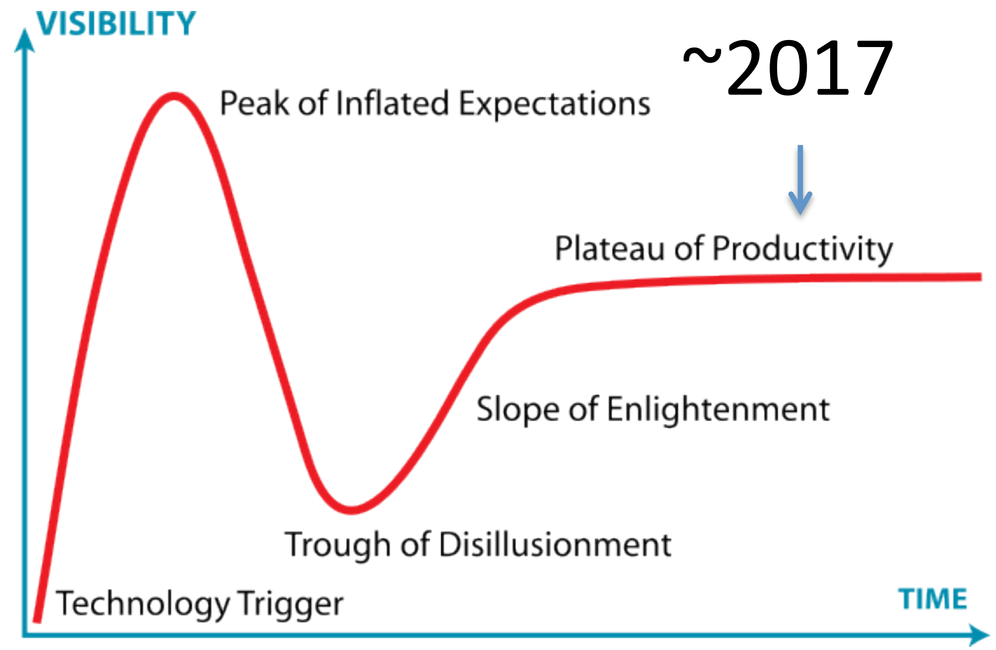
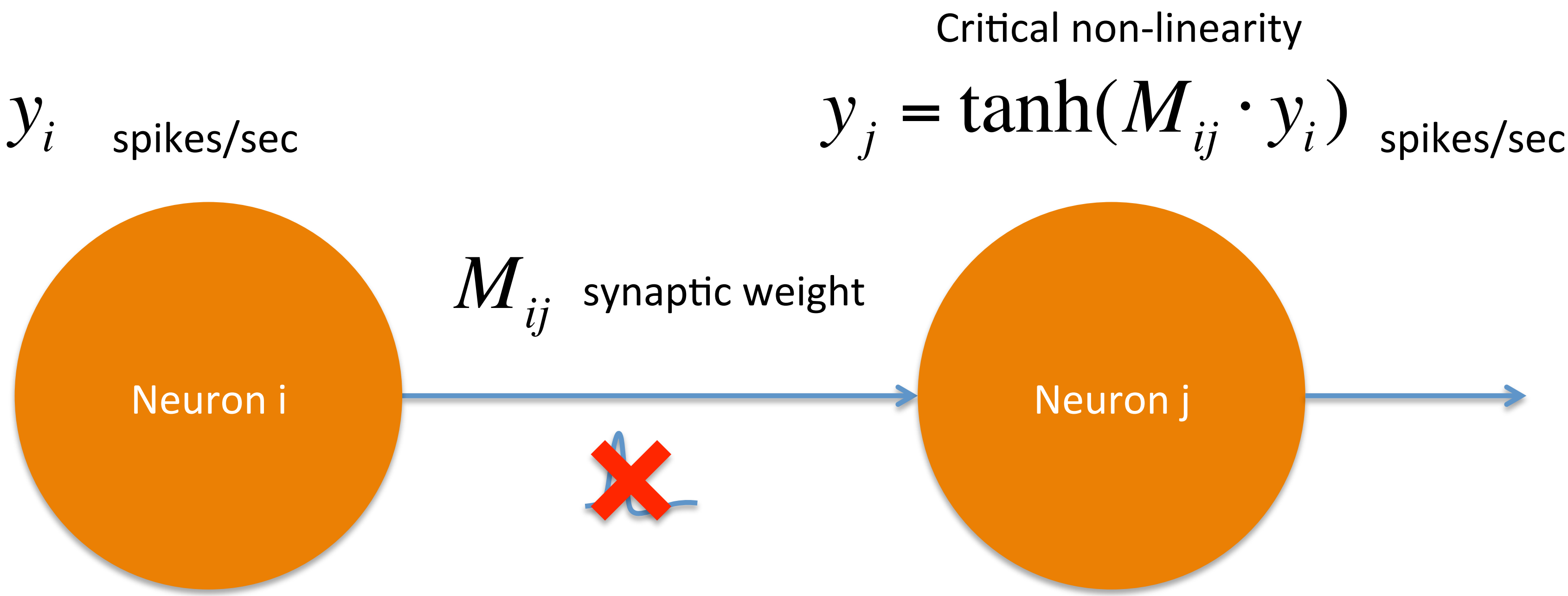
Early Neural Networks

