Lecture 17: Neural Networks and Deep Learning

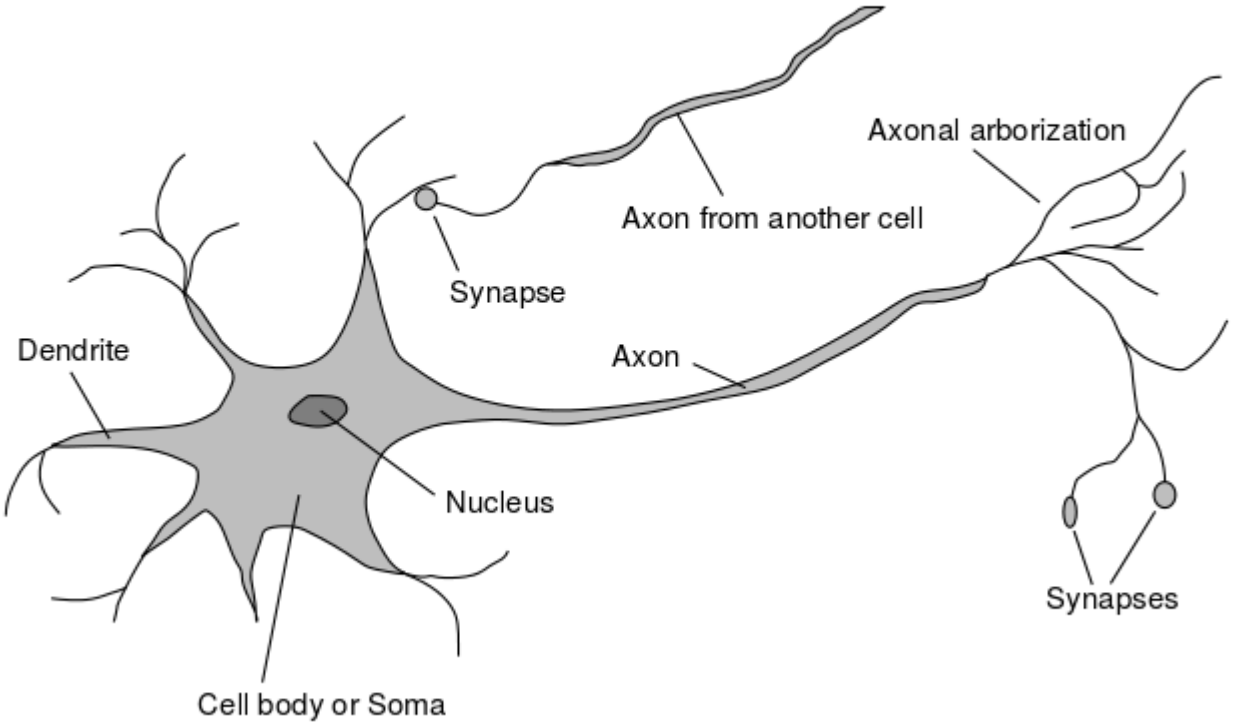
Instructor: Saravanan Thirumuruganathan

Outline

- Perceptron
- Neural Networks
- Deep Learning
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Auto Encoders
  - Neural Turing Machines
  - Adversarial Inputs
  - *Tricks, GPUs and Frameworks

Brains

10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
Signals are noisy "spike trains" of electrical potential



Skynet

Terminator 2 (1991)

JOHN: Can you learn? So you can be... you know. More human. Not such a dork all the time.

TERMINATOR: My CPU is a neural-net processor... a learning computer. But **Skynet** presets the switch to "read-only" when we are sent out alone.

...

We'll learn how to **set** the neural net

TERMINATOR Basically. (starting the engine, backing out) The **Skynet** funding bill is passed. The system goes on-line August 4th, 1997. Human decisions are removed from strategic defense. **Skynet** begins to learn, at a geometric rate. It becomes **self-aware** at 2:14 a.m. eastern time, August 29. In a panic, they try to pull the plug.

SARAH: And **Skynet** fights back.

TERMINATOR: Yes. It launches its ICBMs against their targets in Russia.

