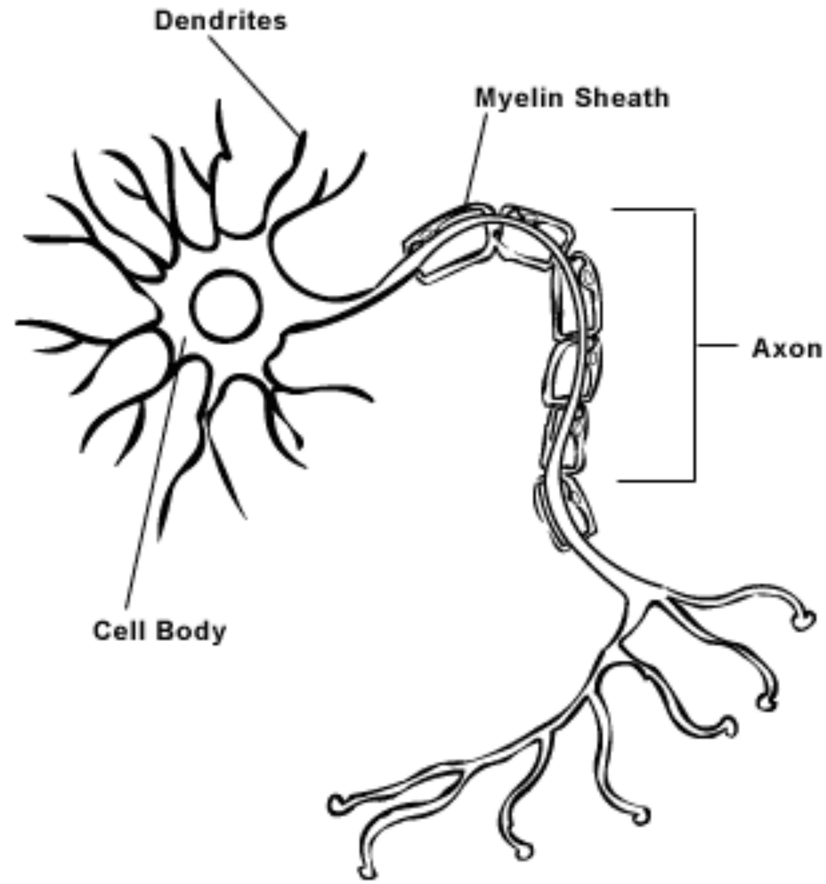
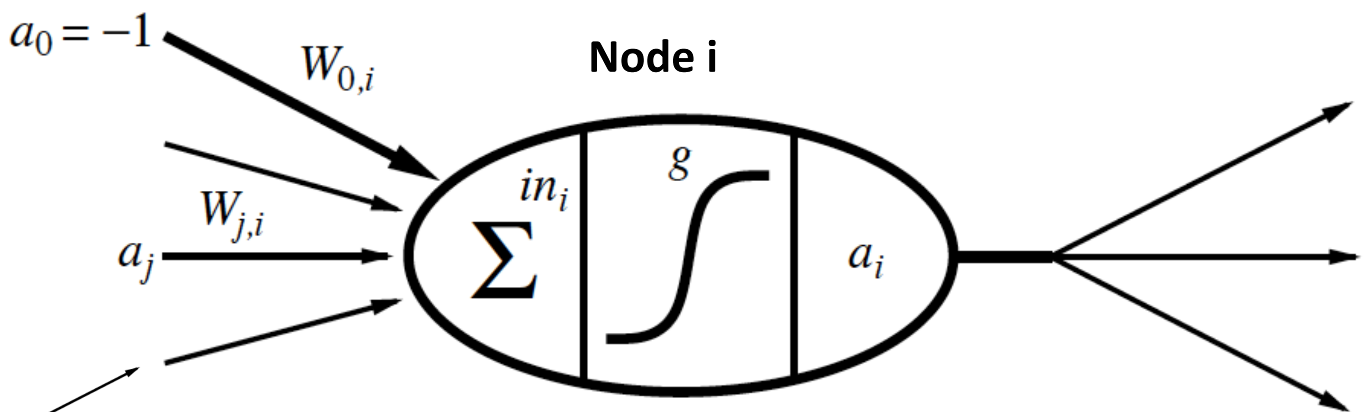
Neural Networks

