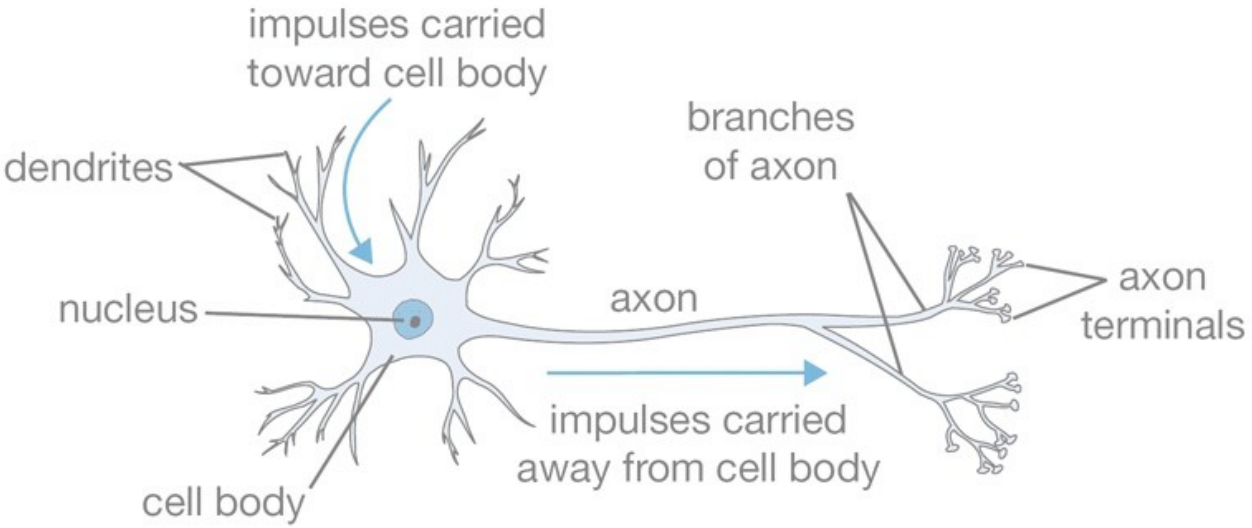
Chapter 6

Deep feedforward networks

Simen, Simon, Magnus & Øyvind

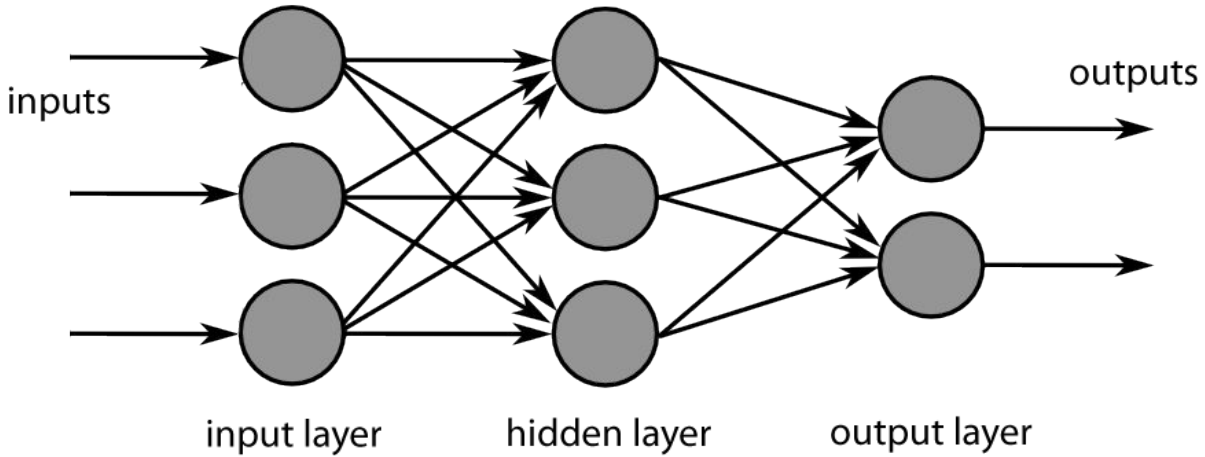
Topology

“...occasionally drawing some insights from what we know about the brain, rather than as models of brain function”