Brief History of Neural Networks

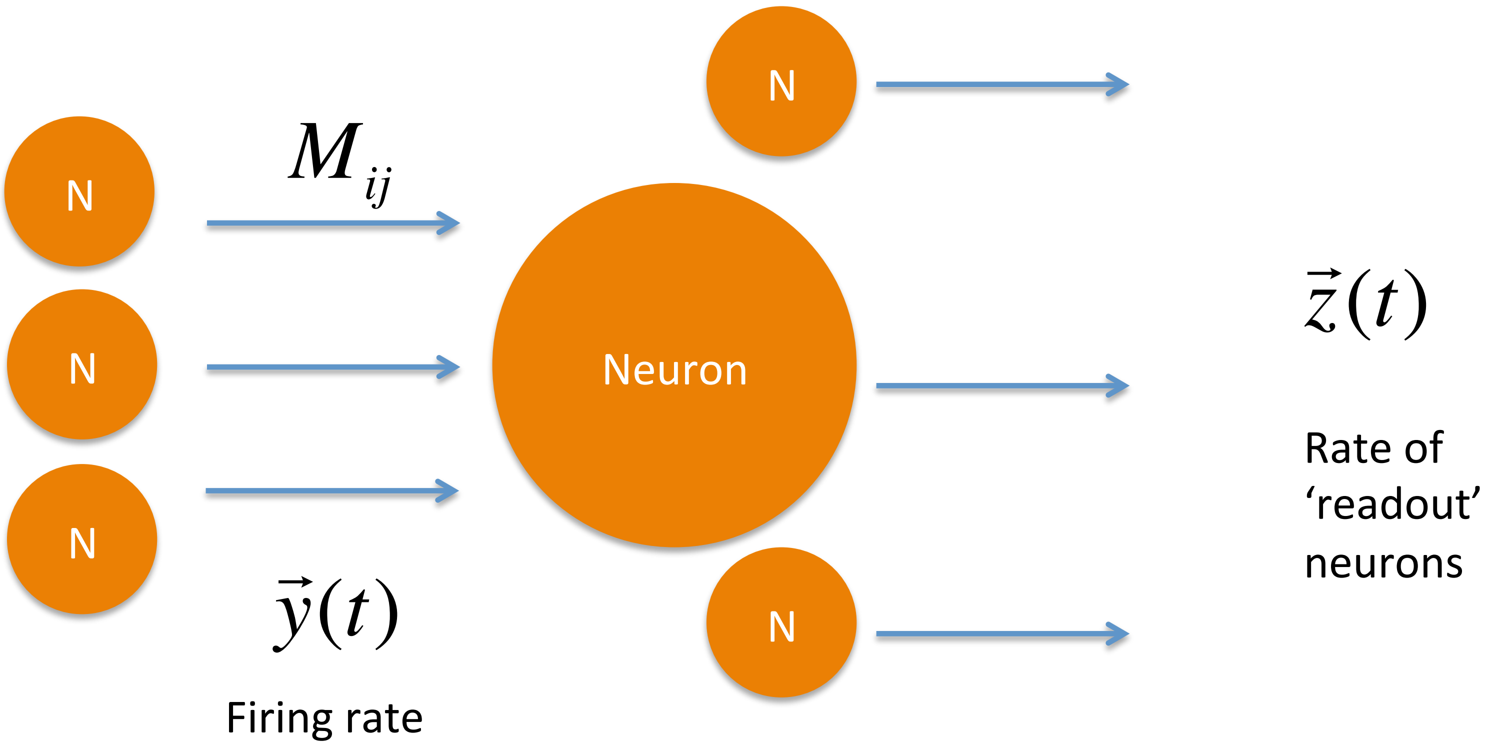
- Hype Curve



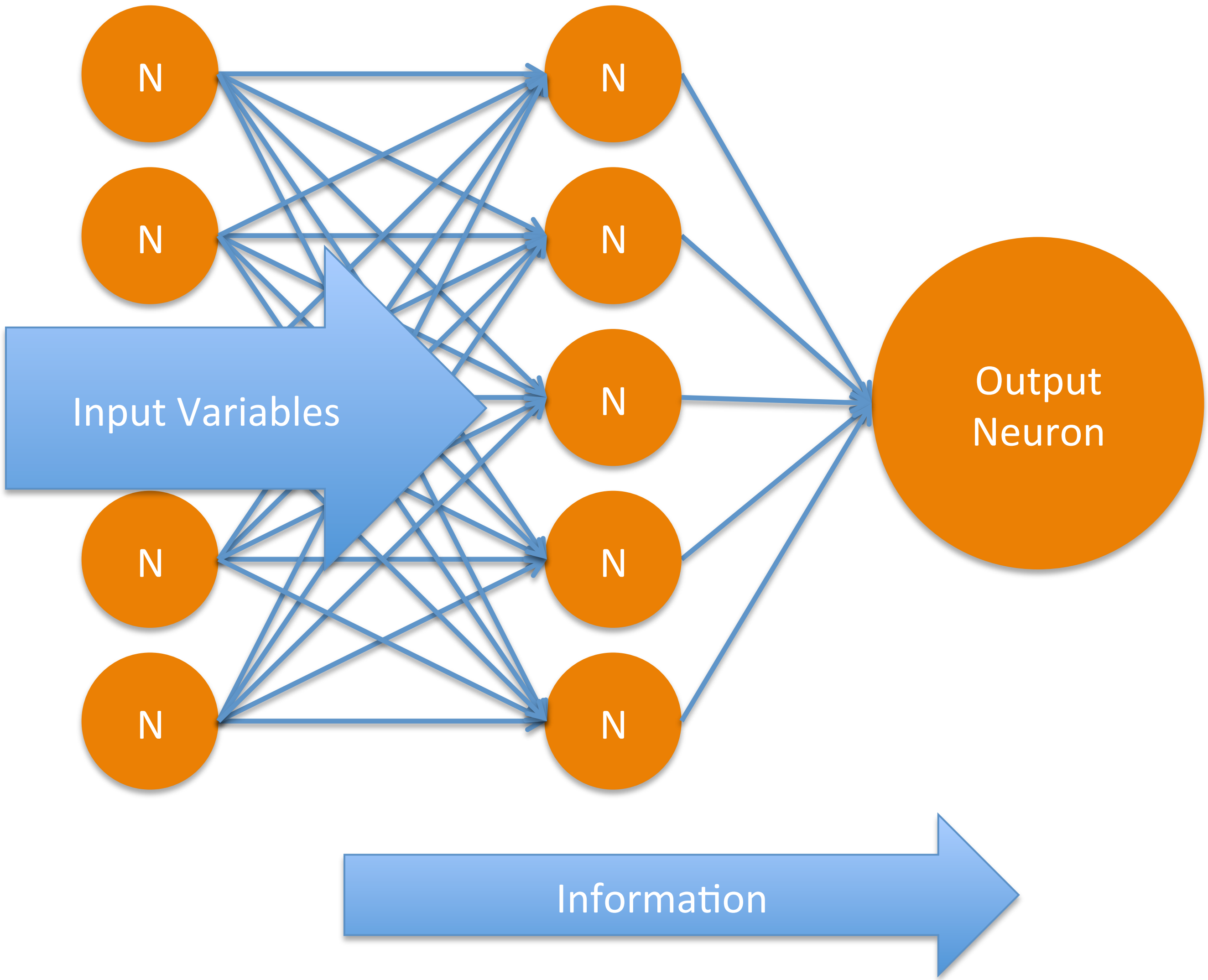
Firing Rate Model



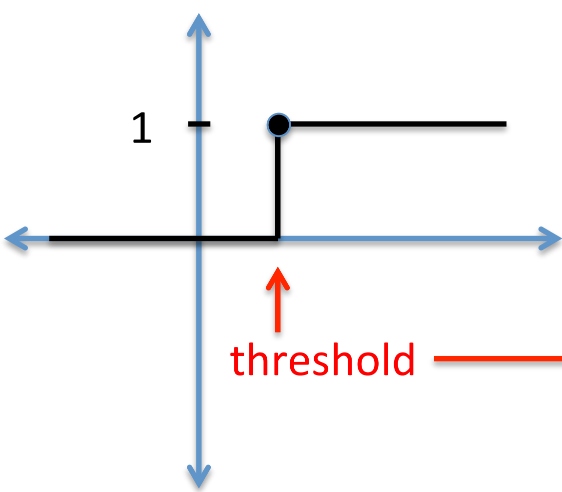