- Neuron
- Brain information processing emerges from networks of neurons

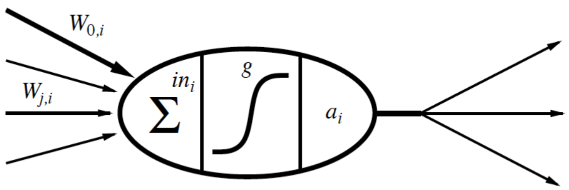


- McCulloch & Pitts (1943)
  - Linear combination of inputs
  - “fire” if threshold is exceeded

Node i



Node j



Nodes are neurons

There is a link from j to i

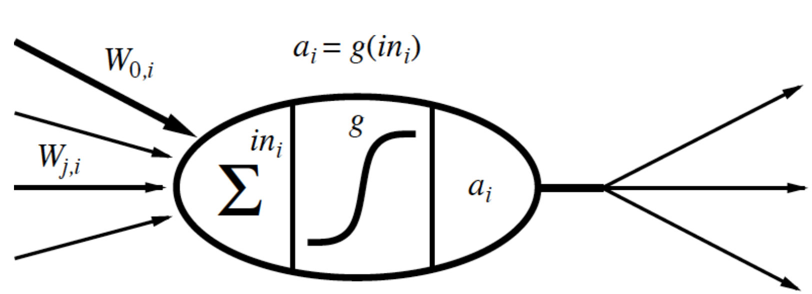
j sends a signal of strength a_j

i receives it with weight $W_{j,i}$

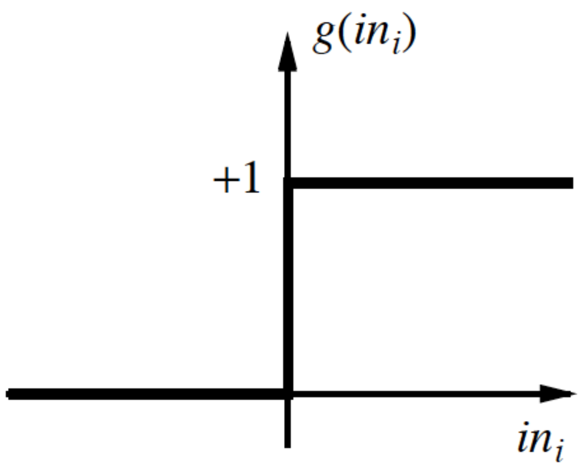
Looks like a matrix lookup

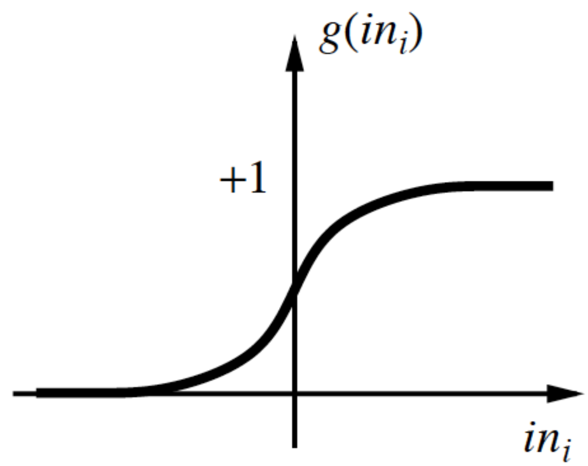
Additionally, each node has a bias, a_0, $W_{0,i}$

- Input to a node: $in_i = \sum_{j=0}^{n} W_{j,i} a_j$
- Activation function: $a_i = g(in_i)$



- What is g?
  - A function that computes near 1.0 when the “right” inputs are given and computers near 0.0 when the “wrong” inputs are given
  - $W_{0,i}$ sets the threshold – actual inputs must overcome bias