SARAH: Why attack Russia?

TERMINATOR: Because **Skynet** knows the Russian counter-strike will remove its enemies here.

http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/neural.pdf

Perceptron

For $\mathbf{x} = x_1, \ldots, x_d$ ("features of the customer"), compute a weighted score and:

$$\text{Approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold},$$

$$\text{Deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}.$$

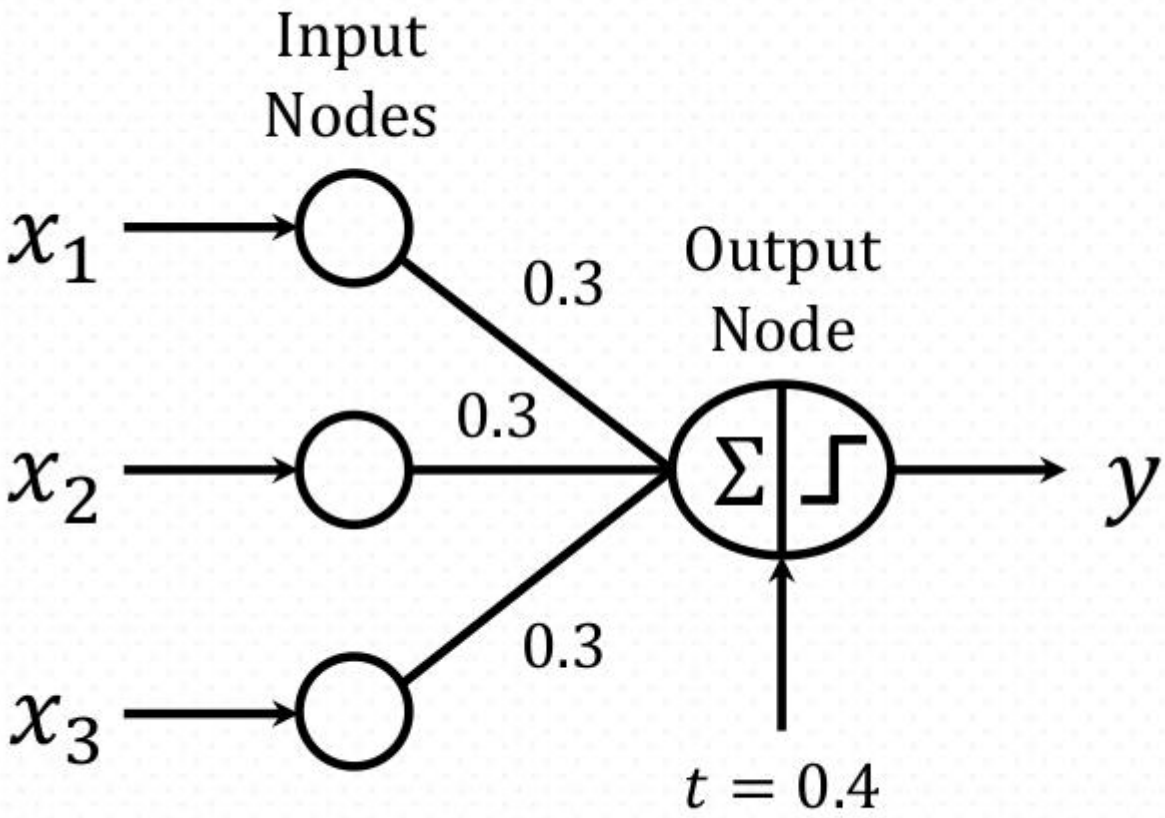
Perceptron

This formula can be written more compactly as

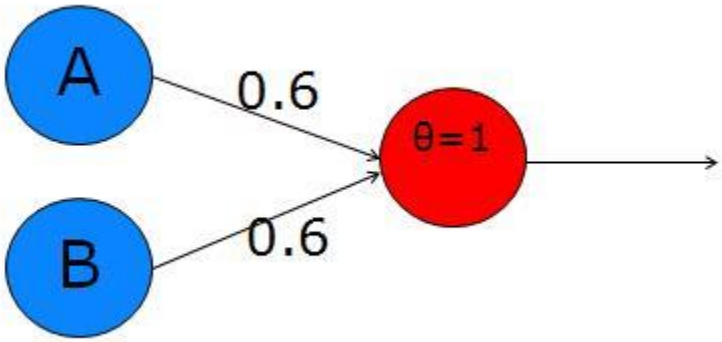
$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right),$$