Neural Models that Compute

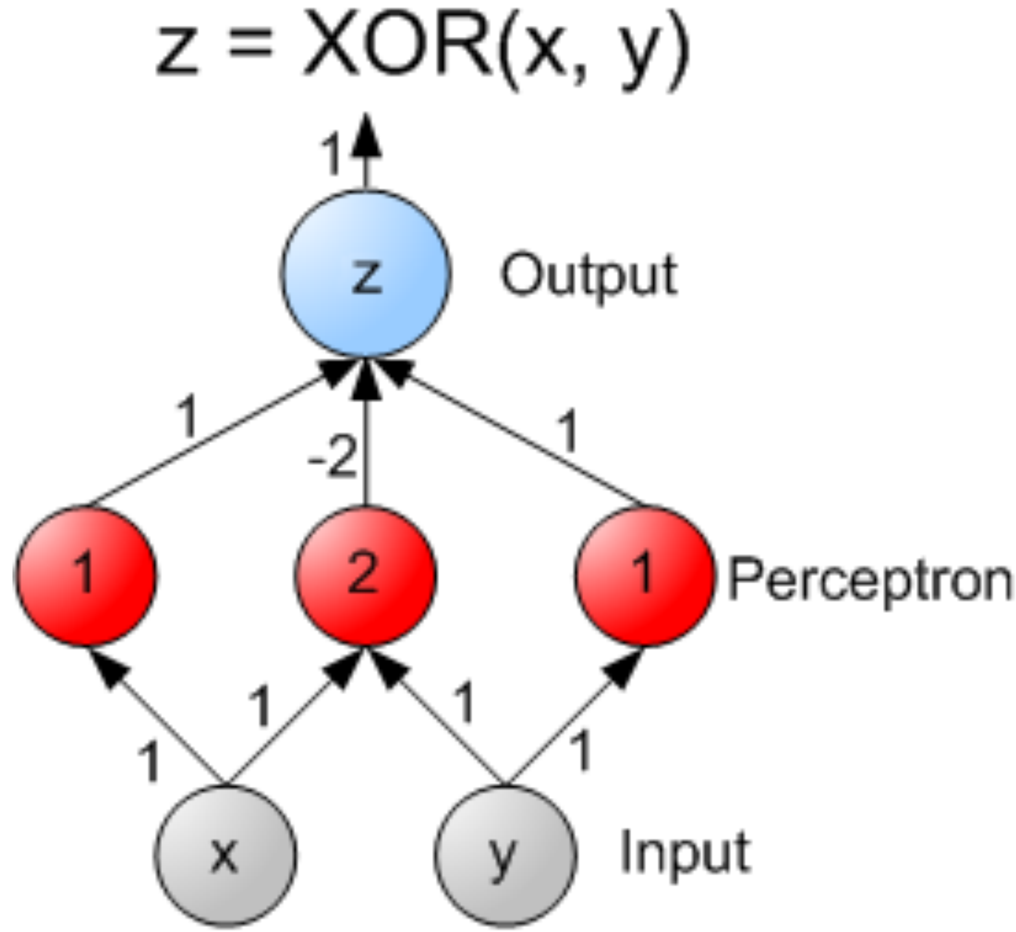


Feed Forward Neural Network



Example function





x=1,y=0, middle layer? z?

x=1,y=1, middle layer? z?

