Artificial Neural Networks

BI-ZUM

Radek Bartyzal

FIT CTU

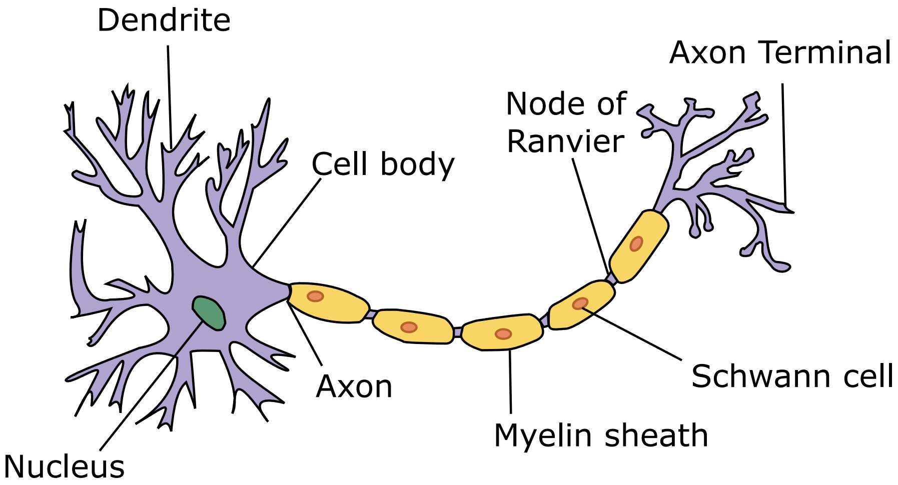
Monday 24^{th} April, 2017

Outline

1. History
   - Biological neuron
   - Perceptron
   - Sigmoid neuron
2. ANN architectures
3. Training
   - Cost function
   - Gradient descent
   - Backpropagation
4. Deep learning
   - Definition
   - Examples

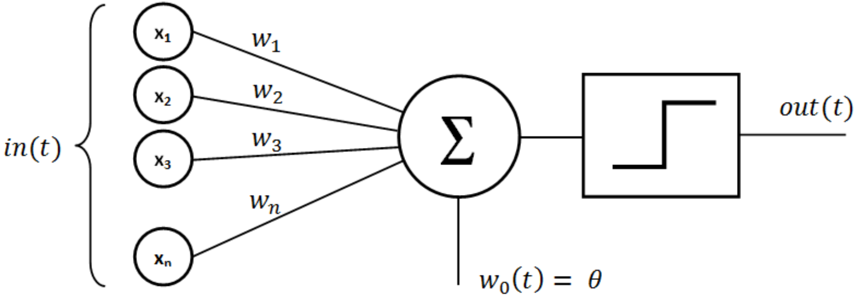
Neural style transfer. (paper, image)

1. History
   - Biological neuron
   - Perceptron
   - Sigmoid neuron
2. ANN architectures
3. Training
   - Cost function
   - Gradient descent
   - Backpropagation
4. Deep learning
   - Definition
   - Examples



Biological neuron. (source)

Perceptron



Perceptron. (source)

Perceptron

- 1950s
- binary inputs and output
- activation function = step function
- is sum of weighted inputs > threshold ?
- threshold = bias = $b = \theta$ = how willing is the perceptron (neuron) to fire
- linear separator

Perceptron

$$\sum_{i=1}^{n} x_i \cdot w_i > threshold = bias = b$$

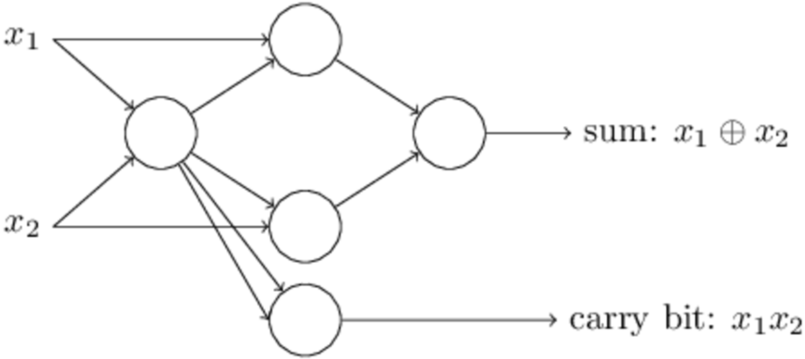
$$\sum_{i=1}^{n} x_i \cdot w_i + b > 0$$

$$\vec{x} \cdot \vec{w} + b > 0?$$

Perceptron vs XOR problem

Perceptron is a linear separator.
Why? Come again? I still don't get it :((Cake source.)