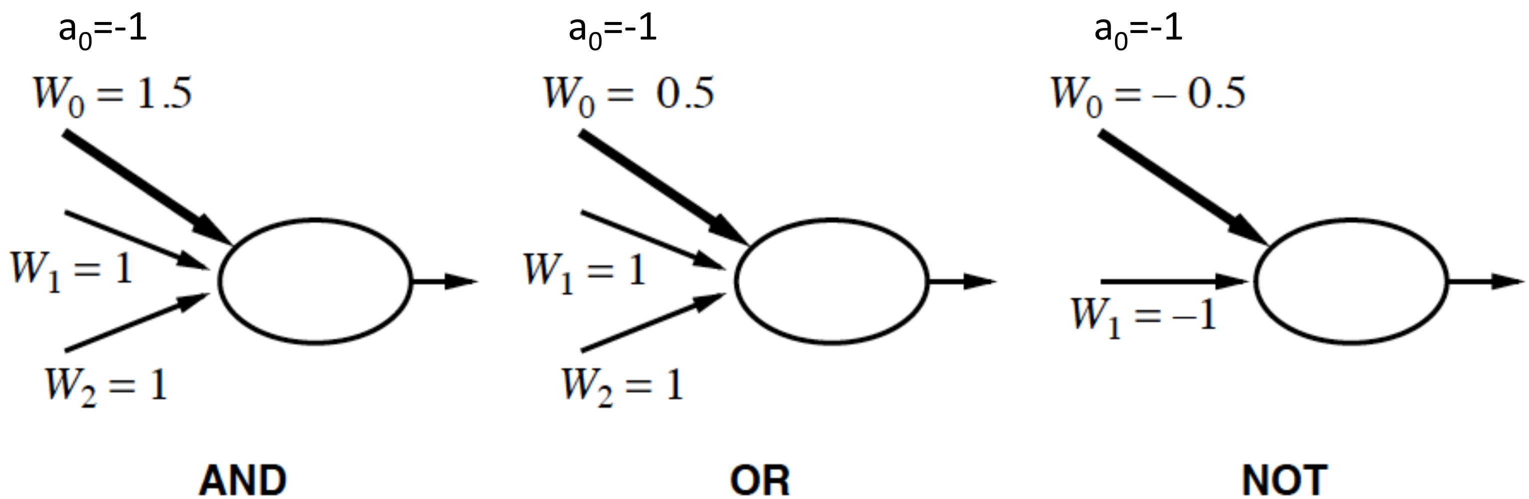




* Better (later when we need to learn)

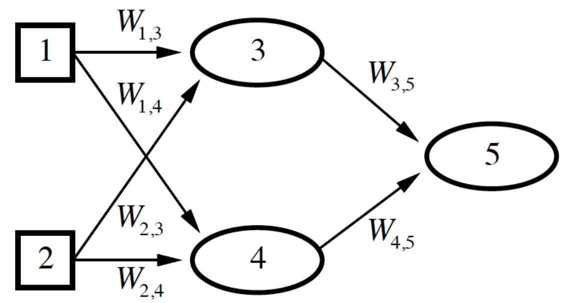
Comparison to logic

- Can replicate logic gates with nodes
- Can compute any boolean logic statement with neural network