Back Propagation

- "Although we cannot claim that the back-propagation algorithm provides an optimal solution for all solvable problems, it has put to rest the pessimism about learning in multilayer machines that may have been inferred from the book by Minsky and Papert (1969)."
  – Haykin, *Neural Networks*, 1999

Back Propagation

Goal: $\vec{y}(t) \rightarrow f(t)$ (Arbitrary pattern generation)

- Very high dimensional space to search
- Minimize error function with respect to the goal output

$$E(t) = \frac{1}{2}\sum_{n=1}^{N}\left[y_i(t) - f_i(t)\right]^2$$

- Use gradient descent to change weights to minimize error $\dfrac{\partial E(t)}{\partial m_{ij}}$

Derivation of Back Propagation

- Starting with the total error on one timestep

$$\frac{\partial E}{\partial m_{ij}} = \frac{\partial E}{\partial e_j}\frac{\partial e_j}{\partial y_j}\frac{\partial y_j}{\partial v_j}\frac{\partial v_j}{\partial m_{ij}}$$

$$e_j$$

Error in neuron j

$$v_j = \sum_i m_{ij} y_i$$

Summed input to j

$$y_j = \varphi(v_j)$$

Nonlinearity of some kind

Derivation of Back Propagation

- Solve all the individual derivatives

$$\frac{\partial E}{\partial e_j} = \frac{\partial}{\partial e_j}\frac{1}{2}\sum_i e_i^2 = e_j$$

$$\frac{\partial e_j}{\partial y_j} = \frac{\partial}{\partial y_j}(correct_j - y_j) = -1$$

$$\frac{\partial y_j}{\partial v_j} = \phi'(v_j)$$

$$\frac{\partial v_j}{\partial m_{ij}} = \frac{\partial}{\partial m_{ij}}\sum_i m_{ij} y_i = y_i$$

Derivation of Back Propagation

- Fill in all the pieces

$$\frac{\partial E}{\partial m_{ij}} = -e_j \phi'(v_j) y_i$$

- Create rule for gradient descent

$$\Delta m_{ij} = -\alpha \frac{\partial E}{\partial m_{ij}} \quad = \alpha e_j y_i \varphi' \left(\sum_i m_{ij} y_i \right)$$

$$\alpha \cdot \partial_j \cdot y_i$$

Problem – Hidden Layers

- What is the error of a hidden layer neuron?

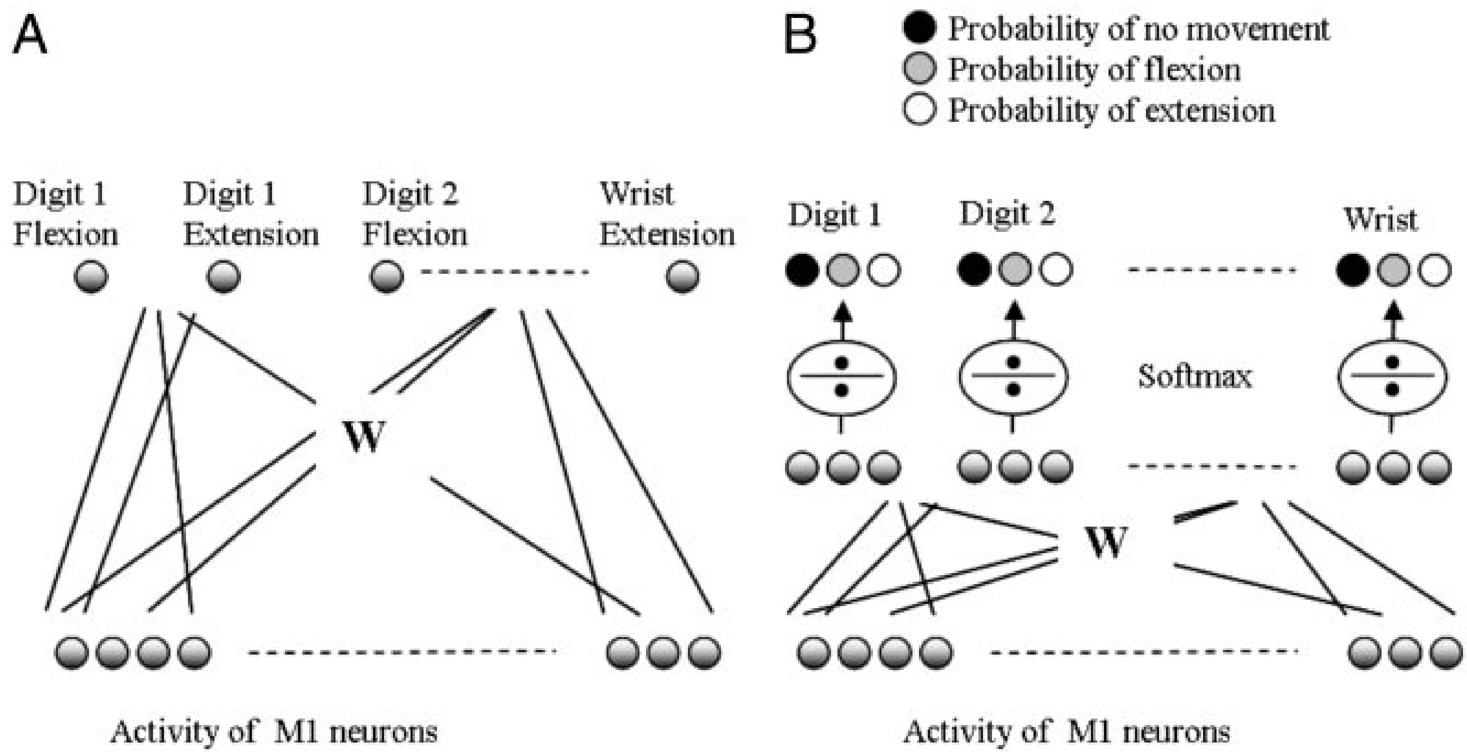
$$\delta_j = e_j \varphi'\left(\sum_i m_{ij} y_i\right)$$ (j is output neuron)

$$\Delta m_{ij} = \alpha y_i \delta_j$$

$$\delta_j = \varphi'\left(\sum_i m_{ij} y_i\right) \sum_k \delta_k m_{jk}$$

(j is hidden, k is output neuron)