Multi Layered Perceptron



MLP representing a full adder circuit. (source)

Multi Layered Perceptron

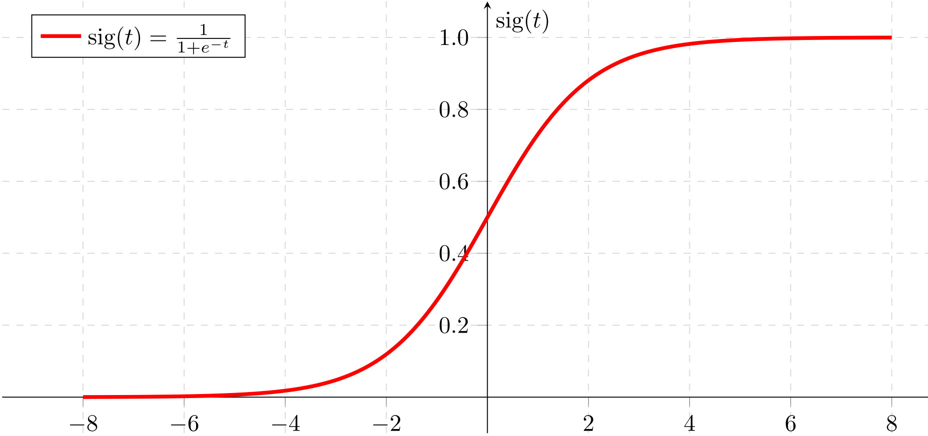
Advantages

Can simulate any logical function:
$w1 = w2 = -2, bias = 3 \implies$ NAND gate

Disadvantages

Small change in weights/biases causes either no or extremely large change in the output $\implies$ very hard to train.

Sigmoid neuron

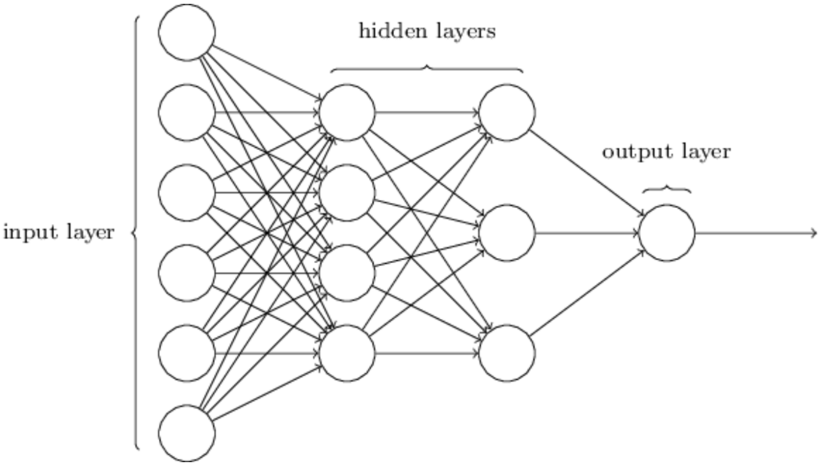


Activation function of sigmoid neuron. (source)

Sigmoid = logistic neuron

- smooth approximation of step function
- allows real inputs and outputs
- small change in weights/biases results in small change of output $\implies$ allows training = tweaking weights and biases to get desired output for each input

