PV021: Neural networks

Tomáš Brázdil

Course organization

Course materials:

- **Main:** The lecture
- Neural Networks and Deep Learning by Michael Nielsen

  `http://neuralnetworksanddeeplearning.com/`

  (Extremely well written modern online textbook.)
- Deep learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville

  `http://www.deeplearningbook.org/`

  (A very good overview of the state-of-the-art in neural networks.)

Course organization

Evaluation:

- Project
  - teams of two students
  - implementation of a selected model + analysis of real-world data
  - implementation either in C++, or in Java **without use of any specialized libraries for data analysis and machine learning**
  - real-world data means *unprepared*, cleaning and preparation of data is part of the project!
- Oral exam
  - I may ask about anything from the lecture **including some proofs that occur only on the whiteboard!**
- (Optional) This year we will try to organize a simple data analysis competition (to try the concept). It is up to you whether to participate, or not. During the competition, you may use whatever tools for training neural networks you want.

FAQ

Q: Why English?

A: Couple of reasons. First, all resources about modern neural nets are in English, it is rather cumbersome to translate everything to Czech (combination of Czech and English is ugly). Second, to attract non-Czech speaking students to the course.

Q: Why are the lectures not recorded?

A: Apart from my personal reasons, I want you to participate actively in the lectures, i.e. to communicate with me and other students during the lectures. Also, in my opinion, online lectures should be prepared in a completely different way. I will not discuss this issue any further.

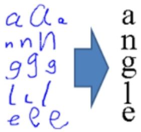
Q: Why we cannot use specialized libraries in projects?

A: In order to "touch" the low level implementation details of the algorithms. You should not even use libraries for linear algebra and numerical methods, so that you will be confronted with rounding errors and numerical instabilities.

Machine learning in general

- Machine learning = construction of systems that may learn their functionality from data

  (... and thus do not need to be programmed.)
  - spam filter
    - learns to recognize spam from a database of "labelled" emails
    - consequently is able to distinguish spam from ham
  - handwritten text reader
    - learns from a database of handwritten letters (or text) labelled by their correct meaning
    - consequently is able to recognize text



  - ...
  - and lots of much much more sophisticated applications ...

- Basic attributes of learning algorithms:
  - **representation**: ability to capture the inner structure of training data
  - **generalization**: ability to work properly on new data

Machine learning in general

Machine learning algorithms typically construct mathematical models of given data. The models may be subsequently applied to fresh data.

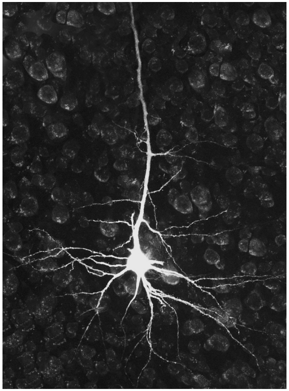
There are many types of models:

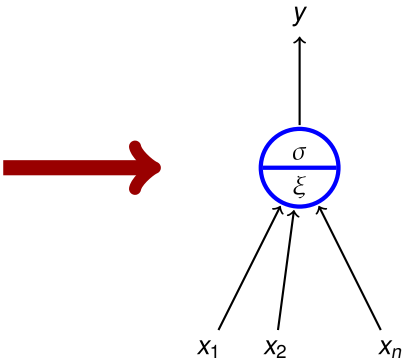
- decision trees
- support vector machines
- hidden Markov models
- Bayes networks and other graphical models
- **neural networks**
- ...

Neural networks, based on models of a (human) brain, form a natural basis for learning algorithms!

Artificial neural networks

- **Artificial neuron** is a *rough mathematical approximation* of a biological neuron.
- **(Aritificial) neural network (NN)** consists of a number of interconnected artificial neurons. "Behavior" of the network is encoded in connections between neurons.





Zdroj obrázku: `http://tulane.edu/sse/cmb/people/schrader/`

Why artificial neural networks?

Modelling of biological neural networks (computational neuroscience).

- simplified mathematical models help to identify important mechanisms
  - How a brain receives information?
  - How the information is stored?
  - How a brain develops?
  - . . .
- neuroscience is strongly multidisciplinary; precise mathematical descriptions help in communication among experts and in design of new experiments.

I will not spend much time on this area!

Why artificial neural networks?

Neural networks in machine learning.

- Typically primitive models, far from their biological counterparts (but often inspired by biology).
- Strongly oriented towards concrete application domains:
  - decision making and control - autonomous vehicles, manufacturing processes, control of natural resources
  - games - backgammon, poker, GO
  - finance - stock prices, risk analysis
  - medicine - diagnosis, signal processing (EKG, EEG, ...), image processing (MRI, roentgen, ...)
  - text and speech processing - automatic translation, text generation, speech recognition
  - other signal processing - filtering, radar tracking, noise reduction
  - . . .