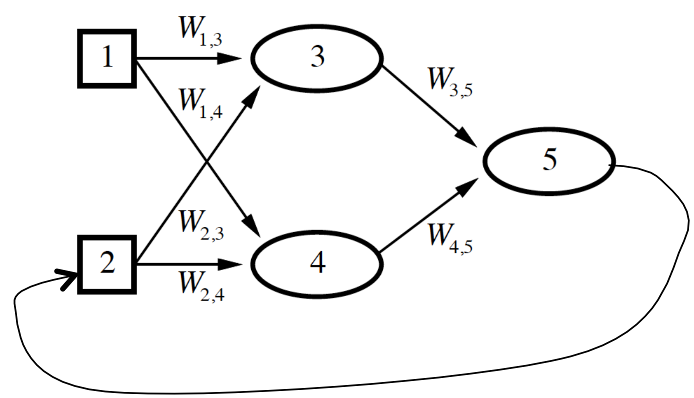


Network Structures

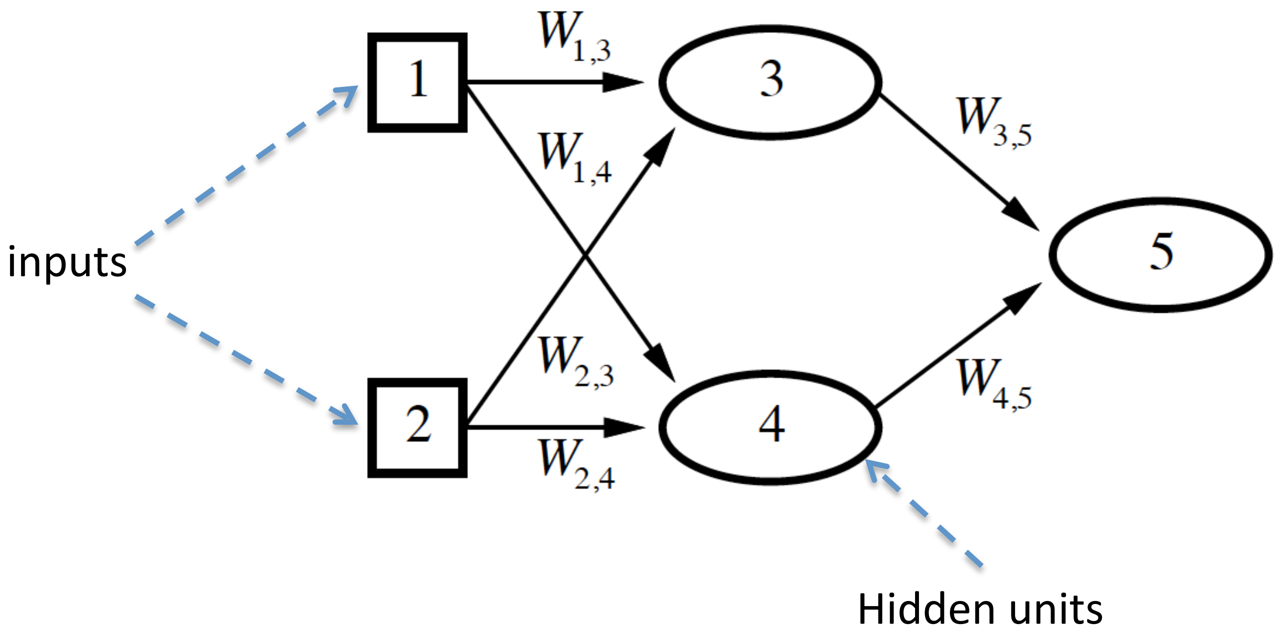
- Feed-forward network
  - Represents a function of current inputs
  - No internal state other than weights
  - Output is the result of the function



- Recurrent network
  - Like feed-forward, but feeds its outputs into its own inputs
  - Can get oscillations (negative weight on its own inputs) or chaotic behavior or stable behavior
  - Can persistent state (like short-term memory)



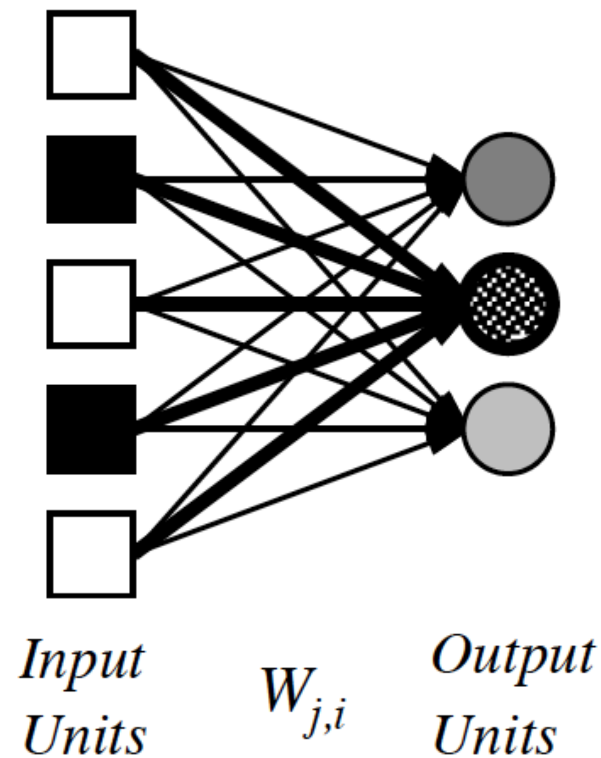
A simple network



$$
\begin{aligned}
a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\
&= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))
\end{aligned}
$$