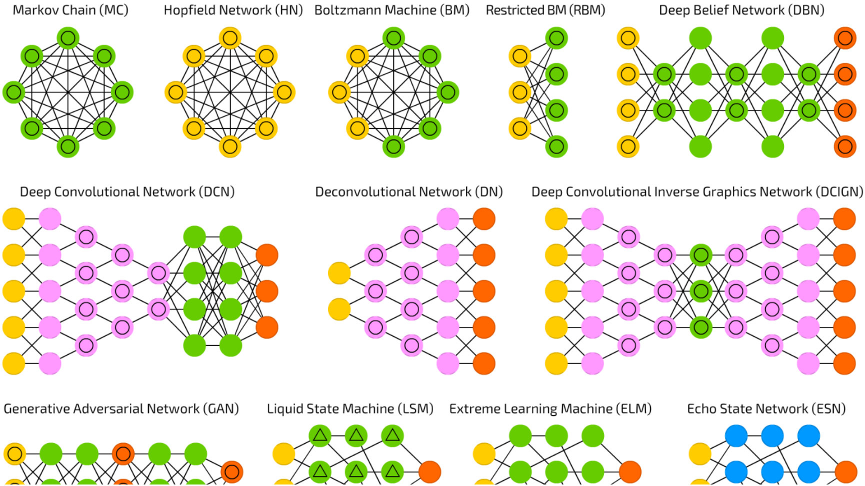
ANN architectures



Feed forward neural network. (source)

Feed forward networks

- the most basic architecture - used in almost every complex network
- input layer = inputs
- computation runs from input layer to output layer
- fully connected layers



Many other architectures exist. (source)

Training

Goal

Set weights w and biases b in such way that the network approximates the desired outputs $y(x)$ for every input x.

Training

Goal

Set weights w and biases b in such way that the network approximates the desired outputs $y(x)$ for every input x.

How to get there?

1. choose a function that tells you the error of the network = cost function
2. minimize the cost function

Training

Goal

Set weights w and biases b in such way that the network approximates the desired outputs $y(x)$ for every input x.

How to get there?

1. choose a function that tells you the error of the network = cost function
2. minimize the cost function

Cost function

Quantifies how well the network approximates the desired outputs $y(x)$ for every input x.

Training 1: Cost function

= loss function = objective function.

Quantifies how well the network approximates the desired outputs $y(x)$ for every input x.

Mean Squared Error (MSE)

w = weights

b = biases

n = number of inputs

x = one input

a = output of network for x

$y(x)$ = desired output for x

$$C(w, b) = \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Training 1: Cost function

Our goal is to minimize the cost function = move w and b in a direction that lowers the value of the cost function.

Why Mean Squared Error ?

- **smooth function of weights and biases** - even small changes of w or b result in a change of the function value
- **easily derivable** - we need derivation of the cost function to calculate the direction in which we should change the w and b

Training 2: Gradient descent

Gradient

Vector = direction in which the function increases the most in value.

Training 2: Gradient descent

Gradient

Vector = direction in which the function increases the most in value.

Gradient descent

1. calculate gradient of the cost function
2. take step in opposite direction = change w and b in a way that lowers the value of the cost function

Training 2: Gradient descent

Gradient

Vector = direction in which the function increases the most in value.

Gradient descent

1. calculate gradient of the cost function
2. take step in opposite direction = change w and b in a way that lowers the value of the cost function

Learning rate = η

How large is the step we take.

Training 3: Backpropagation

Backpropagation algorithm

Efficient way of calculating the gradient of the cost function with respect to any neuron = to any weight or bias.

Tells us in which direction should we move the weights and biases to reduce the error.

Works by propagating the error back from the output layer to the input layer.

Training summary

1. propagate input forward = get output
2. calculate error = loss = cost function
3. propagate errors back = calculate error corresponding to each neuron
4. calculate gradient of the loss function with respect to any neuron (= to any weight and bias)
5. update weights by gradient descent (or any other optimization algorithm using gradient)