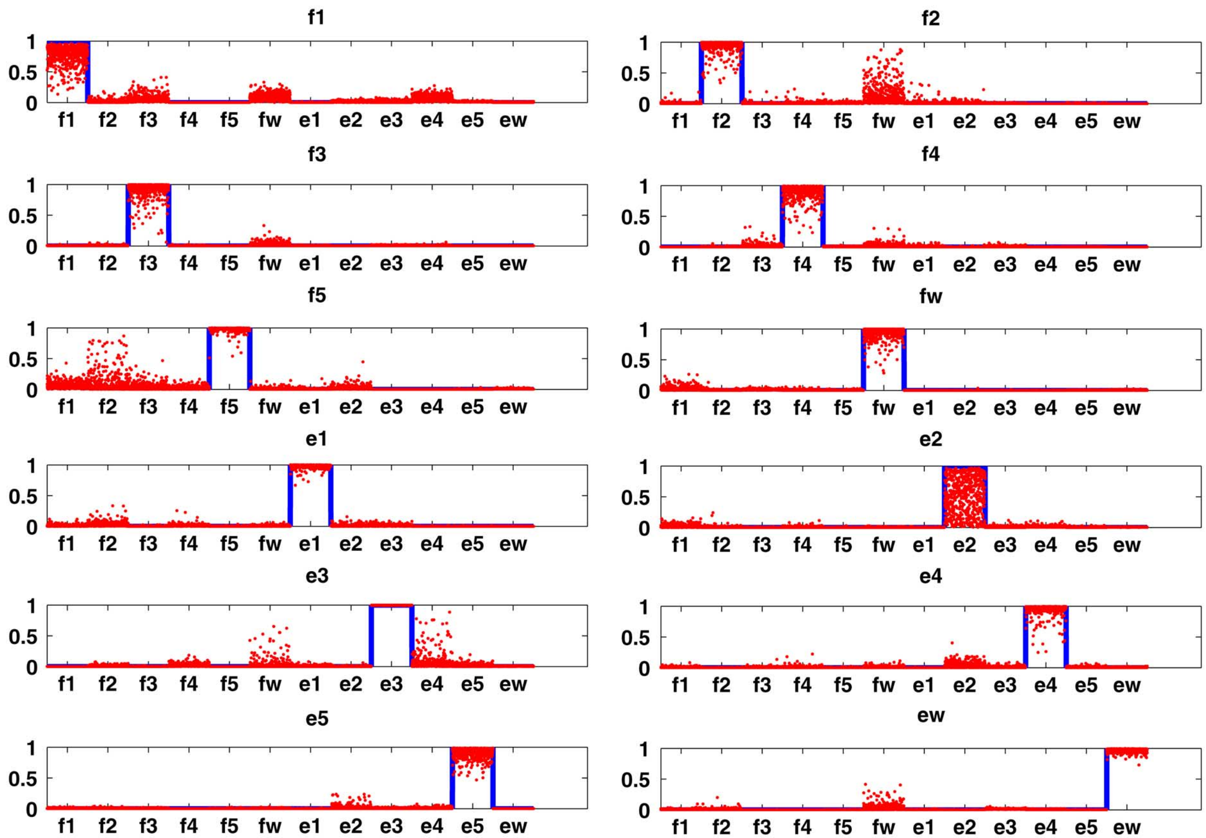
Feed-Forward Network Applications



$$y_i^k = \frac{1}{1 + \exp\left(-\beta \sum_{j=1}^{N} w_{ij} r_j^k + b_i\right)}$$

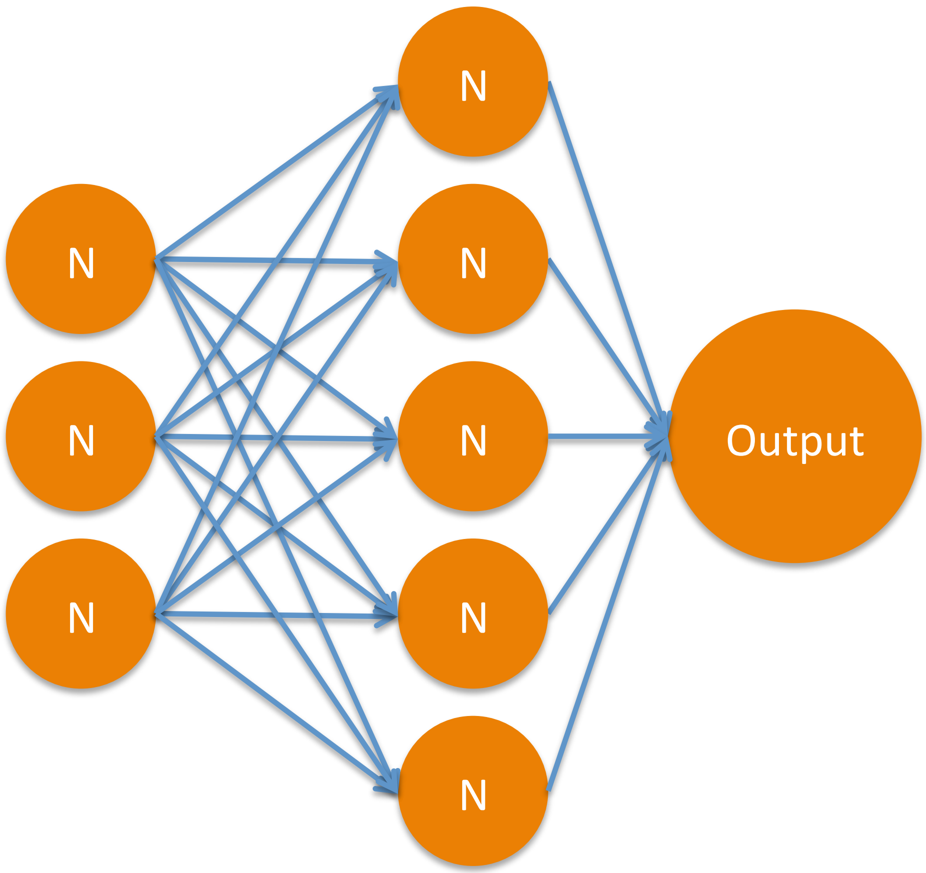
(Hamed et al., 2007)