- Adjusting the weights changes the function that the network represents
- This is how learning occurs

- Classification
  – Boolean: single output node
  – K-way: k output nodes
- Perceptron network
  – Single-layer feed-forward classification

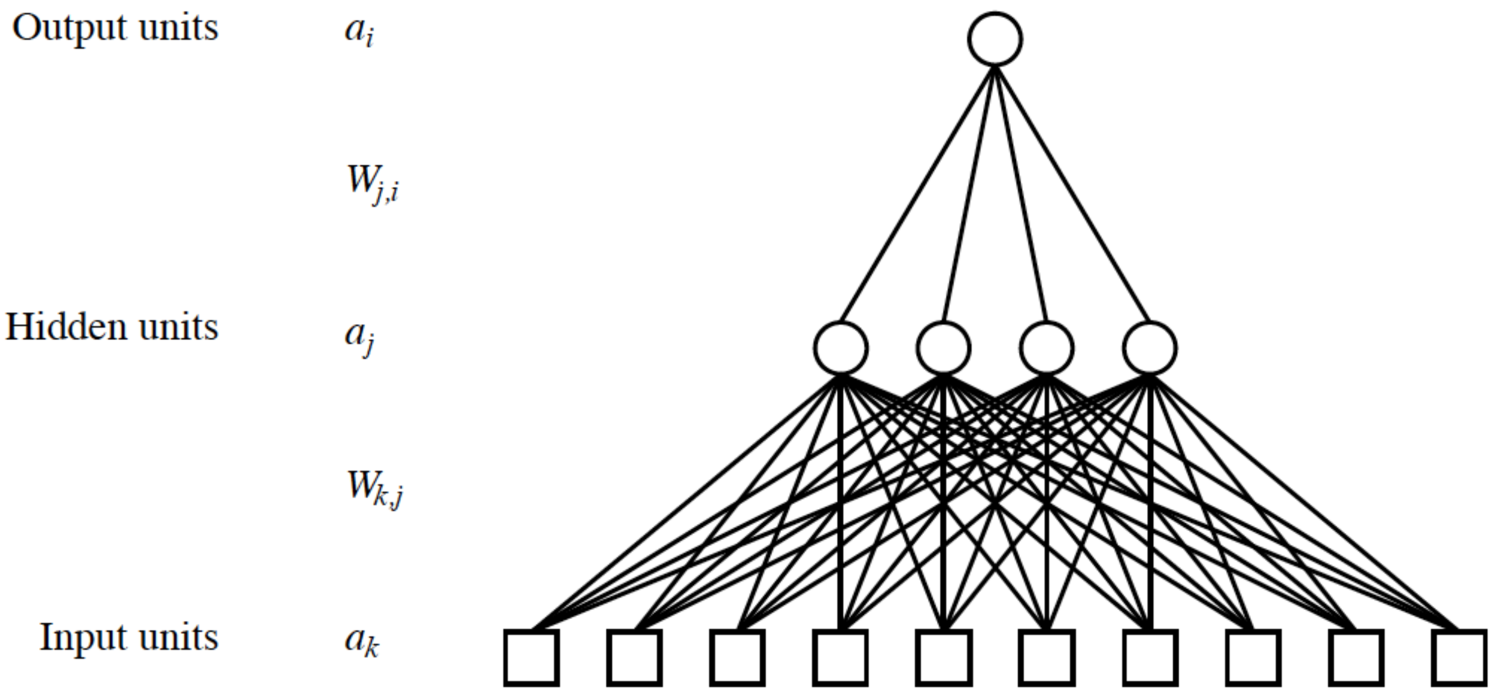


Perceptron learning

- Want the network to learn to replicate some function
- Adjust the weights of the network to minimize error on the training set
  - Optimization search in weight space
  - How to measure error: sum of squared errors