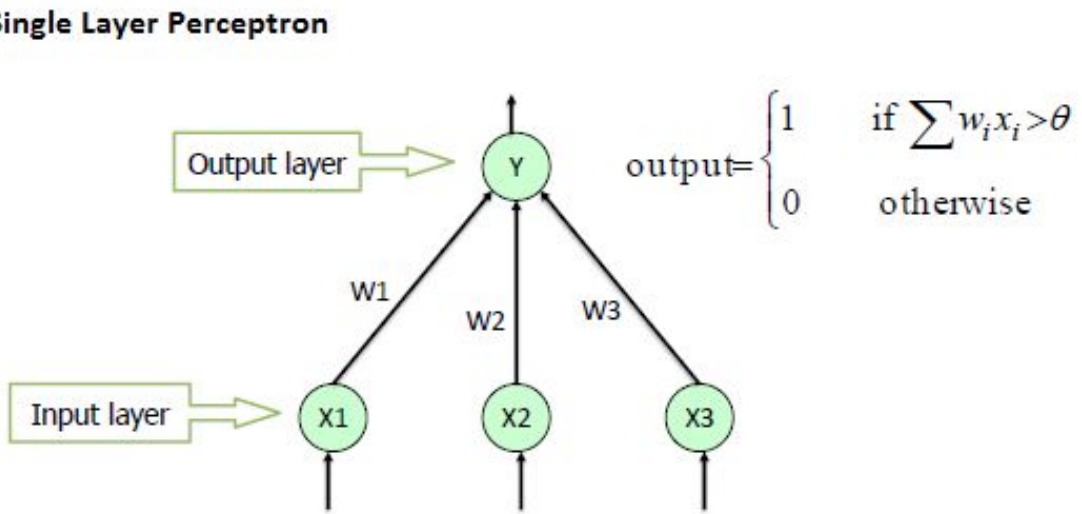
Artificial neural network

- Feedforward: Connections **do not** form a cycle
  - directed acyclic graph (DAG)
- Recurrent: Network with cycles

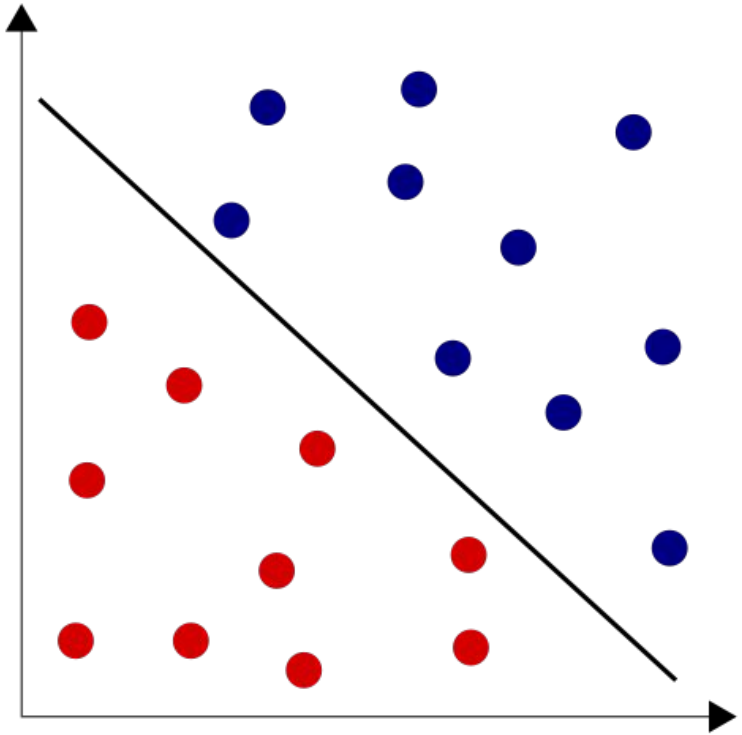


Single-layer perceptron

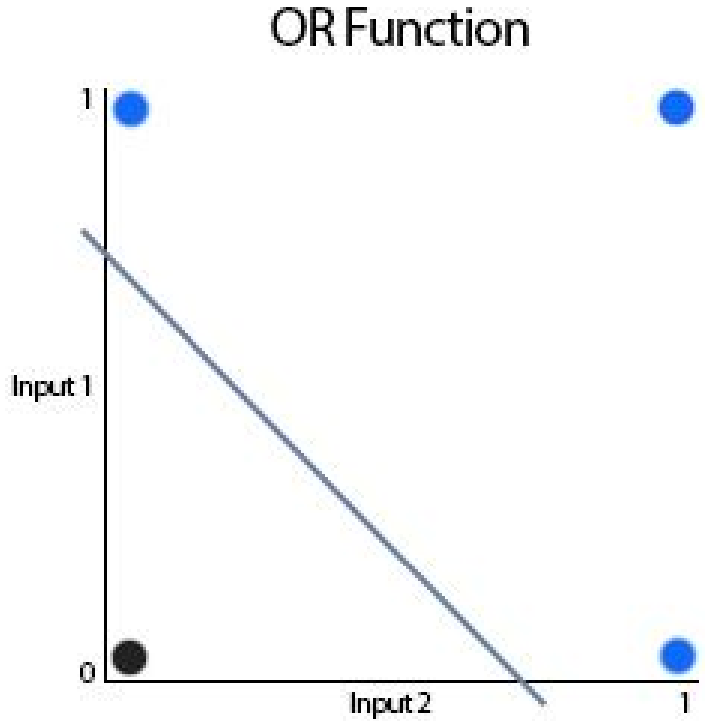
- Perceptron
  - Maps a real-valued vector to a binary output
- Input and output layer only
- 1940s
- Limitations?