I will concentrate on this area!

Important features of neural networks

- Massive parallelism
  - many slow (and "dumb") computational elements work in parallel on several levels of abstraction
- Learning
  - a kid learns to recognize a rabbit after seeing several rabbits
- Generalization
  - a kid is able to recognize a new rabbit after seeing several (old) rabbits
- Robustness
  - a blurred photo of a rabbit may still be classified as a picture of a rabbit
- Graceful degradation
  - Experiments have shown that damaged neural network is still able to work quite well
  - Damaged network may re-adapt, remaining neurons may take on functionality of the damaged ones

The aim of the course

- We will concentrate on
  - basic techniques and principles of neural networks,
  - fundamental models of neural networks and their applications.
- You should learn
  - basic models
    (multilayer perceptron, convolutional networks, recurent network (LSTM), Hopfield and Boltzmann machines and their use in pre-training of deep nets)
  - Standard applications of these models
    (image processing, speech and text processing)
  - Basic learning algorithms
    (gradient descent & backpropagation, Hebb's rule)
  - Basic practical training techniques
    (data preparation, setting various parameters, control of learning)
  - Basic information about current implementations
    (TensorFlow, CNTK)

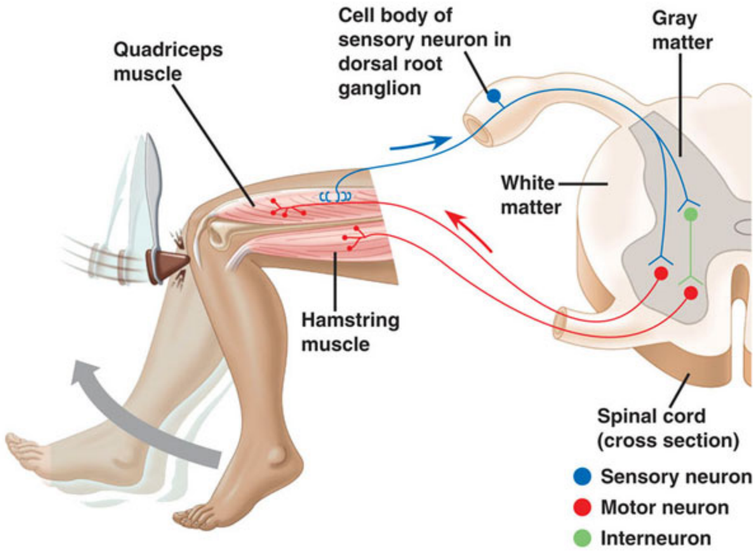
Biological neural network

- Human neural network consists of approximately 10^{11} (100 billion on the short scale) neurons; a single cubic centimeter of a human brain contains almost 50 million neurons.
- Each neuron is connected with approx. 10^4 neurons.
- Neurons themselves are very complex systems.

Rough description of nervous system:

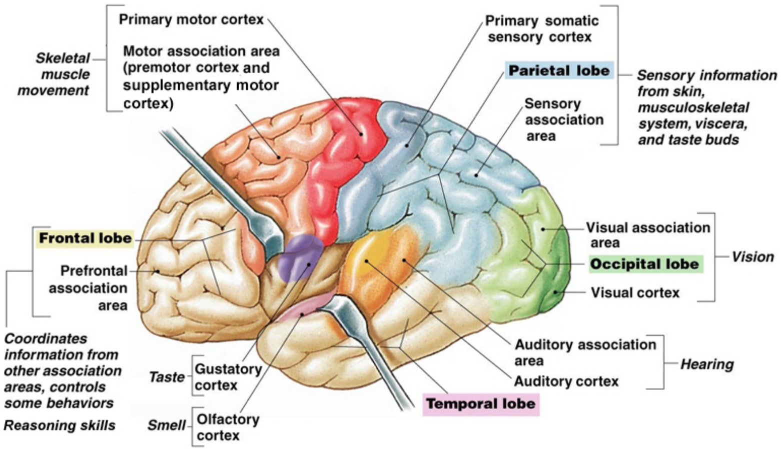
- External stimulus is received by *sensory receptors* (e.g. eye cells).
- Information is futher transfered via peripheral nervous system (PNS) to the central nervous systems (CNS) where it is processed (integrated), and subseqently, an output signal is produced.
- Afterwards, the output signal is transfered via PNS to *effectors* (e.g. muscle cells).

