\right\}$
9.        Learning:
10.       If $\delta \neq 0$
11.           For all output neurons $\Omega$ do
12.                 $w_{\Omega,new,} := \Delta w_{\Omega,},$ where $\eta\delta_{p,\Omega} \cdot o_{p,i}$
13.           $w_{\Omega,old,}$ +End for
14.       End If
15. Test stopping condition: If no weights changed in step 2, stop; else continue
16. End While
```

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Example:

- Assume that you have a Single layer perceptron that is used to solve (learn) the following problem by using Delta learning rule:
- The bias is 1

x_1	x_2	x_3	target
0	0	1	0
1	1	1	1
1	0	1	1
0	1	1	0

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Example:

- Use a Single layer perceptron that learns by using the online delta rule to solve (learn) the following problem :
- The bias is 1

x_1	x_2	x_3	target
0	0	1	0
1	1	1	1
1	0	1	1
0	1	1	0

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ Solution:

- Assume that, for simplicity during calculation the learning rate $\eta = 1$
- Initialize all weights by 0
- The activation function is

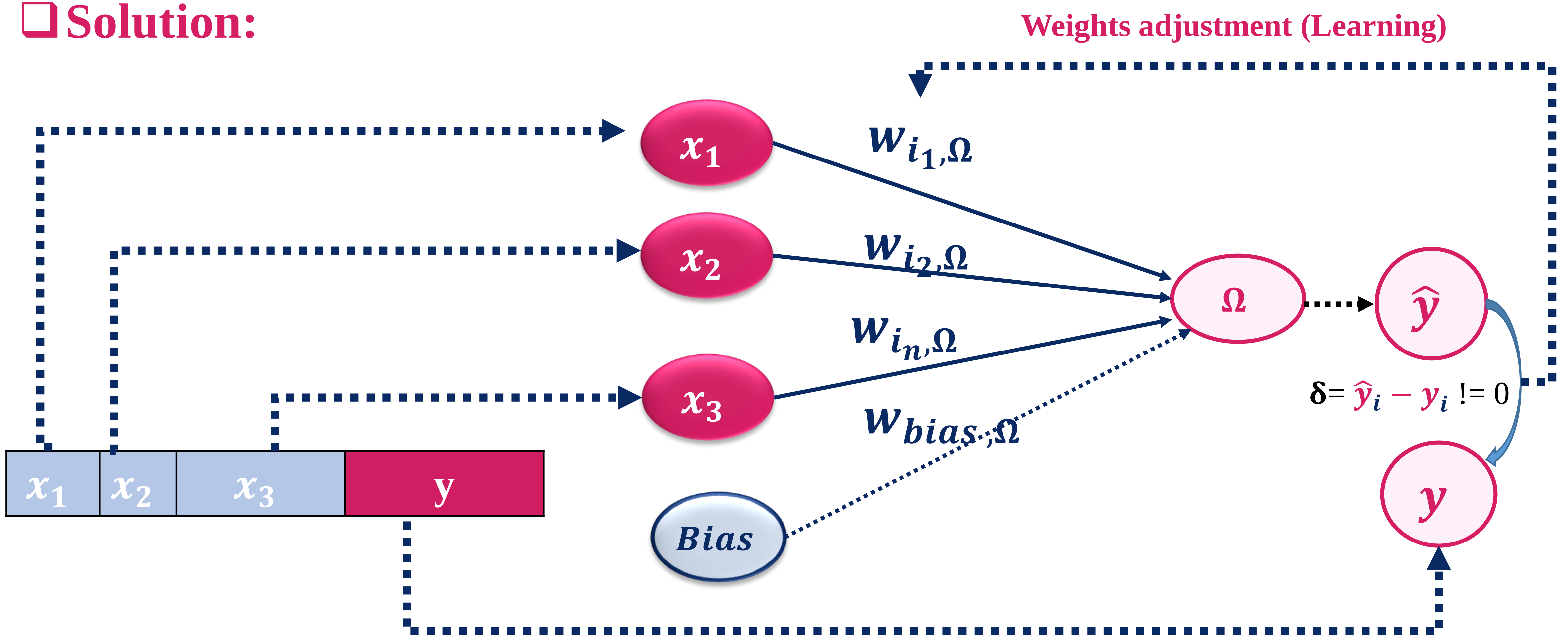
$$a(o_i) = \begin{cases} 1\,,\ if\ o_i > \theta \\ 0, otherwise \end{cases}$$

- Thresold is $\theta = 0$.