where $h(\mathbf{x}) = +1$ means 'approve credit' and $h(\mathbf{x}) = -1$ means 'deny credit'; $\text{sign}(s) = +1$ if $s > 0$ and $\text{sign}(s) = -1$ if $s < 0$. This model is called a *perceptron*.

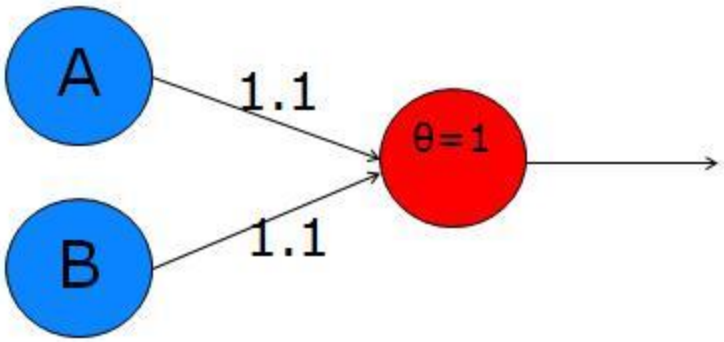
Perceptron



Perceptron

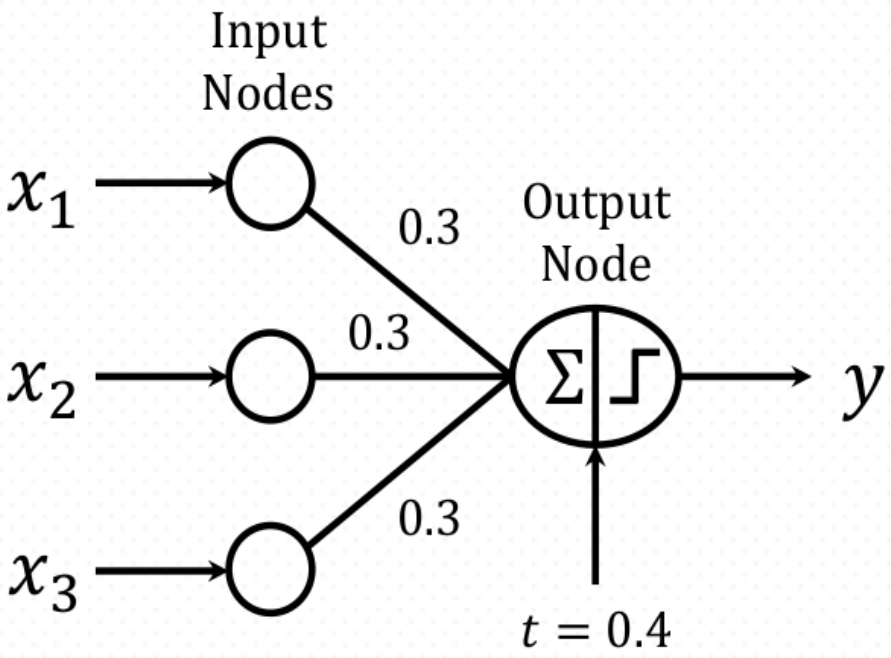


Perceptron



Perceptron

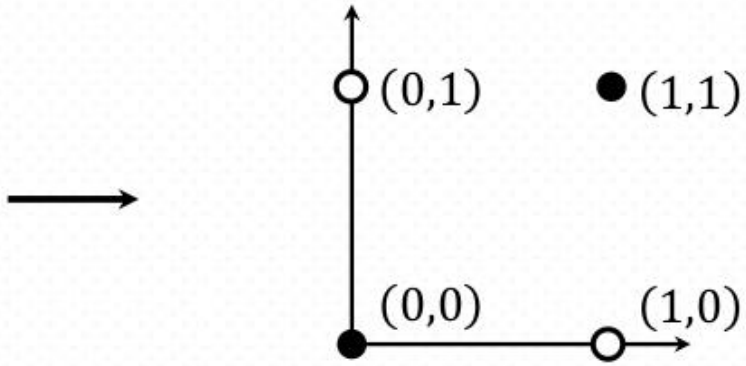
x_1	x_2	x_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



