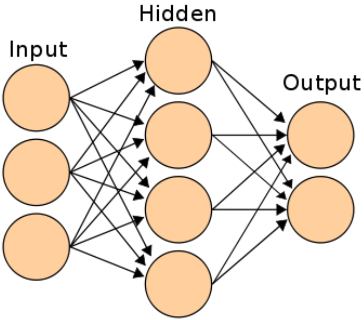
Neural networks

Janos Török

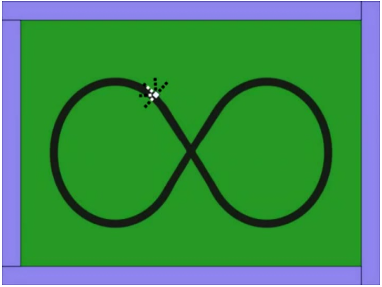
October 30, 2018

Neural networks



- Input pattern
- Output pattern
- Adaptive wights
- Approximating non-linear functions

- Machine learning
- Pattern recognition
- Handwriting
- Speech recognition



Neural networks

- Input vector I
- Output vector $O(I)$
- Transition matrix $W_{ij} \in [-1, 1]$
- Learning using a cost function
- Test goodness

Neural networks: Learning

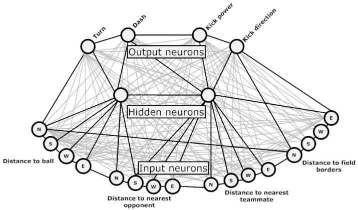
- Supervised learning
- Data training:
  - Superwised learning
  - Fitness function, energy:

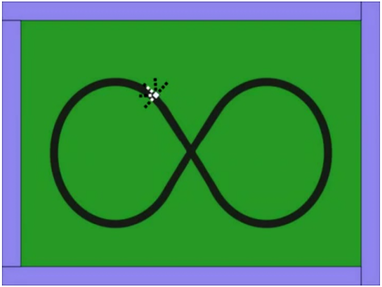
    $$E = T(I) - O(I),$$

    where $T(I)$ is the target vector for input I
  - Minimize E for available set of $\{I, I(O)\}$ pairs
  - Deep learning: many layers of neurons in the neural network
- Test goodness:
  - Use only part of $\{I, I(O)\}$ pairs for learning, the rest is for testing.
- Used for: pattern recognition, classification, etc.

Neural networks: Learning

- Reinforcement learning
- Cost function is a long time performance on an agent making decisions based on the neural network.
- Test goodness:
  - Compare with other agents which can be algorithmical or based on neural networks
- Used for: control problems, AI, complex optimization





Neural networks: Learning

- Unsupervised learning, weight is increased for neurons that fire together
- Supervised learning: cost function

Deep learning

- Literature: Introduction to deep learning: `https://www.cs.princeton.edu/courses/archive/spring16/cos495/`

Deep learning: how to

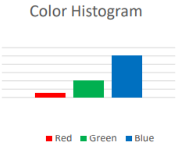
- Classification
- Perception
- Support Vector Machine
- Train the machine
- Regularization

Deep learning: Feed forward

Features

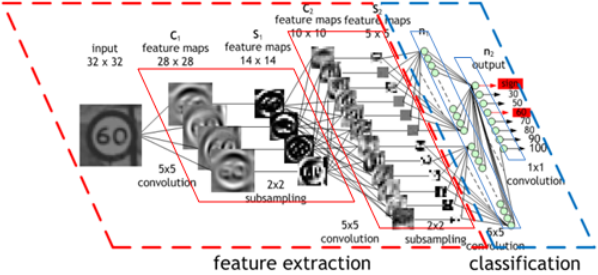
x

Extract features



build hypothesis

$y = w^T \phi(x)$



Deep learning: Feed forward

Motivation: representation learning

- Why don't we also learn $\phi(x)$?