Linear separability

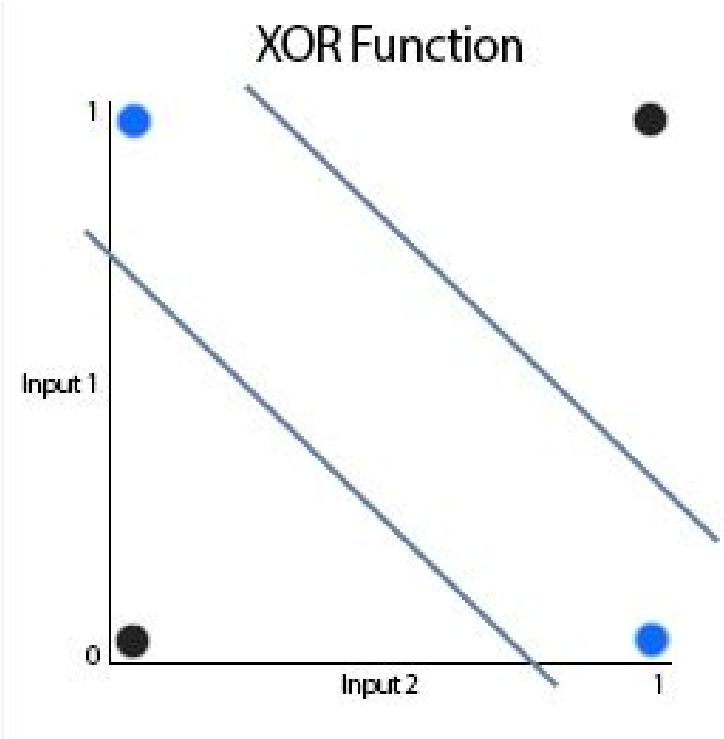


Linear separability OR



A	B	out
0	0	0
0	1	1
1	0	1
1	1	1

Linear separability XOR

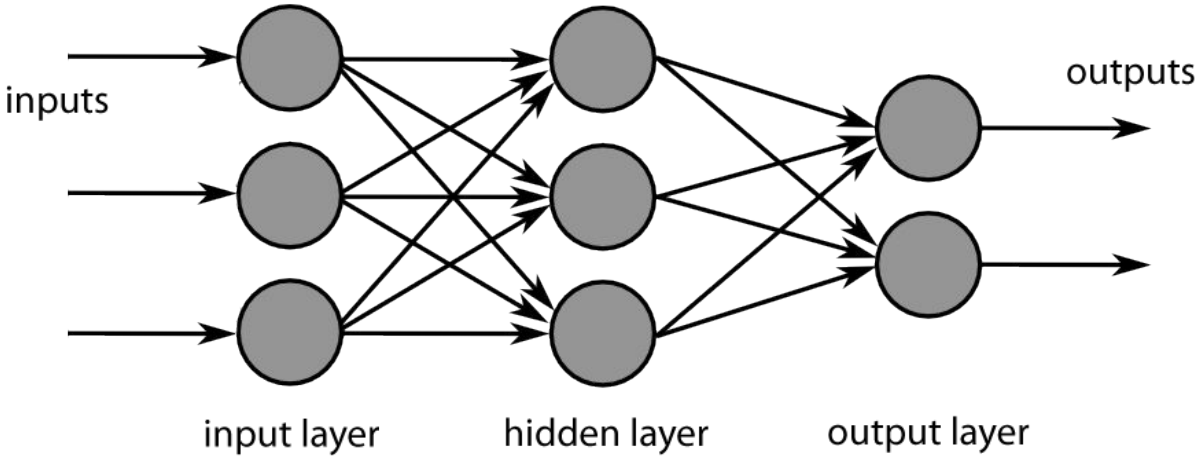


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

The first AI winter
Late 60s - Mid 1980s

Multi-layer perceptron

- 1 or more hidden layers
- Can learn nonlinear functions
- Previous problem: How to train the hidden layers?
  - Gradient descent



TensorFlow Playground

http://playground.tensorflow.org/

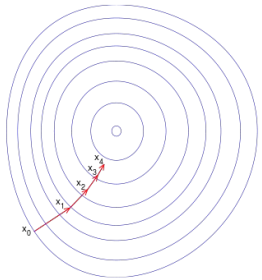
Back-propagation

Prerequisites: important discoveries

- Chain rule of calculus [Leibniz, 1676]
  - Expresses the derivative of the composition of functions.

