EE496 : COMPUTATIONAL INTELLIGENCE
NN02 : GENERAL ARTIFICIAL NEURAL NETWORKS

UGUR HALICI
METU: Department of Electrical and Electronics Engineering (EEE)
METU-Hacettepe U: Neuroscience and Neurotechnology (NSNT)

Basic graph theoretic notions

A (directed) **graph** is a pair $G = (V, E)$ consisting of a (finite) set V of **nodes** or **vertices** and a (finite) set $E \subseteq V$ x V of **edges**.

We call an edge $\mathrm{e} = (\mathrm{u}, \mathrm{v}) \in \mathrm{E}$ **directed** from node u to node v.

Let $\mathrm{G} = (\mathrm{V},\mathrm{E})$ be a (directed) graph and $\mathrm{u} \in \mathrm{V}$ a node. Then the nodes of the set

$$\mathrm{pred}(\mathrm{u}) = \{\mathrm{v} \in \mathrm{V} \mid (\mathrm{v}, \mathrm{u}) \in \mathrm{E}\}$$

are called the **predecessors** of the node u

and the nodes of the set

$$\mathrm{succ}(\mathrm{u}) = \{\mathrm{v} \in \mathrm{V} \mid (\mathrm{u}, \mathrm{v}) \in \mathrm{E}\}$$

are called the **successors** of the node u.

General definition of a neural network

An (artificial) **neural network** is a (directed) graph $G = (U,C)$,
whose nodes $u \in U$ are called **neurons** or units and
whose edges $c \in C$ are called **connections**.

The set U of nodes is partitioned into

- the set U_{in} of **input neurons**,
- the set U_{out} of **output neurons**, and
- the set U_{hidden} of **hidden neurons**.

It is

$$U = U_{in} \cup U_{out} \cup U_{hidden},$$

$$U_{in} \neq \varnothing,\ U_{out} \neq \varnothing,\ U_{hidden} \cap (U_{in} \cup U_{out}) = \varnothing.$$

General definition of a neural network

Each connection $(v, u) \in C$ possesses a **weight** w_{uv} (be careful on the notation, the order of subscripts may be different in different resources) and each neuron $u \in U$ possesses three (real-valued) state variables:

- the network input net_u,
- the activation act_u, and
- the output out_u.

Each input neuron $u \in U_{in}$ also possesses a fourth (real-valued) state variable:

- the external input ex_u.

(note: for feed forward NN act_u is same as net_u)

General definition of a neural network

Furthermore, each neuron $u \in U$ possesses three functions:

- the network input function
  $f^{(u)}_{net} : \mathbb{R}^{2|pred(u)|+\kappa 1(u)} \rightarrow \mathbb{R}$
- the activation function
  $f^{(u)}_{act} : \mathbb{R}^{\kappa 2(u)} \rightarrow \mathbb{R}$, and
- the output function
  $f^{(u)}_{out} : \mathbb{R} \rightarrow \mathbb{R}$,

which are used to compute the values of the state variables.

Types of (artificial) neural networks

- If the graph of a neural network is acyclic, it is called a **feed-forward** network.
- If the graph of a neural network contains cycles (backward connections), it is called a **recurrent network**.