Deep learning

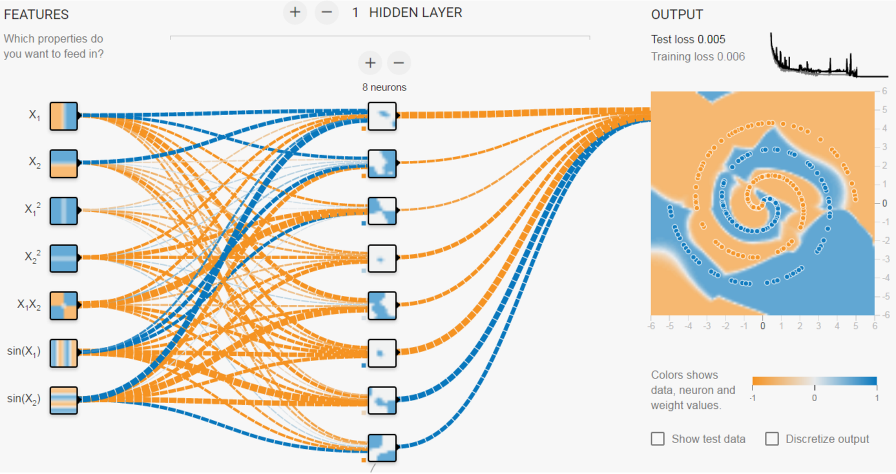
Deep learning

1. The idea of building complex concepts from simple ones.
2. Buzzword describing usage of deep neural networks.

Deep neural network

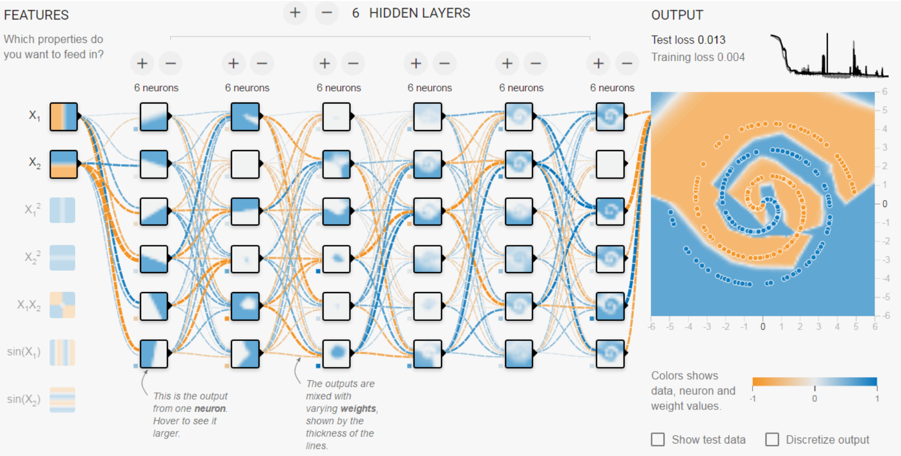
Neural network with 3 or more hidden layers.

Traditional learning



Traditional approach: complex features - simple training.

Deep learning

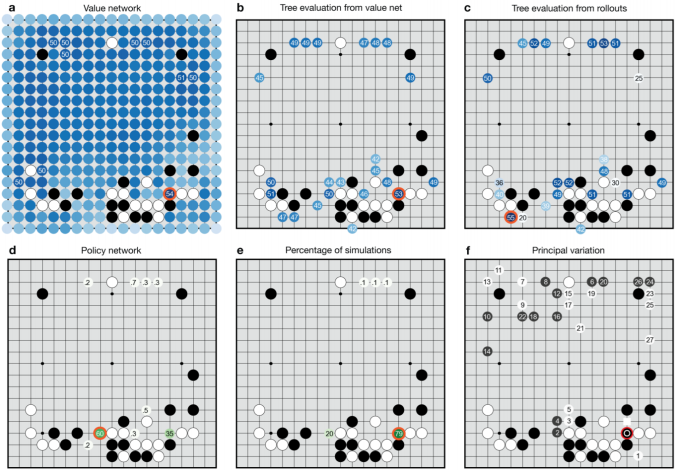


Deep learning: simple features - complex training.

Annotation of images (11/2014)

Show and Tell: A Neural Image Caption Generator. (paper)

AlphaGo (10/2015)



Mastering the Game of Go with Deep Neural Networks and Tree Search. (paper)

Image generation based on text (5/2016)

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

Generative Adversarial Text to Image Synthesis. (paper)

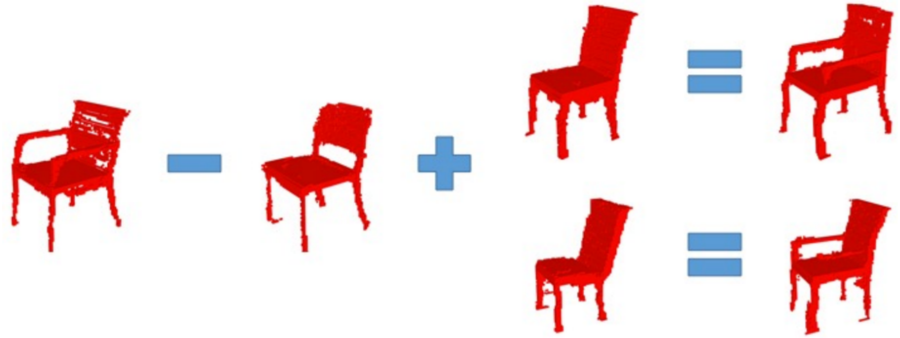
Image generation based on text (5/2016)