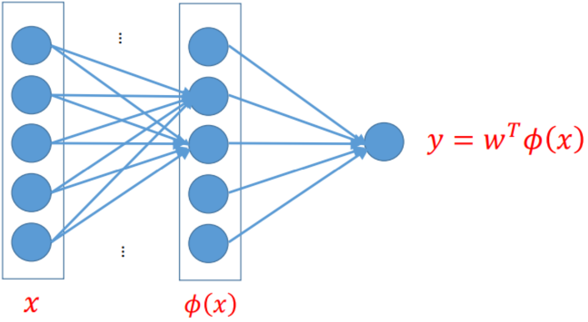
x

Learn $\phi(x)$ → $\phi(x)$ → Learn w → $y = w^T\phi(x)$

Deep learning: Feed forward

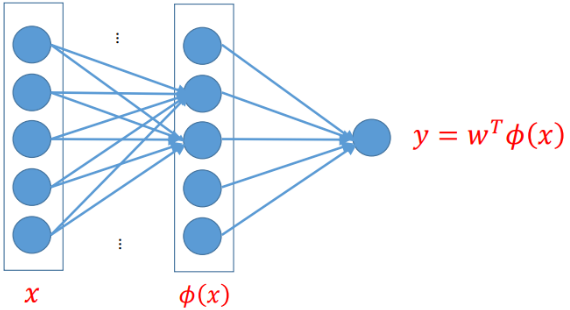
Feedforward networks

- View each dimension of $\phi(x)$ as something to be learned



Feedforward networks

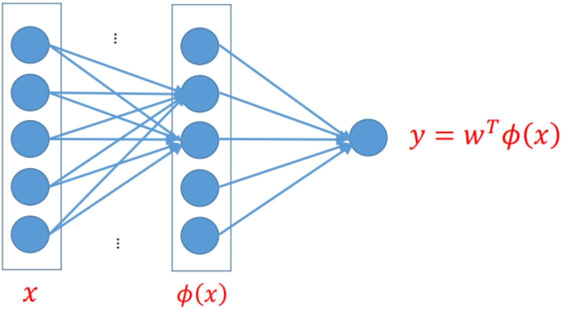
- Linear functions $\phi_i(x) = \theta_i^T x$ don't work: need some nonlinearity



Deep learning: Feed forward

Feedforward networks

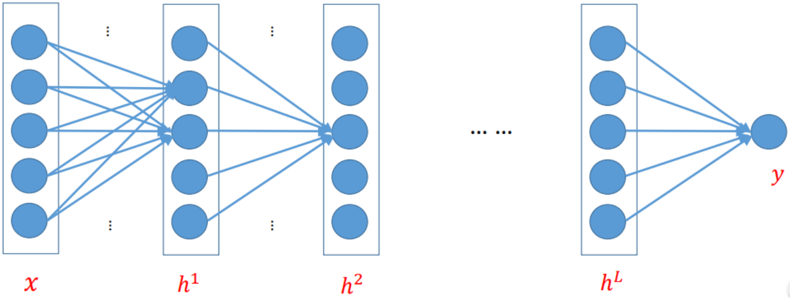
- Typically, set $\phi_i(x) = r(\theta_i^T x)$ where $r(\cdot)$ is some nonlinear function



Deep learning: Feed forward

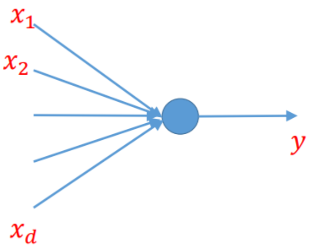
Feedforward deep networks

- What if we go deeper?



Motivation: abstract neuron model

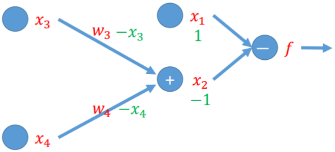
- Neuron activated when the correlation between the input and a pattern θ exceeds some threshold b
- $y = \text{threshold}(\theta^T x - b)$
  or $y = r(\theta^T x - b)$
- $r(\cdot)$ called activation function



Deep learning: Backpropagation

- Gradient of the loss is simple
  - E.g., $l(f_\theta, x, y) = (f_\theta(x) - y)^2/2$
  - $\frac{\partial l}{\partial \theta} = (f_\theta(x) - y)\frac{\partial f}{\partial \theta}$
- Key part: gradient of the hypothesis

Weights on the edges

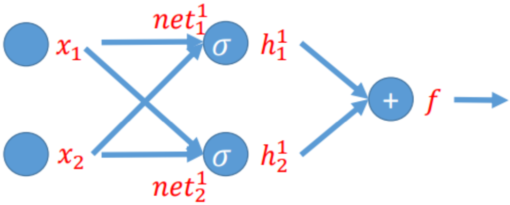


Function: $f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4)$

Gradient: $\frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial w_3} = -1 \times x_3 = -x_3$

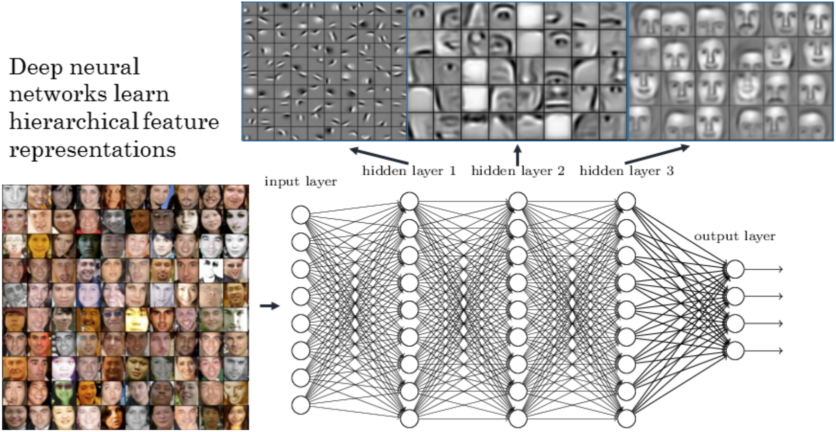
Deep learning: Backpropagation

- Forward to compute f
- Backward to compute the gradients



Deep learning: Features example

Deep neural networks learn hierarchical feature representations



Deep learning: Convolutional Neural Network