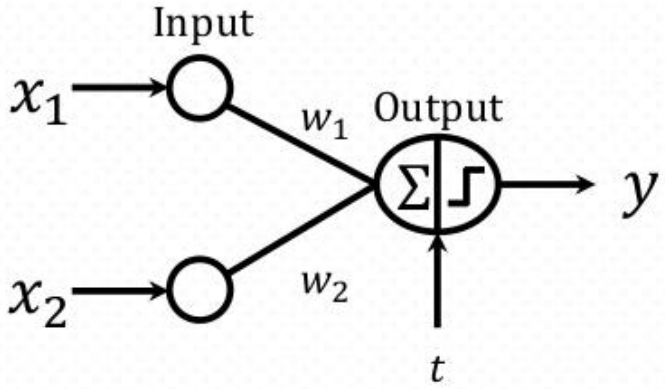
Perceptron: Limitations (XOR)

Data

x_1	x_2	$y = x_1$ XOR x_2
1	1	0
1	0	1
0	1	1
0	0	0



Model



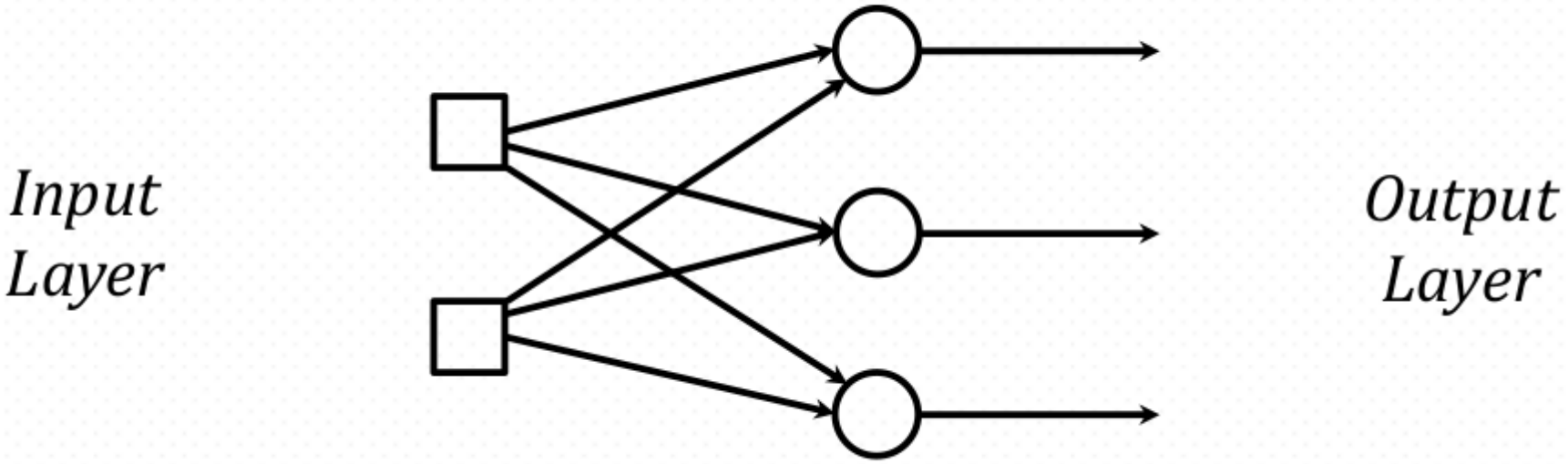
The following cannot all be true:

$$w_1 \times 1 + w_2 \times 1 < t$$
$$w_1 \times 1 + w_2 \times 0 > t$$
$$w_1 \times 0 + w_2 \times 1 > t$$
$$w_1 \times 0 + w_2 \times 0 < t$$

Network Architectures

- Three different architectures
  - Single-layer feed-forward
  - Multi-layer feed-forward
  - Recurrent
- The architecture of a neural network is linked with the learning algorithm used to train.