Representation of the connection weights by a matrix

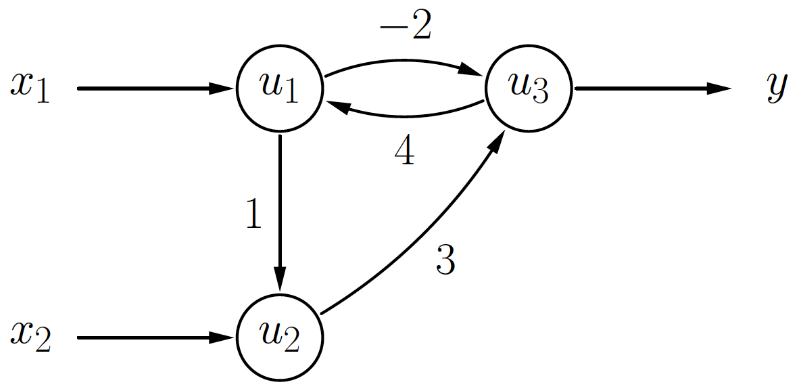
$$\begin{array}{cc} \begin{array}{cccc} u_1 & u_2 & \dots & u_r \end{array} & \\ \begin{pmatrix} w_{u_1u_1} & w_{u_1u_2} & \dots & w_{u_1u_r} \\ w_{u_2u_1} & w_{u_2u_2} & & w_{u_2u_r} \\ \vdots & & & \vdots \\ w_{u_ru_1} & w_{u_ru_2} & \dots & w_{u_ru_r} \end{pmatrix} & \begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_r \end{array} \end{array}$$

Note: row i corresponds to the weight vector of node i, here i=2 .

If the w_{uv} was used instead of w_{vu} to represent the connnection from unit u to v, then the column *i* would corresponds to the weight vector of node i,