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

❑ **Solution:**



Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

w	x	t_i	o_i	$\hat{y}_i$	δ	$w_{i,new}$
0,0,0,0	0 , 0 , 1	0	o_1= (0×0)+(0×0)+(0×1)+0 o_1 = 0	0	0	No change
0,0,0,0	1 , 1 , 1	1	o_2= (0×1)+(0× 1)+(0×1)+0 o_2 = 0	0	1	1,1,1,1
1,1,1,1	1 , 0 , 1	1	o_2= (1×1)+(1× 0)+(1×1) +1 o_2 = 3	1	0	No change
1,1,1,1	0 , 1 , 1	0	o_2= (1×0)+(1× 1)+ (1×1) + 1 o_2 = 3	1	-1	1,0,0,0

Remimber*: *Delta rule is $\Delta w_{i,\Omega} = \eta\, \delta o_i$

Supervised Learning Network Paradigms

The delta rule as a gradient based learning strategy for SLPs

w	x	y_i	o_i	a_i	δ	w_{new}
1,0,0,0	0 , 0 , 1	0	o_1= (1×0)+(0×0)+(0×1)+0 o_1 = 0`	0	0	No change
1,0,0,0	1 , 1 , 1	1	o_2= (1×1)+(0× 1)+(0×1)+0 o_2 = 1	1	0	No change
1,0,0,0	1 , 0 , 1	1	o_2= (1×1)+(0× 0)+(0×1) +0 o_2 = 1	1	0	No change
1,0,0,0	0 , 1 , 1	0	o_2= (1×0)+(0× 1)+ (0×1) + 0 o_2 = 0	0	0	No change

Remimber*: *Delta rule is $\Delta w_{i,\Omega} = \eta\,\delta o_i$

Supervised Learning Network Paradigms

Multi-Layer perceptron

❑ Multi-Layer Perceptron:

- A multilayer perceptron contains **more trainable weight layers.**
- Multilayer perceptron (MLP) refers to a perceptron with two or more trainable weight layers.
- MLP is more powerful than an SLP. As we know, a singlelayer perceptron can divide the input space by means of a hyperplane (in a two-dimensional input space by means of a straight line).