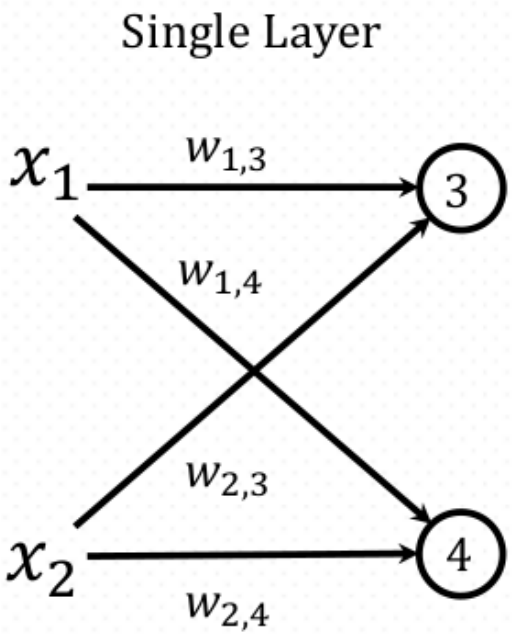
Single-Layer Feed-Forward (FF) NN

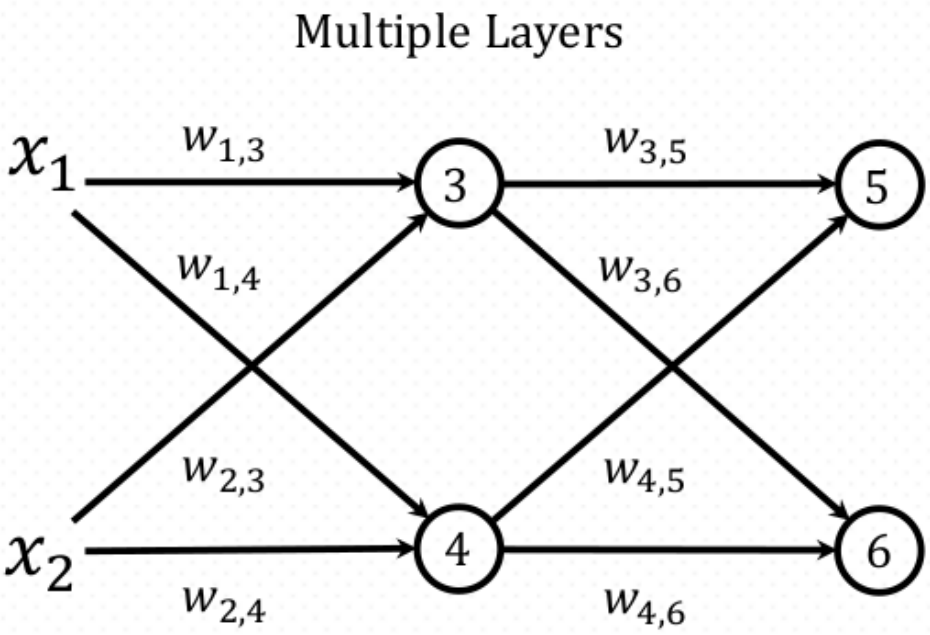


Multilayer Artificial Neural Networks

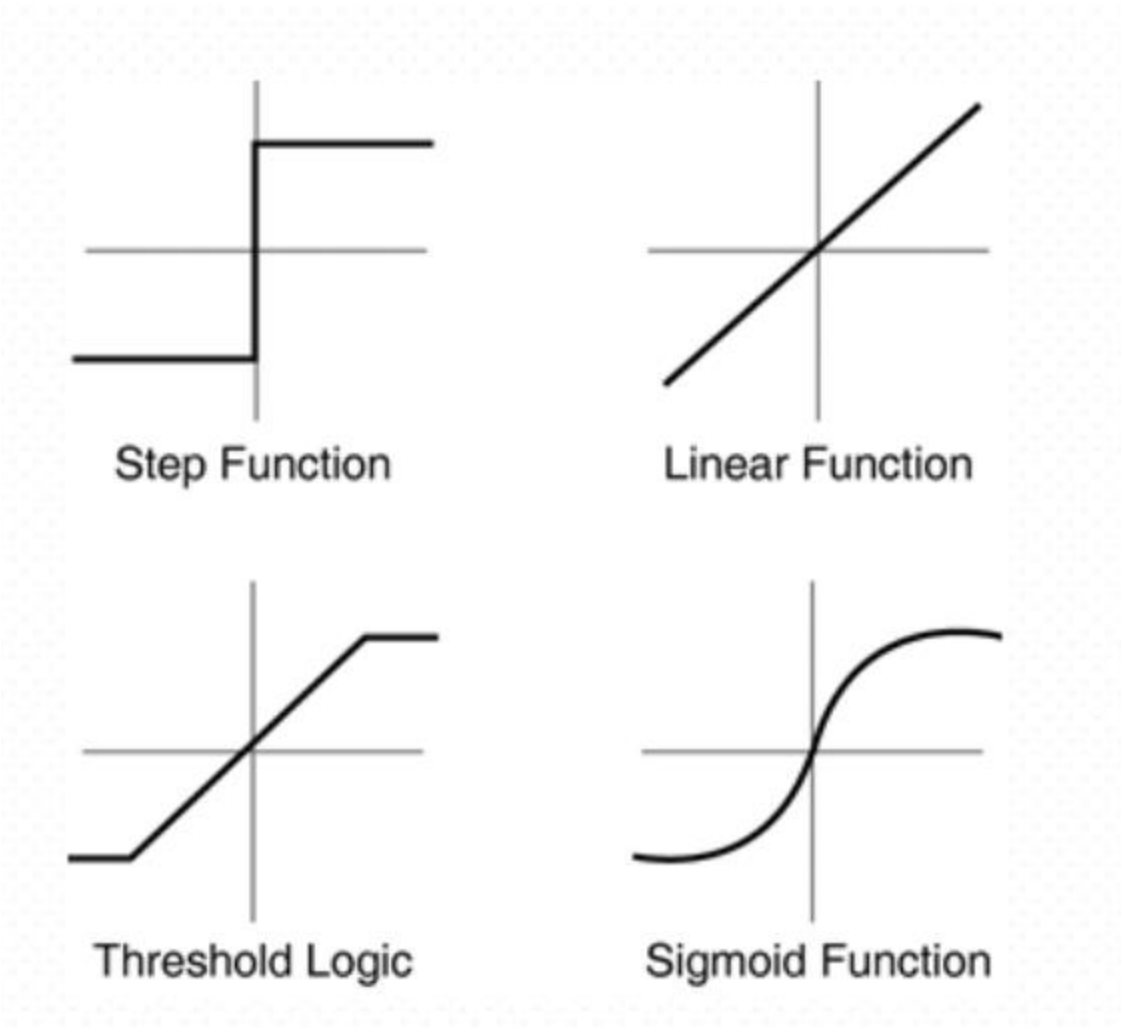
- An artificial neural network (ANN) has a more complex structure than that of a perceptron model. The additional complexities may arise in a number of ways:
  - The network may contain several intermediary layers between its input and output layers.
  - The network may use types of activation functions other than the sign function.