Finger Classification Results

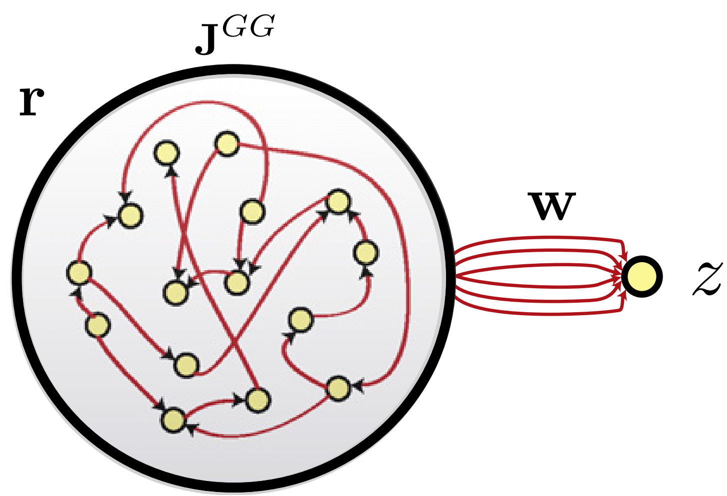


(Aggarwal et al., 2008)

Recurrent Networks



Non-Biological



Better

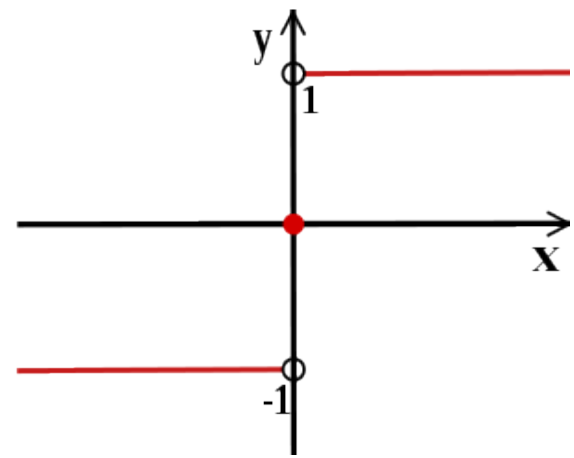
Hopfield Network

- Properties

$$y_i(t+\Delta t) = y_i' = \operatorname{sgn}(M_{ij} \cdot y_j(t))$$

$$M_{ij} = M_{ji} \qquad M_{ii} = 0$$

$\operatorname{sgn}(x)$



$$E = -\frac{1}{2}\sum_{i,j} M_{ij} y_i y_j$$

Call this the "Energy" of the network, will go down until a stable state is reached

Hopfield Derivation

- Calculate change in energy over time. First put in terms of a particular neuron index k

1) $$y_k \longrightarrow k \neq i, k \neq j \qquad\qquad E = -\frac{1}{2}\sum_{i,j} M_{ij} y_i y_j$$

2) $$E = -\frac{1}{2}\left(\begin{array}{c} M_{1k} y_1 y_k + M_{2k} y_2 y_k + \ldots \\ \ldots + M_{k1} y_k y_1 + M_{k2} y_k y_2 + \ldots \end{array} \right)$$

Hopfield Derivation

- Can write in terms of an i, j, and k. (Sum across all neurons indexed by i on the transmitting side, all neurons indexed by j on the receiving side

2) $$E = -\frac{1}{2}\left(\begin{array}{c} M_{1k} y_1 y_k + M_{2k} y_2 y_k + \dots \\ \dots + M_{k1} y_k y_1 + M_{k2} y_k y_2 + \dots \end{array} \right)$$

3) $$E = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k + \sum_j M_{kj} y_k y_j \right)$$

Hopfield Derivation

- Define prime operator to mean what happens when it passes through signum nonlinearity to get to the state for the next time step

$$E \rightarrow E' = E + \Delta E \qquad\qquad y_k' = \operatorname{sgn} \sum_j M_{jk} y_j = y_k(t + \Delta t)$$

- Find full expression for E' to see difference from E

$$E' = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k' + \sum_j M_{kj} y_k' y_j \right)$$

Hopfield Derivation

- Subtract E'-E to see how E changes each timestep

3) $$E = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k + \sum_j M_{kj} y_k y_j\right)$$

$$E' = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k' + \sum_j M_{kj} y_k' y_j\right)$$

4) $$E' - E = \Delta E = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k' + \sum_j M_{kj} y_k' y_j - \sum_i M_{ik} y_i y_k - \sum_j M_{kj} y_k y_j\right)$$

Hopfield derivation

- Simplify to get in terms of yk'-yk

4) $$E' - E = \Delta E = -\frac{1}{2}\left(\sum_i M_{ik} y_i y_k' + \sum_j M_{kj} y_k' y_j - \sum_i M_{ik} y_i y_k - \sum_j M_{kj} y_k y_j \right)$$