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}.$$

- Gradient descent [Cauchy, 1847]
  - Iterative optimization algorithm



Status before back-propagation

- Current status of neural nets before back-propagation:
  - Multilayer perceptron networks
  - Learning of nonlinear functions
- Problem: How to we train the multilayer perceptron networks?
  - How to we find the **gradients**; in what direction to move the weights

Forward-propagation

- Training set: Set of (x,y) pairs that describes the function we want to approximate.
- Feed the x to the input-layer of the neural net
- The input propagates through the net, and is adjusted by weights and functions
- Receive the output from the neural net

$$x \rightarrow \hat{y}$$

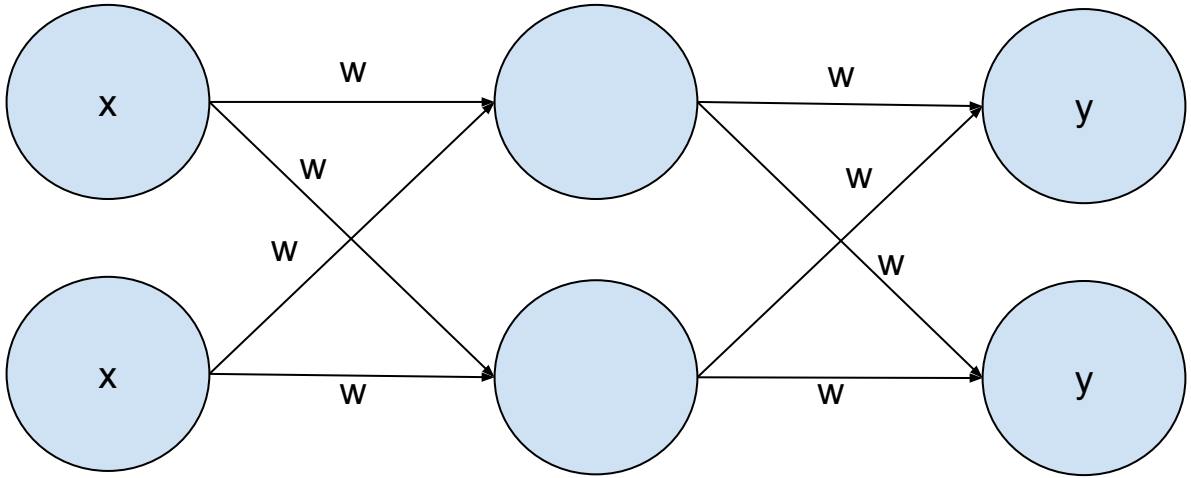
Forward-propagation

- Did the network correctly predict the output y?
- Error

$$(y - \hat{y}) = Err$$

- What can we change?
  - Number of hidden layers?
  - Number of nodes in them?
  - The weights?

Forward-propagation



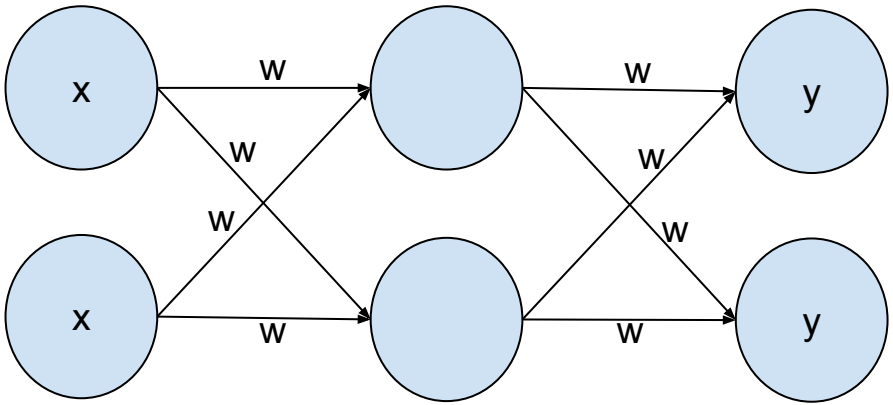
How much does each weight contribute to the error?

Back-propagation

- Back-Propagation [Rumelhart et al., 1986]

- Method for computing the gradient for gradient descent
  - Chain rule of calculus