Multilayer Perceptrons

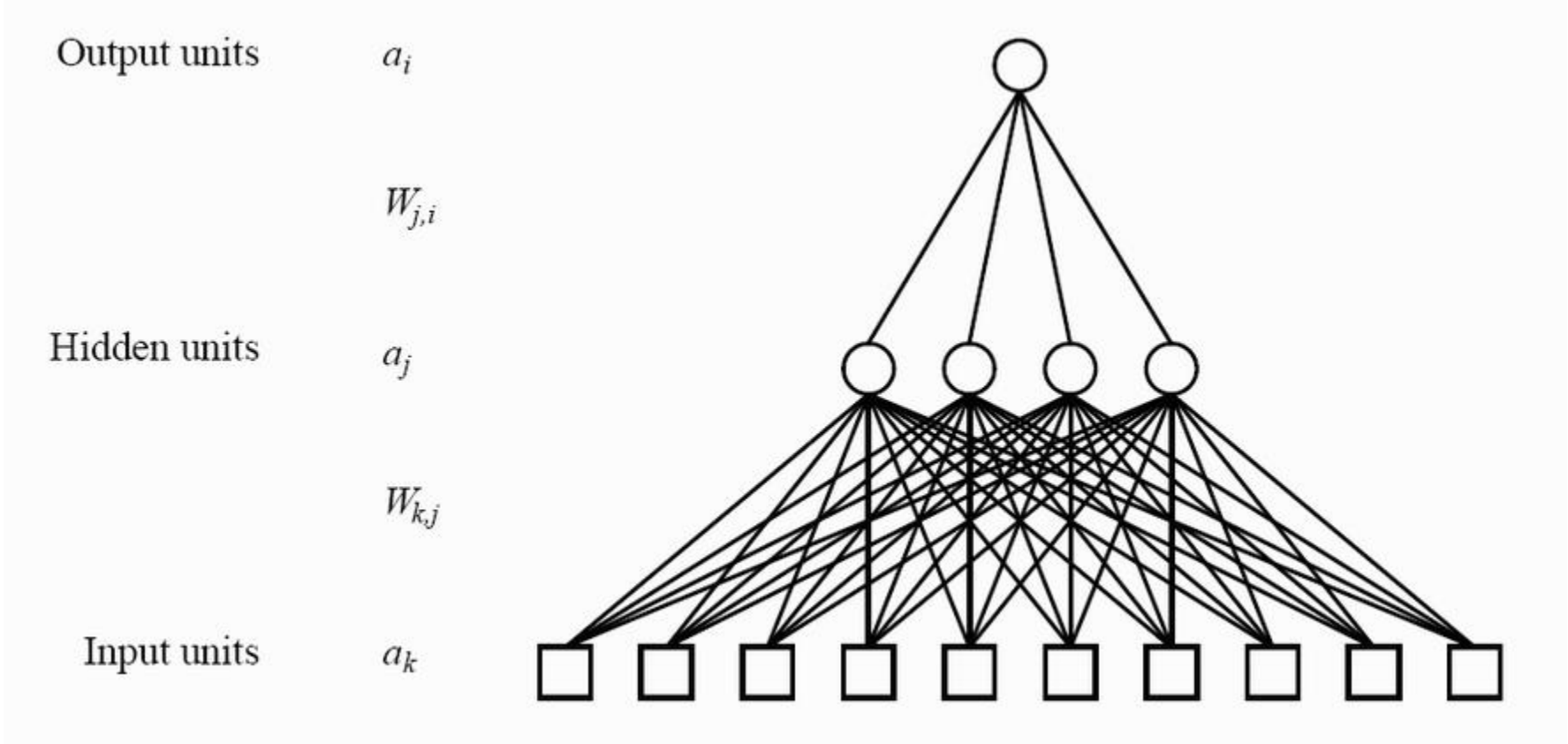




Activation Functions



Multi Layer Perceptrons



Multi Layer Perceptrons

- Layers are typically fully connected
- Number of hidden units typically chosen by hand

ANN Learning

To learn the weights of an ANN model, we need an efficient algorithm that converges to the right solution when a sufficient amount of training data is provided.

Gradient Descent

The goal of the ANN learning algorithm is to determine a set of weights **w** that minimize the total sum of squared errors:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

The weight update formula used by gradient descent is:

$$w_j \leftarrow w_j - \lambda \frac{\partial E(\mathbf{w})}{\partial w_j}$$, where λ is the learning rate.