5) $$E' - E = \Delta E = -\frac{1}{2}\left(y_k' - y_k \right)\left(\sum_i M_{ik} y_i + \sum_j M_{kj} y_j \right)$$

Hopfield Derivation

- Take advantage of Mkj=Mjk

5)

$$E' - E = \Delta E = -\frac{1}{2}\left(y_k' - y_k\right)\left(\sum_i M_{ik} y_i + \sum_j M_{kj} y_j\right)$$

6)

$$\Delta E = -\frac{2}{2}\left(y_k' - y_k\right)\sum_j M_{jk} y_j$$

Hopfield Derivation

- Sum is now proportional to yk', has the same sign

6)

$$\Delta E = -\frac{2}{2}\left(y_k' - y_k\right)\sum_j M_{jk} y_j$$

$$y_k' = \operatorname{sgn}\sum_j M_{jk} y_j = y_k(t + \Delta t)$$

7) $\Delta E \propto -\left(y_k' - y_k\right) y_k'$

Hopfield Derivation

- Why we care: What happens to change in E every time step?

$$\Delta E \propto -\left(y_k' - y_k\right) y_k'$$

$$y_k' = y_k \rightarrow \Delta E = 0$$

$$y_k' = -1, y_k = 1 \rightarrow \Delta E < 0$$

$$y_k' = 1, y_k = -1 \rightarrow \Delta E < 0$$

- E goes down until no further neurons flip their state

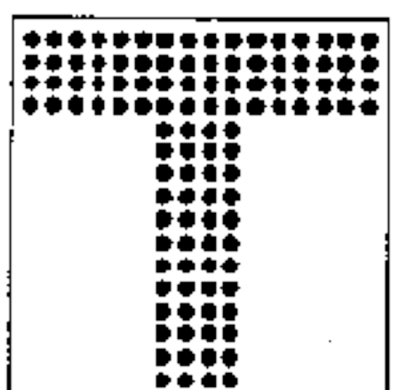
Hopfield "Attractor" Network

Desired stable state

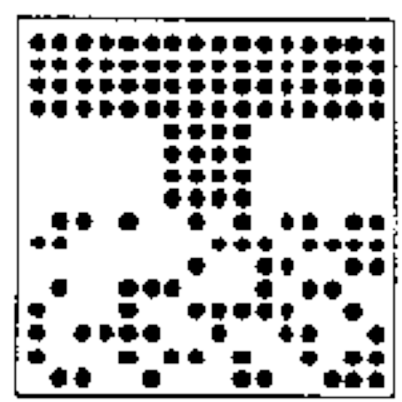
$$\vec{y} = (y_1, y_2, \dots y_N)$$

$$M_{ij} = \frac{y_i y_j}{N}$$ Stores one pattern

$$M_{ij} = \frac{y_i^1 y_j^1 + y_i^2 y_j^2}{N}$$ Store two patterns

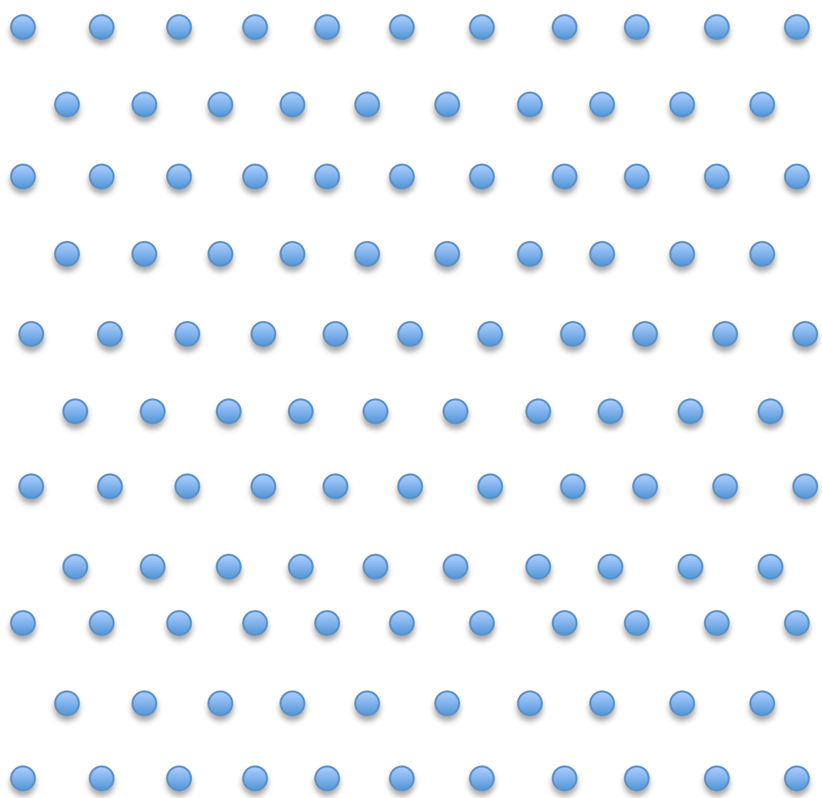


Original 'T'