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}.$$



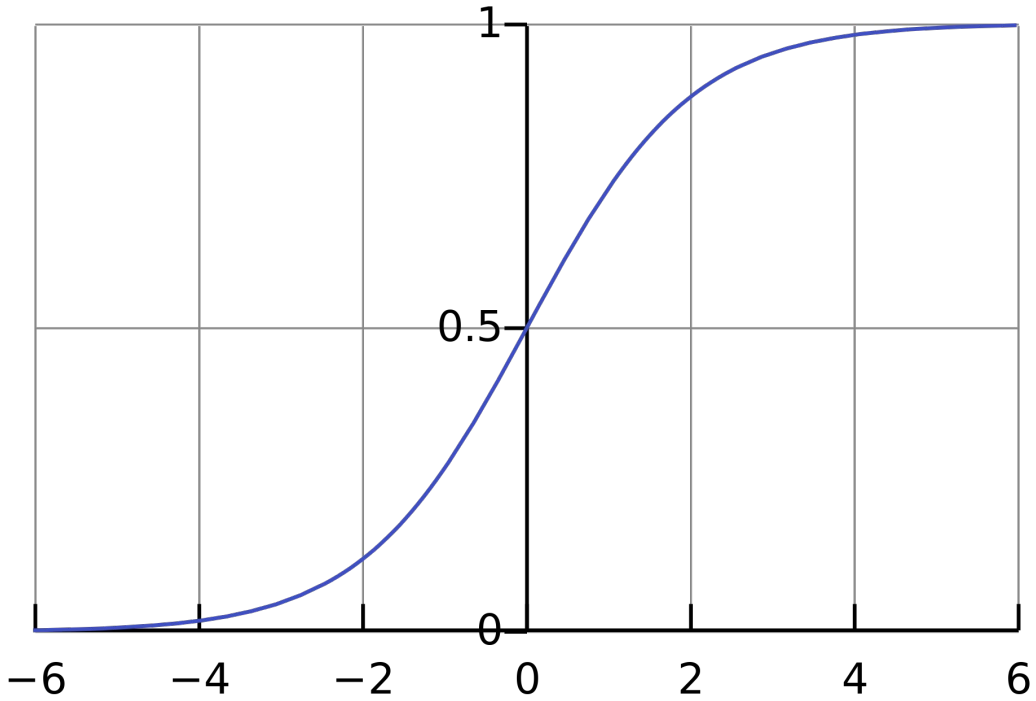
Status after back-propagation

- Gradient descent + back-propagation = dynamite
- Multilayer networks now trainable with gradient descent

- Neural nets gained popularity and reached a peak in the early 90's
- Core ideas from the late 80's still in use today
- Improvements of modern ANN's come from
  - More available, larger data sets
  - Faster software and hardware