Generative Adversarial Text to Image Synthesis. (paper)

Images to 3D object (10/2016)

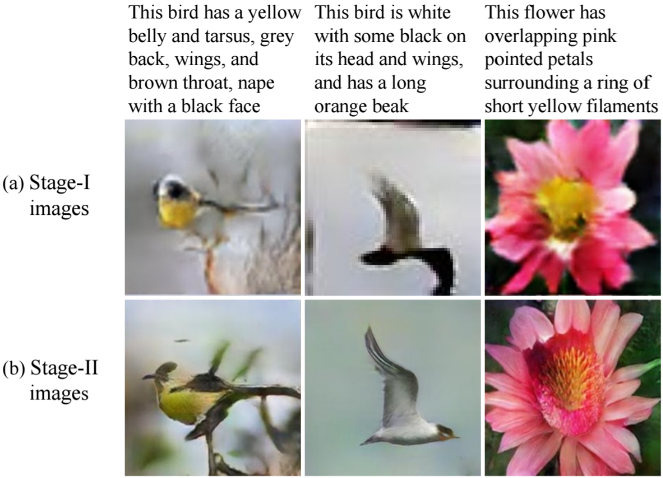
Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. (paper)

Images to 3D object (10/2016)

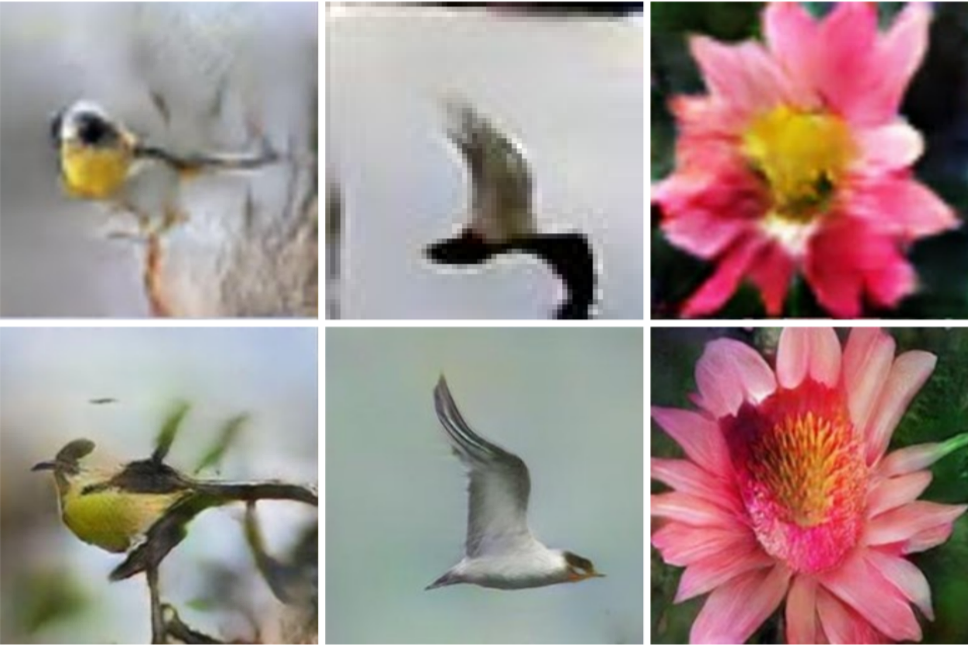


Shape arithmetic done using learned vector representations. Note added arm of the chair. (paper)

High-res image generation (12/2016)



StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. (paper)



Honorable mentions

- Distributed Representations of Words and Phrases and their Compositionality (10/2013)
- WaveNet: A Generative Model for Raw Audio (7/2016)
- Zero-Shot Translation with Googles Multilingual Neural Machine Translation System (11/2016)
- TensorFlow Playground = Play with ANN.
- OpenAI Gym = Standardized environments for Reinforcement Learning agents.
- OpenAI Universe = Game environments for AI agents.