half of image corrupted by noise

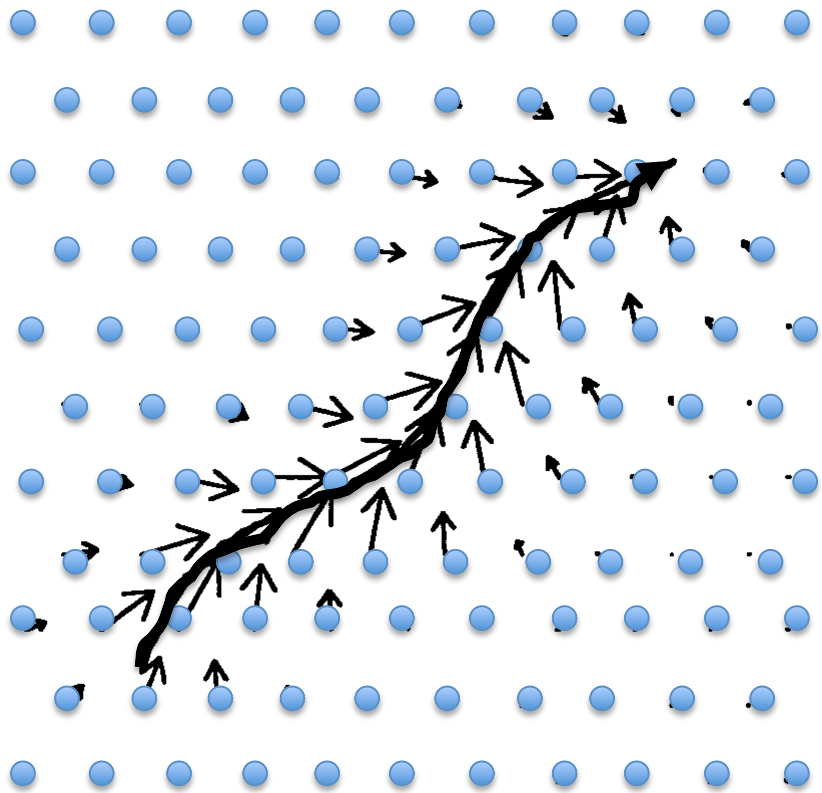
Long Term Potentiation



- Start: Sensory neurons, no connection
- N1 fires before N2: *positive weight, M12 goes up*
- N2 fires before N1: *no change*

(Abbott and Blum, 1996)

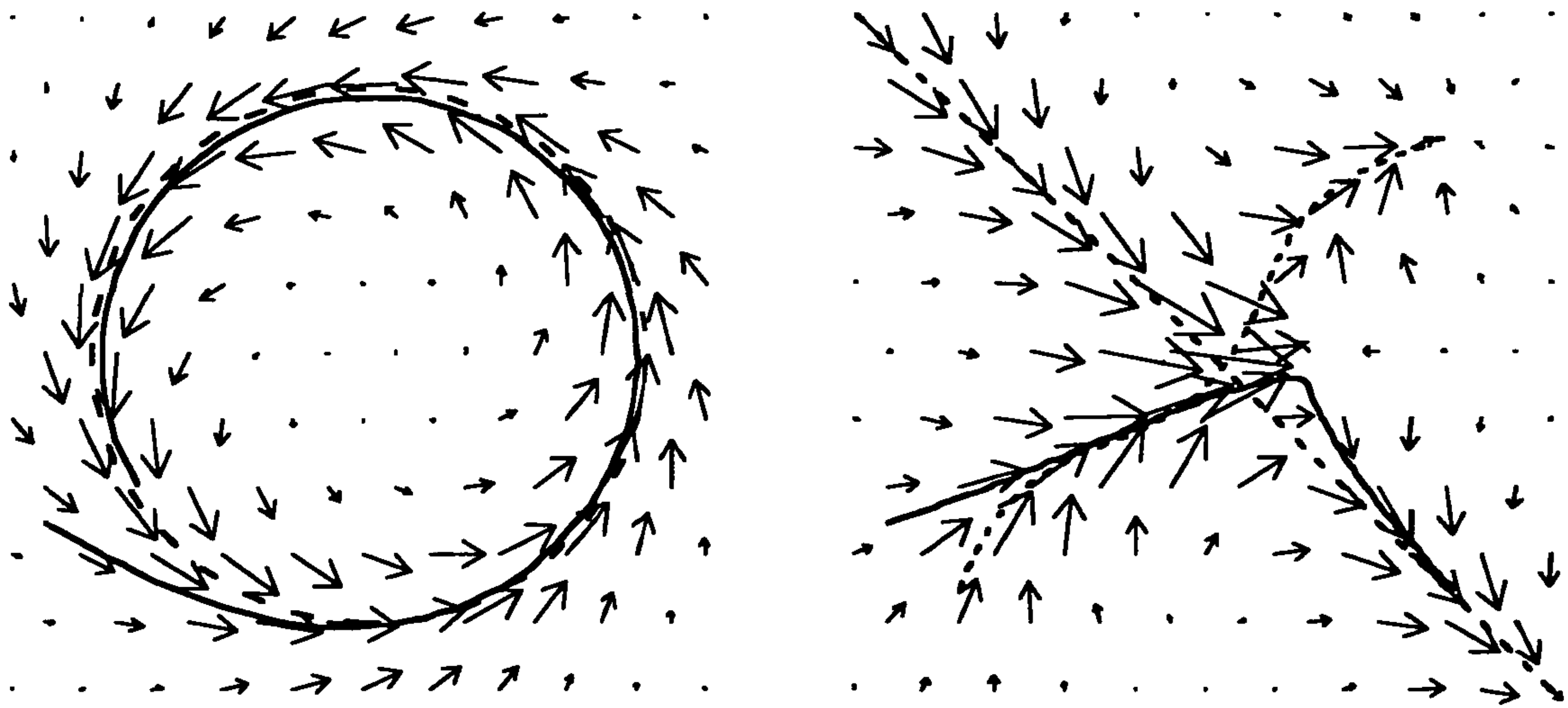
Long Term Potentiation



- Neurons excite the next neurons in the pattern
- Pattern can generate itself

(Abbott and Blum, 1996)

Network Features



(Abbott and Blum, 1996)