General Neural Networks: Example

A simple recurrent neural network

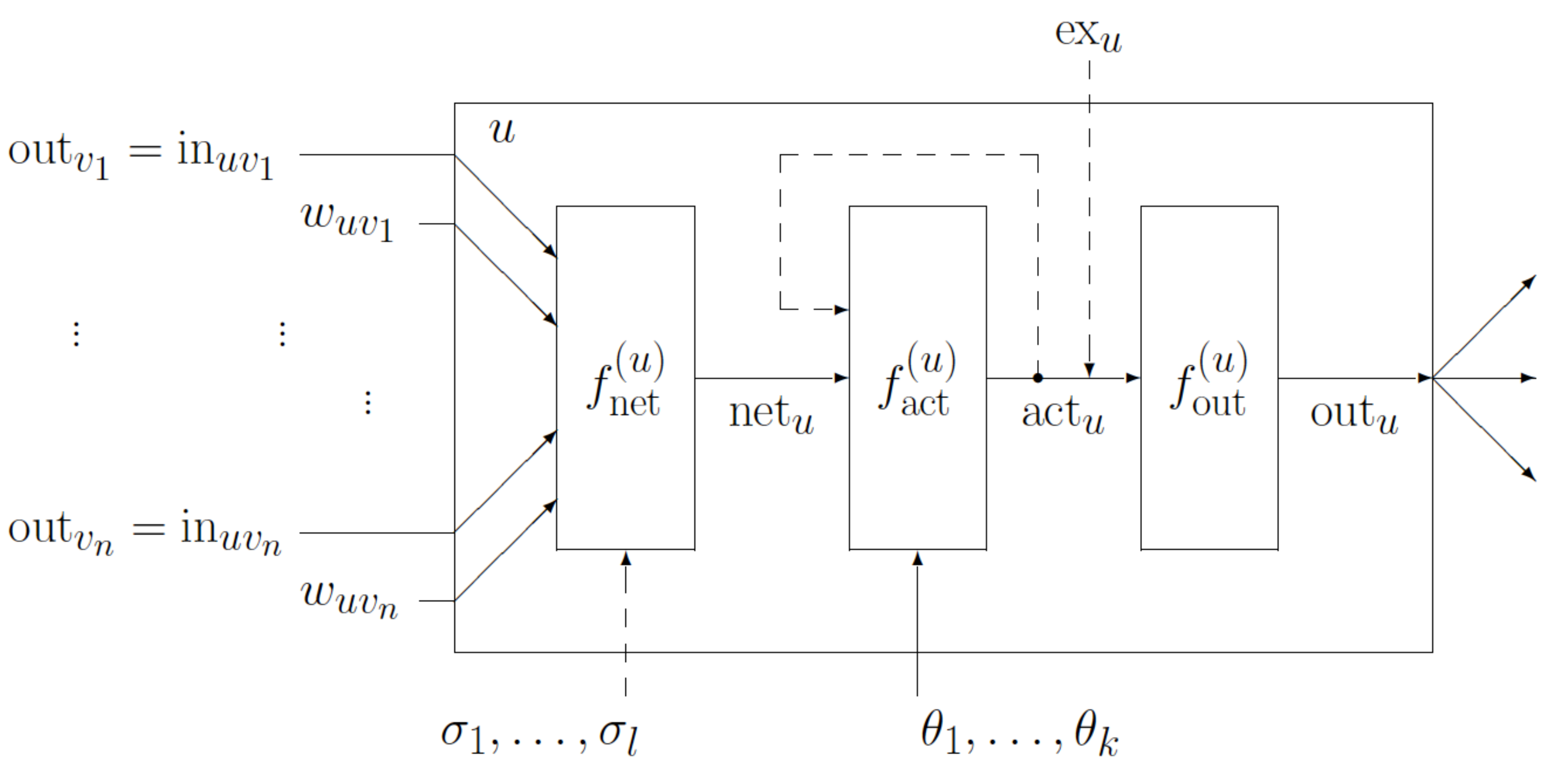


Weight matrix of this network

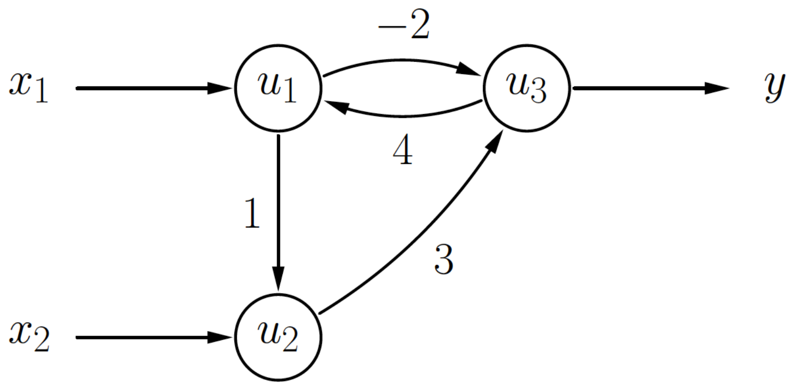
$$
\begin{array}{cc}
\begin{array}{ccc} \quad u_1 & u_2 & u_3 \end{array} & \\
\begin{pmatrix} 0 & 0 & 4 \\ 1 & 0 & 0 \\ -2 & 3 & 0 \end{pmatrix} & \begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}
\end{array}
$$

Structure of a Generalized Neuron

A generalized neuron is a simple numeric processor.



General Neural Networks: Example



$$f^{(u)}_{net}(\vec{\mathbf{w}}_u, \vec{\mathbf{in}}_u) = \sum_{v \in \mathrm{pred}(u)} w_{uv} in_{uv} = \sum_{v \in \mathrm{pred}(u)} w_{uv}\, out_v$$

$$f^{(u)}_{act}(net_u, \theta) = net_u$$

$$f^{(u)}_{out}(act_u) = \begin{cases} 1, & \text{if } act_u \geq \theta, \\ 0, & \text{otherwise} \end{cases}$$

Updating the activations of the neurons

	u_1	u_2	u_3	
initial state	**1**	**0**	**0**	input phase
$\mathrm{net}_{u3} = -2$	1	0	**0**	work phase
$\mathrm{net}_{u1} = 0$	**0**	0	0	
$\mathrm{net}_{u2} = 0$	0	**0**	0	
$\mathrm{net}_{u3} = 0$	0	0	**0**	
$\mathrm{net}_{u1} = 0$	**0**	0	0	
				converged

- Order in which the neurons are updated:

  $u_3, u_1, u_2, u_3, u_1, u_2, u_3, \ldots$

- **Input phase**: activations and outputs of the initial state (first row)
- The activation of the currently neuron (bold) is calculated by considering the other neurons and weights.
- A stable state with a unique output is reached.