Gradient Descent

Key Idea:

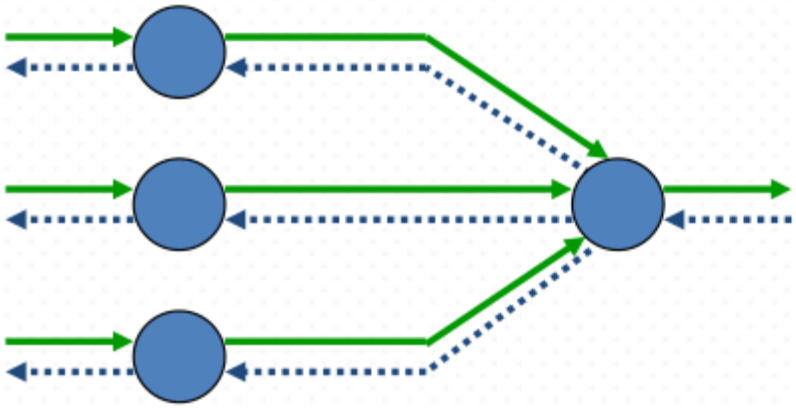
Intuitively, we increase the weight in a direction that reduces the overall error term. The greater the decrease in error, the greater the increase in weight.

Problem:

This only applies to learning the weights of the output and hidden nodes of a neural network.

BackPropagation

- Backpropagation adjusts the weights of the NN in order to minimize the network total mean squared error.



Network activation
Forward Step

Error propagation
Backward Step

BackPropagation

- Developed to address the problem of computing the error for the hidden nodes, since their output values are not known a priori.
- After the weights have been computed for each layer (activation), the weight update formula is applied in the reverse direction.
- Backpropagation allows us to use the errors for neurons at layer k + 1 to estimate the errors for neurons at layer k.

BackPropagation - Training

- Backpropagation consists of the repeated application of the following two passes.
  - **Forward pass:** The network is activated on one example and the error of (each neuron of) the output layer is computed.
  - **Backward pass:** The network error is used for updating the weights. The error is propagated backwards from the output layer through the network layer by layer. This is done by recursively computing the local gradient of each neuron.

BackPropagation - Termination

- Total mean squared error change:
  - absolute rate of change in the average squared error per epoch is small enough
- Generalization based criterion:
  - After each epoch, test NN for generalization performance
  - If performance is acceptable, then stop

Neural Network Topology

- The number of layers and neurons depend on the specific task.
- In practice this issue is solved by trial and error.
- Two types of adaptive algorithms can be used:
  - Begin with a large network and successively remove some neurons and links until network performance degrades.
  - Begin with a small network and introduce new neurons until performance is satisfactory.

Neural Network Design Issues

- Data representation
- Network Topology
- Network Parameters

Deep Learning

- Concept of depth
  - Length of path from input to output
- Motivations for Deep Architectures
  - Insufficient depth can hurt
    - Depth 2 is enough in many cases
    - price: required number of nodes in the graph (and number of parameters) grows very large
  - The brain has a deep architecture
  - Cognitive processes seem deep

Cognitive processes seem deep

- Humans organize their ideas and concepts **hierarchically**.
- Humans first learn **simpler** concepts and then compose them to represent more **abstract** ones.
- Engineers break-up solutions into multiple levels of abstraction and processing

http://www.iro.umontreal.ca/~pift6266/H10/notes/deepintro.html

Deep Learning Breakthroughs

- Training deep architectures is challenging
- Geoff Hinton, Yoshua Bengio, Yann LeCun (independently) published a series of papers in 2006/2007 that provided breakthroughs.

Deep Learning : Demos

- Demo
  - https://www.youtube.com/watch?v=Nu-nlQqFCKg&t=187
    - Skype translate!
  - Google image search
  - DeepMind: Playing Atari Games
    - https://www.youtube.com/watch?v=V1eYniJ0Rnk

Slide References

- Stanford CS 231n
- Univ of Notre Dame: CSE 60647, Spring 2014
- Bayesian Behavior Lab, UNW
- Princeton, COS598 Spring 2015: The Unreasonable Effectiveness of Big Visual Data
- Slides from Deep Learning tutorials
- Slides from DeepMind