Multilayer network learning

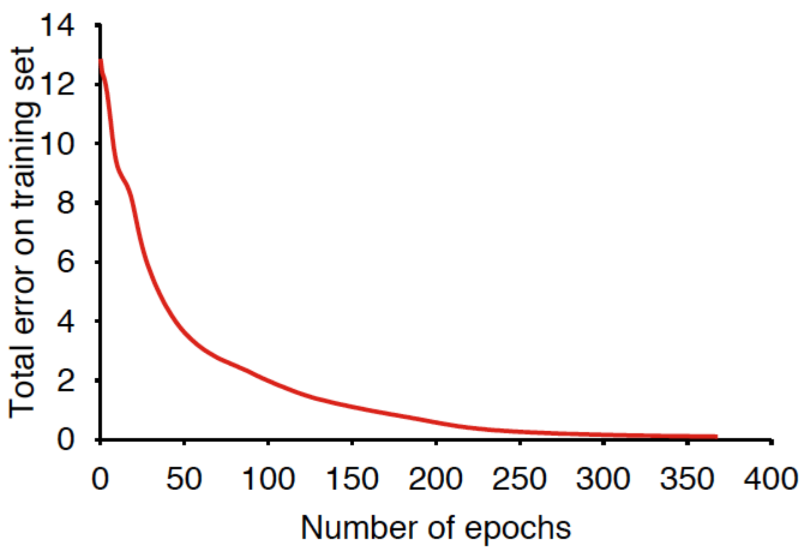
- Error can be caused by hidden nodes
- Back-propagation of error
  - Each hidden node is responsible for some fraction of the total error based on strength of connections
  - Update weights based on the amount of error each node is responsible for

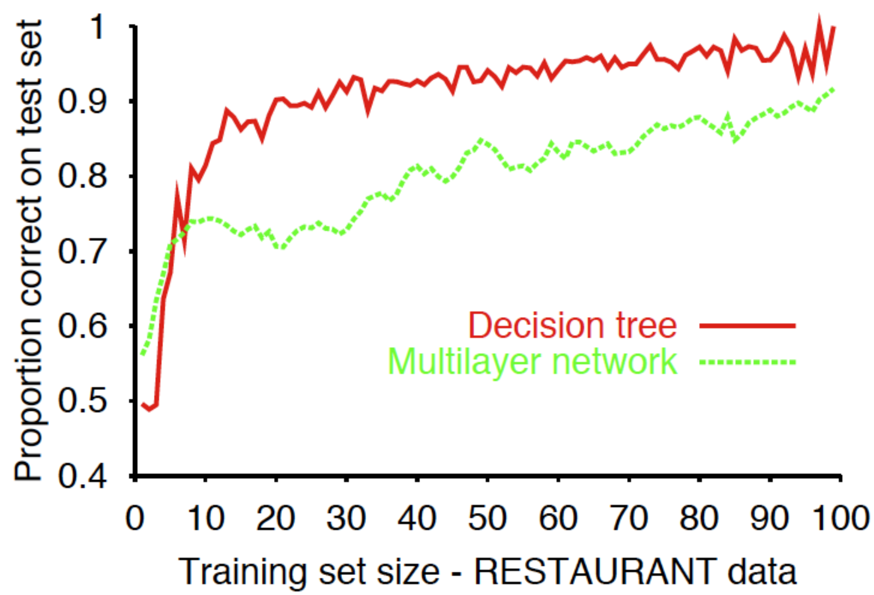


Setting up a neural network learning problem

- Decide structure
  - Perceptron network
  - Multilayer network
    - Multilayer usually has one hidden layer
    - How many hidden nodes is a trial-and-error process
  - Should layers be fully connected?
    - Usually yes

- Training on restaurant data
- Error decreases to zero – converges to a perfect fit on training data
- Training curve is slower to reach asymptote than decision tree
- May need to manually tweak network structure to get perfectly optimal





Learning network structure

- Previous examples assume structure is given
- Random restart, changing number of hidden nodes
- Neuro-evolution
  - Genetic algorithm is used to "grow" a network using crossover and mutation
  - Fitness function is minimization of error

- http://eplex.cs.ucf.edu/dance_evolution/

Neuroevolution of structure

- Binary encoding
  - Bit string encodes entire connection matrix
  - Crossover swaps part of bit string
- Graph encoding
  - Each gene is a node or a link (in, out)
  - Crossover: subgraph swapping
  - Mutation: add link, add node
- Evolutionary programming
  - No crossover – it can lead to loss of functionality or illegal variants