Updating the activations of the neurons

	u_1	u_2	u_3	
initial state	**1**	**0**	**0**	input phase
$net_{u3} = -2$	1	0	**0**	work phase
$net_{u2} = 1$	1	**1**	0	
$net_{u1} = 0$	**0**	1	0	
$net_{u3} = 3$	0	1	**1**	
$net_{u2} = 0$	0	**0**	1	
$net_{u1} = 4$	**1**	0	1	
$net_{u3} = -2$	1	0	**0**	
				oscillates

- Order in which the neurons are updated:

  $u_3, u_2, u_1, u_3, u_2, u_1, u_3, \ldots$

- **Input phase**: activations and outputs of the initial state (first row)
- The activation of the currently neuron (bold) is calculated by considering the other neurons and weights.
- A stable state with a unique output is reached.

Definition of learning tasks for a neural network

A **fixed (i.e. supervised) learning task** L_{fixed} for a neural network with

- n input neurons, i.e. $U_{\text{in}} = \{u_1, \ldots, u_n\}$, and
- m output neurons, i.e. $U_{\text{out}} = \{v_1, \ldots, v_m\}$,

is a set of training patterns $L = (\vec{\imath}^{\,(l)}, \vec{o}^{\,(l)})$, each consisting of

- an input vector $\vec{\imath}^{\,(l)} = (\mathrm{ex}^{(l)}_{u1}, \ldots, \mathrm{ex}^{(l)}_{un})$ and
- an output vector $\vec{o}\,(l) = (o^{(l)}_{v1}, \ldots, o^{(l)}_{vm})$.

A fixed learning task is solved, if for all training patterns $L \in L_{\text{fixed}}$ the neural network computes from the external inputs contained in the input vector $\vec{\imath}^{\,(l)}$ of a training pattern l, the outputs contained in the corresponding output vector $\vec{o}^{\,(l)}$

Solving a fixed learning task: Error definition

- Measure how well a neural network solves a given fixed learning task.
- Compute differences between desired and actual outputs.
- Do not sum differences directly in order to avoid errors canceling each other.
- Square has favorable properties for deriving the adaptation rules.

$$e = \sum_{l \in L_{fixed}} e^{(l)} = \sum_{v \in U_{out}} e_v = \sum_{l \in L_{fixed}} \sum_{v \in U_{out}} e^{(l)}_v$$

i.e do summation for each pattern and for each output

where

$$e^{(l)}_v = (o^{(l)}_v - out^{(l)}_v)^2$$

i.e. square of the difference beteen desired and actual output

Definition of learning tasks for a neural network

A **free (i.e. unsupervised) learning task** L_{free} for a neural network with

- n input neurons, i.e. $U_{in} = \{u_1, \ldots, u_n\}$, and

is a set of training patterns $L = (\vec{\imath}^{\,(l)})$, each consisting of

- an input vector $\vec{\imath}^{\,(l)} = (ex^{(l)}_{u1}, \ldots, ex^{(l)}_{un})$

  i.e. no desired output

Properties:

- There is no desired output for the training patterns.
- Outputs can be chosen freely by the training method.
- Solution idea: **Similar inputs should lead to similar outputs.**
  (clustering of input vectors)

Normalization of the input vectors

In order to avoid unit and scaling problems

- Compute expected value and standard deviation for each input:

$$\mu_k = \frac{1}{|L|} \sum_{l \in L} \mathrm{ex}_{u_k}^{(l)} \qquad \text{and} \qquad \sigma_k = \sqrt{\frac{1}{|L|} \sum_{l \in L} \left(\mathrm{ex}_{u_k}^{(l)} - \mu_k \right)^2},$$

- Normalize the input vectors to
  - expected value 0 and
  - standard deviation 1:

$$\mathrm{ex}_{u_k}^{(l)(\mathrm{neu})} = \frac{\mathrm{ex}_{u_k}^{(l)(\mathrm{alt})} - \mu_k}{\sigma_k}$$

neu: new
alt: old
☺