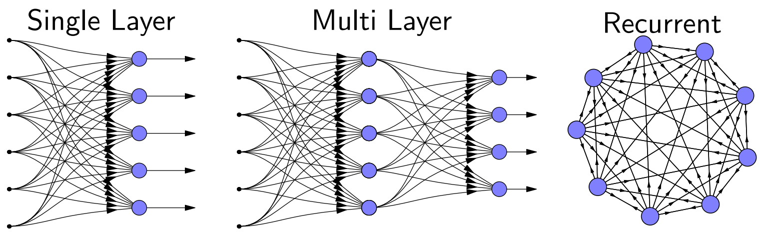
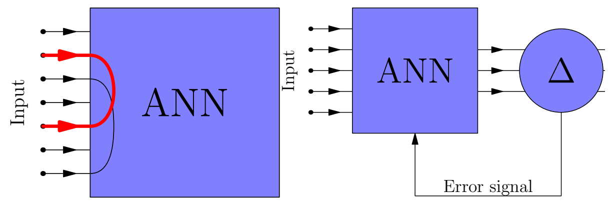
Linear Layered Networks

Connection Topologies



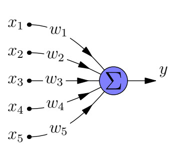
Learning Principles

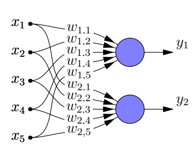
- **Coincidence Detection**
- **Error Correction**
- **Competitive Learning**



Linear Feed-Forward Networks

What can be computed by a linear network?





$$y = \vec{w}^T \cdot \vec{x}$$

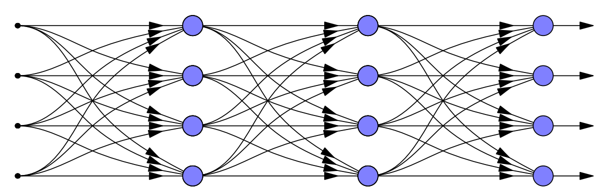
$$\vec{y} = W\vec{x}$$

$\vec{w}$ — Weight Vector

W — Weight Matrix

Linear Feed-Forward Networks

Cascaded Linear Networks



$$\vec{y} = W_3(W_2(W_1\vec{x})) = (W_3W_2W_1)\vec{x}$$

$$\text{Let } W = W_3W_2W_1 \quad \Rightarrow \quad \vec{y} = W\vec{x}$$

Still a linear mapping

Storing Mappings

The "program" resides in the weights

How do we find the right weights?

Learning ≈ Change the weights to achieve better performance

Hebbs Learning Hypothesis

Simultaneous activation of two neurons
strengthens the synaptic connection between them

Common interpretation:

$$\Delta w_{ij} = x_j y_i$$

Note! Outer Product

Storing Mappings

Storing a mapping using Hebbs rule

$$\vec{x}_1 \rightarrow \vec{y}_1 \qquad \vec{x}_2 \rightarrow \vec{y}_2 \qquad \vec{x}_3 \rightarrow \vec{y}_3 \qquad \cdots$$

Hebbs rule

$$\Delta w_{ij} = x_j y_i$$

Result

$$W = \sum_p \vec{y}_p \vec{x}_p^T$$

Correlation Memory

Storing Mappings

Retrieving a Memory Trace

$$W = \sum_p \vec{y}_p \vec{x}_p^T$$

$$\vec{x}_k \rightarrow ?$$

$$\vec{y}_{\mathrm{out}} = W\vec{x}_k = \sum_p (\vec{y}_p \vec{x}_p^T)\vec{x}_k = \sum_p \vec{y}_p (\vec{x}_p^T \vec{x}_k) =$$

$$= \vec{y}_k (\vec{x}_k^T \vec{x}_k) + \sum_{p \neq k} \vec{y}_p (\vec{x}_p^T \vec{x}_k) \approx \alpha \vec{y}_k$$

- Perfect memory if the patterns $\vec{x}_p$ are orthogonal