Supervised Learning Network Paradigms

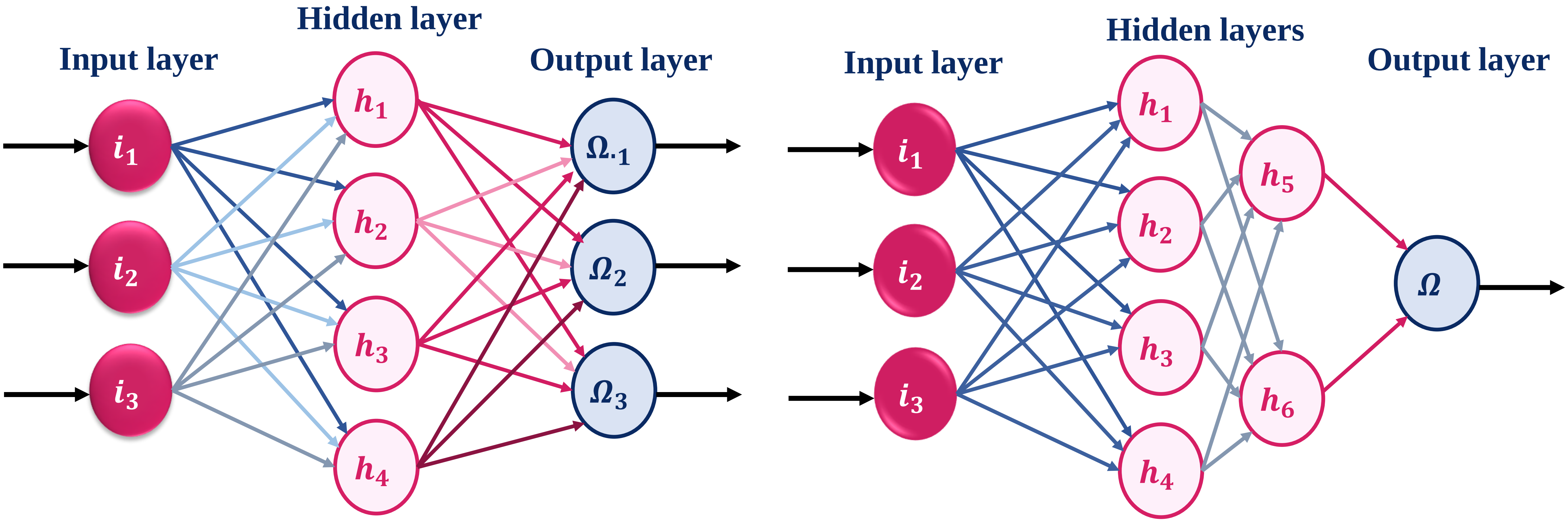
Multi-Layer perceptron

❑ Definition 5.1(Multi-Layer Perceptron):

- Perceptron **with more than one layer of variably weighted connections** are referred to as multilayer perceptron (MLP). An n-layer or n-stage perceptron has thereby exactly n variable weight layers and n+1 neuron layers (the retina is disregarded here) with neuron layer 1 being the input layer. **The MLP uses backpropagation algorithm for training process. The technical view of an MLP is shown in figure 5.3.**

Supervised Learning Network Paradigms

Multi-Layer perceptron



Figure(5.3): Left side: illustration of a single layer perceptron with four input neurons and one output neuron. Right side: illustration of a single layer perceptron with four input neurons and three output neuron.

Supervised Learning Network Paradigms

Multi-Layer perceptron

❑ Characteristics of MLP architecture:

- No connections within a layer
- No direct connections between input and output layers
- Fully connected amoung layers
- Often more than 3 layers (i.e. input, hidden and output layers)
- Number of **output units need not equal number of input units**
- Number of hidden units per layer can be **more or less than input or output units**
- They must have **at least one hidden layer**
- Hidden units must be **non-linear units (usually with sigmoid activation functions)**

Supervised Learning Network Paradigms

Hidden layers

- There are no concrete guidelines for the determination of the number of **hidden layer units.**
- Often obtained by experimentation, or by using optimizing techniques such as swarm optimization algorithms e.g. whale optimization algorithm.
- It is possible to remove hidden units that are not participating in the learning process by examining the weight values on the hidden units periodically as the network trains, on some units the weight values change very little, and these units may be removed from the network.

Assignments

❑ Assignment (5.1)

- Train a SLP to classify the following training set:
- Assume that the learning rate η =1, bias is -1 and initial weights (0,1,-1).
- Hint: the training process will stop after 10 epoch.

x_1	x_2	target
4	5	T
6	1	T
4	1	F
1	2	F

Any Questions!?

Thank you

References

- Kriesel, David. "A Brief Introduction to Neural Networks. 2007." URL http://www. dkriesel. com (2007).
- Sergio A. Alvarez . "CS345, Machine Learning Training Perceptrons using Gradient Descent Search" http://www.cs.bc.edu/~alvarez/ML/gradientSearch.pdf.