Hidden units

Sigmoid Tanh ReLU Maxout

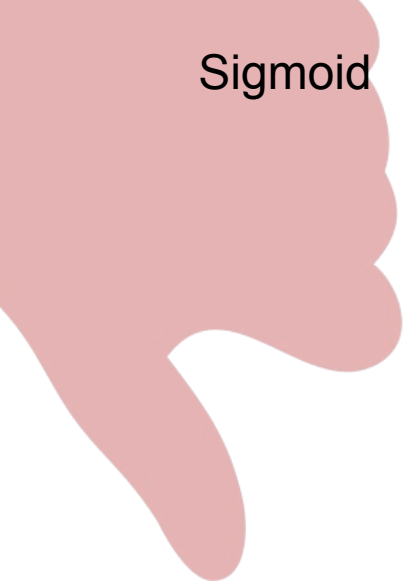


Sigmoid Tanh ReLU Maxout

Bounded

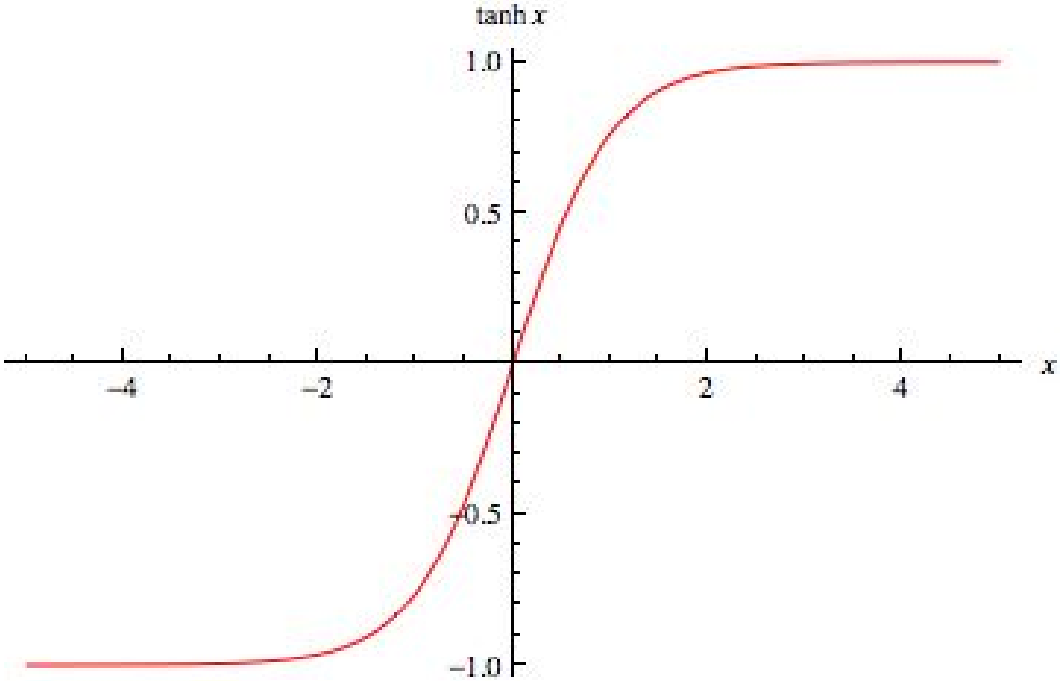
Continuous

Sigmoid Tanh ReLU Maxout



Easily saturated

Only positive output

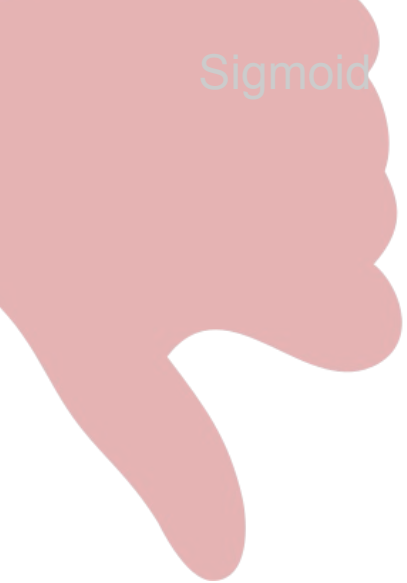

tanh x
1.0
0.5
−0.5
−1.0
−4
−2
2
4
x

Similar to Sigmoid

Both positive and negative output

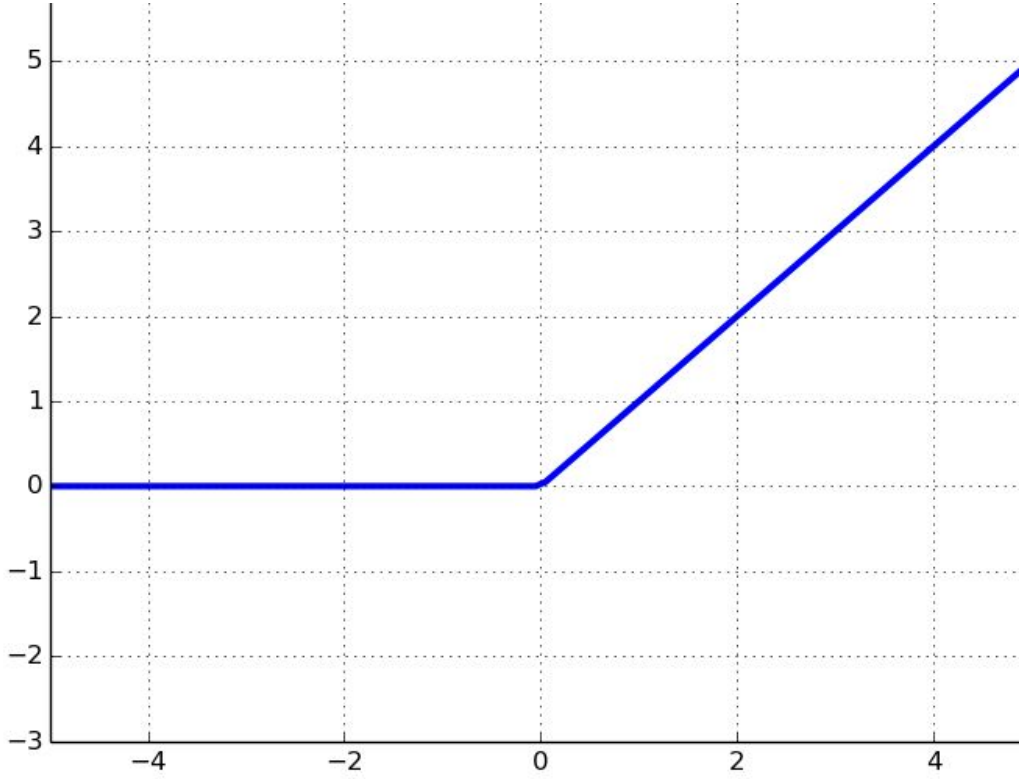
Usually performs better than Sigmoid

Sigmoid Tanh ReLU Maxout



Easily saturated

Sigmoid Tanh **ReLU** Maxout

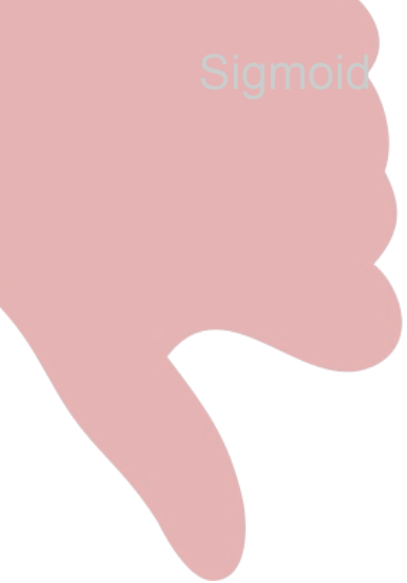


Easy to implement

Does not saturate

Promotes sparse networks

Sigmoid Tanh **ReLU** Maxout

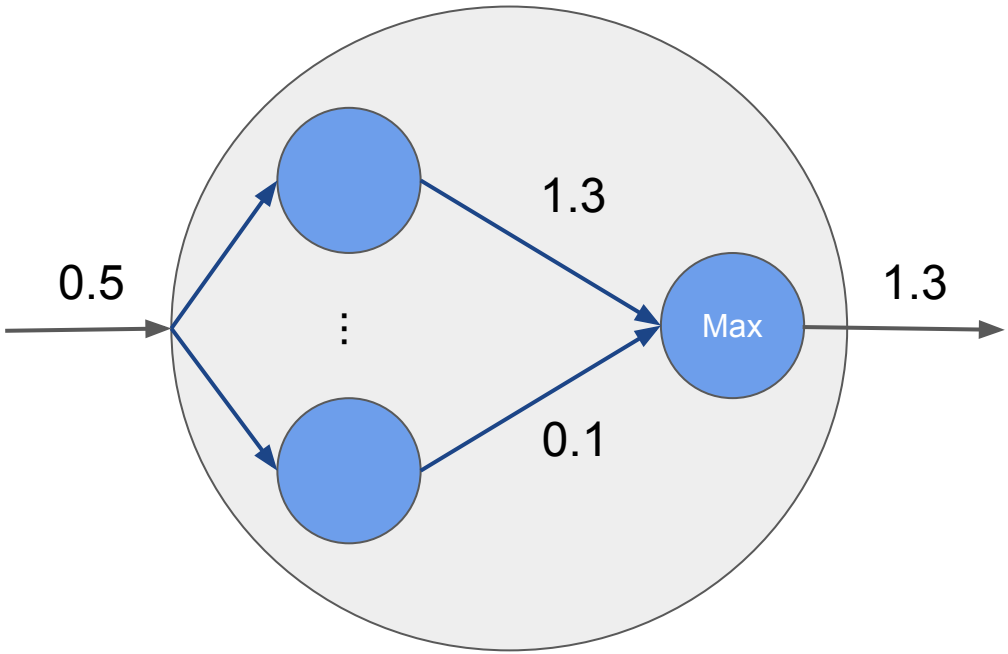


Unbounded

Not continuous

Dying neurons

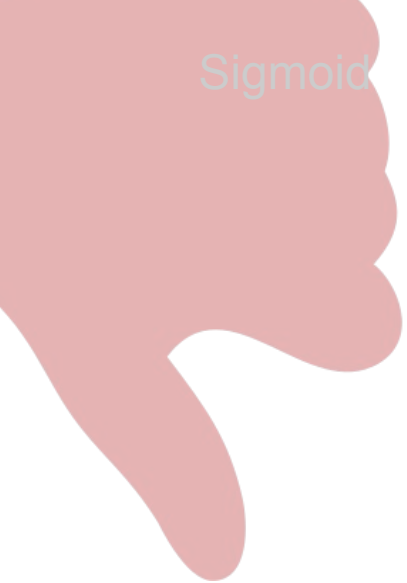
Sigmoid Tanh ReLU **Maxout**



A generalisation for ReLU

No dying neurons

Sigmoid Tanh ReLU **Maxout**



Unbounded

Not continuous

Extra parameters to train

What to remember

ReLU is an excellent default choice

Do not use Sigmoid

TensorFlow activation functions demo

http://playground.tensorflow.org/#activation=sigmoid&dataset=circle&networkShape=3

Network Architecture

How to design the network?

Goal of the neural network?

- Represent a given (Borel measurable) function

Universal approximation theorem

- “...regardless of what function we are trying to learn, we know that a large MLP will be able to represent this function”
- Does not state what the network will look like
- May not learn the function; overfitting and not finding correct parameter values

What to consider?

- Number of hidden layers
- Width of each layer
- Connectivity
- Activation functions

How to design the network?

Trial and error, often hard to know what will work

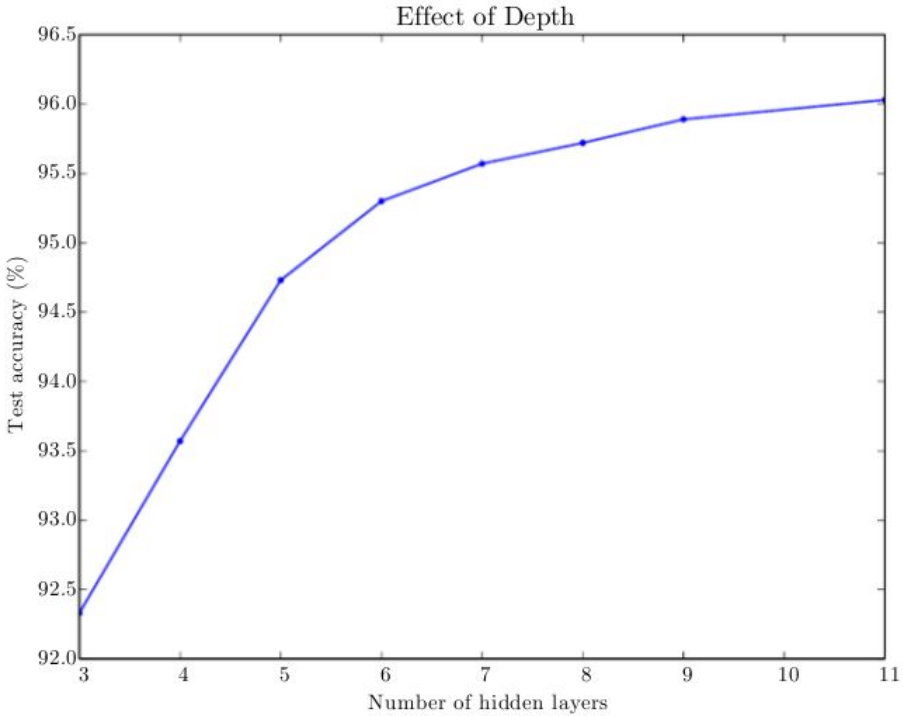
Feedforward networks a common approach

- Chained layers
- Extra features
  - Sparse connections
  - Skip connections
    - Connections from layer *i* to *i* + *2* and higher
    - Improved flow of gradient from output to input

Some problem-specific guidelines

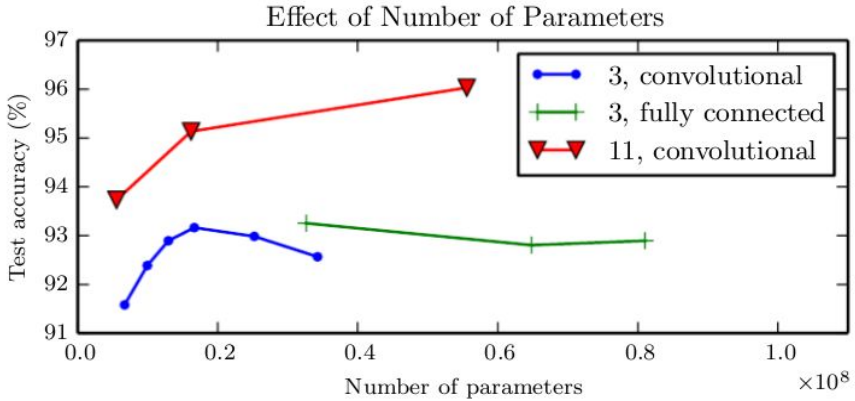
- Convolutional networks for computer vision (chapter nine)
  - Specialized patterns of sparse connections
- Recurrent networks for sequence processing (chapter ten)
  - Inter-layer connections, recurrent connections, backwards connections
  - “Internal memory”

Depth of network



Deeper network ⇨ fewer parameters

Deeper networks generalize better



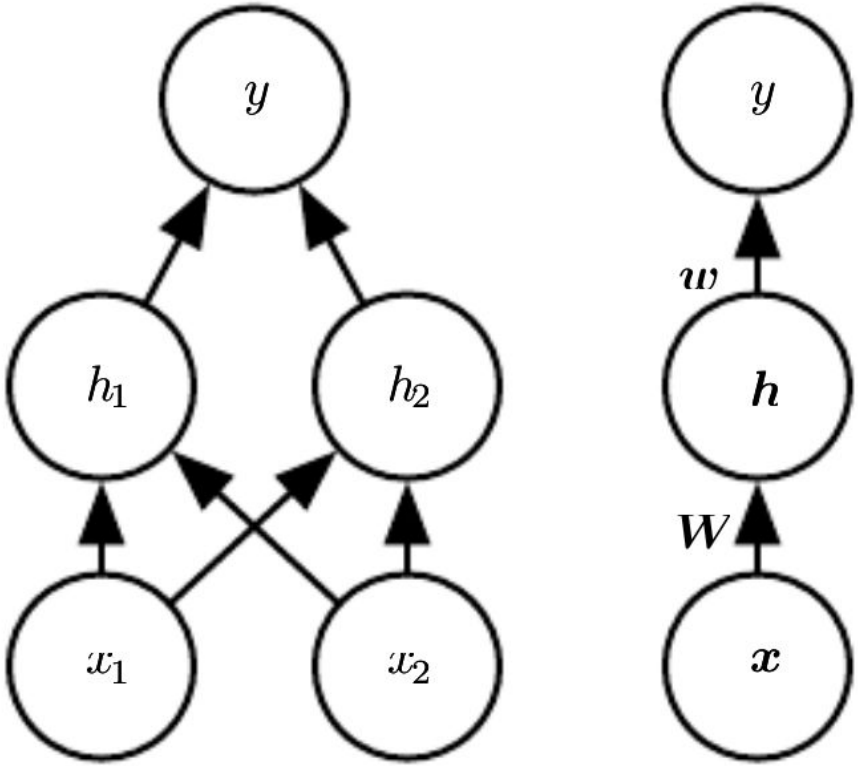
Transcription of multi-digit numbers from photographs

Tensorflow

Demo Tensorflow

Introduction to Tensorflow.

Simple XOR example.



Q?