Biological neural network



Zdroj: N. Campbell and J. Reece; Biology, 7th Edition; ISBN: 080537146X

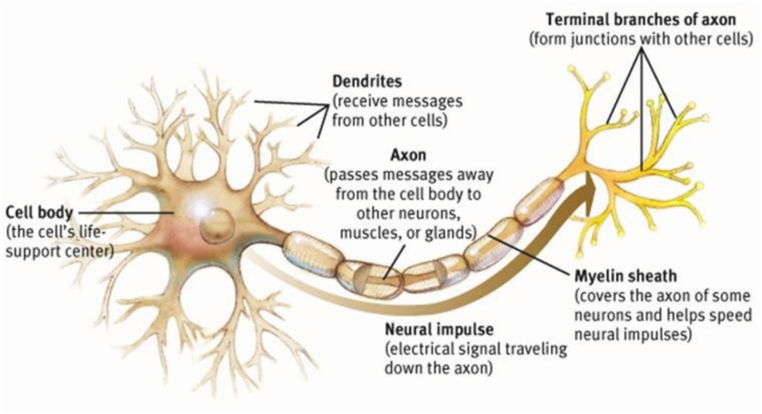
Cerebral cortex



Primary motor cortex
Motor association area (premotor cortex and supplementary motor cortex)
Skeletal muscle movement
Primary somatic sensory cortex
Parietal lobe
Sensory association area
Sensory information from skin, musculoskeletal system, viscera, and taste buds
Frontal lobe
Prefrontal association area
Coordinates information from other association areas, controls some behaviors
Reasoning skills
Visual association area
Occipital lobe
Visual cortex
Vision
Auditory association area
Auditory cortex
Hearing
Taste
Gustatory cortex
Smell
Olfactory cortex
Temporal lobe

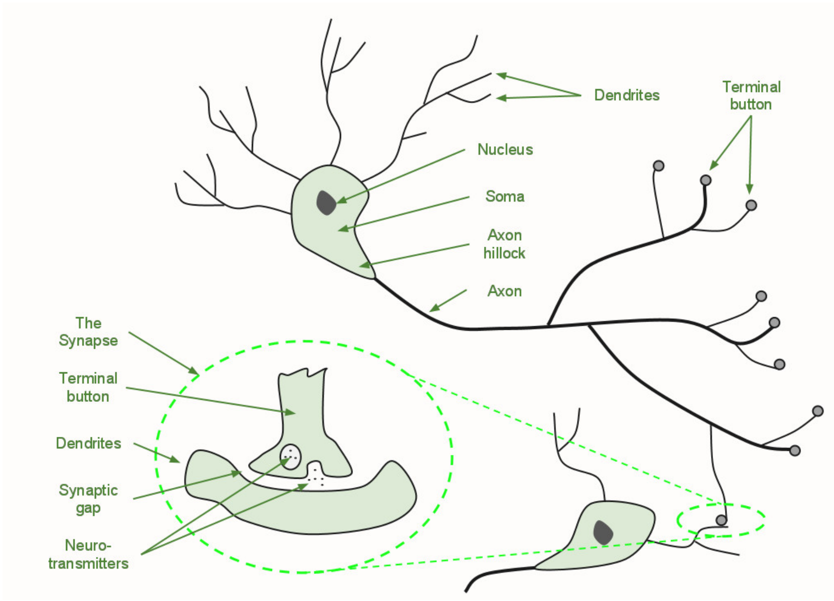
Fig. 9-15

Biological neuron


Terminal branches of axon
(form junctions with other cells)
Dendrites
(receive messages from other cells)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Cell body
(the cell's life-support center)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)
Neural impulse
(electrical signal traveling down the axon)

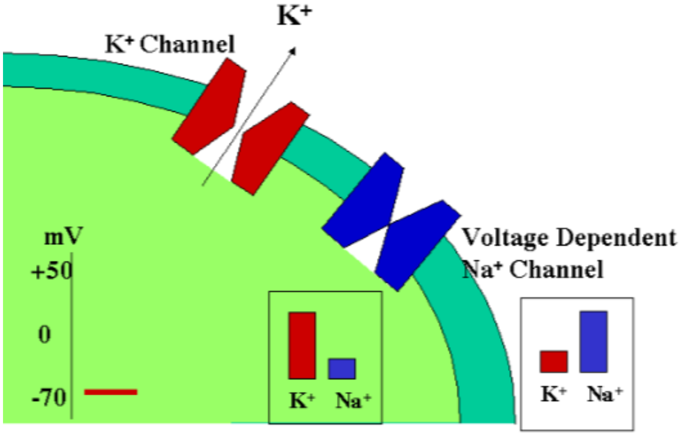
Zdroj: `http://www.web-books.com/eLibrary/Medicine/Physiology/Nervous/Nervous.htm`

Synaptic connections

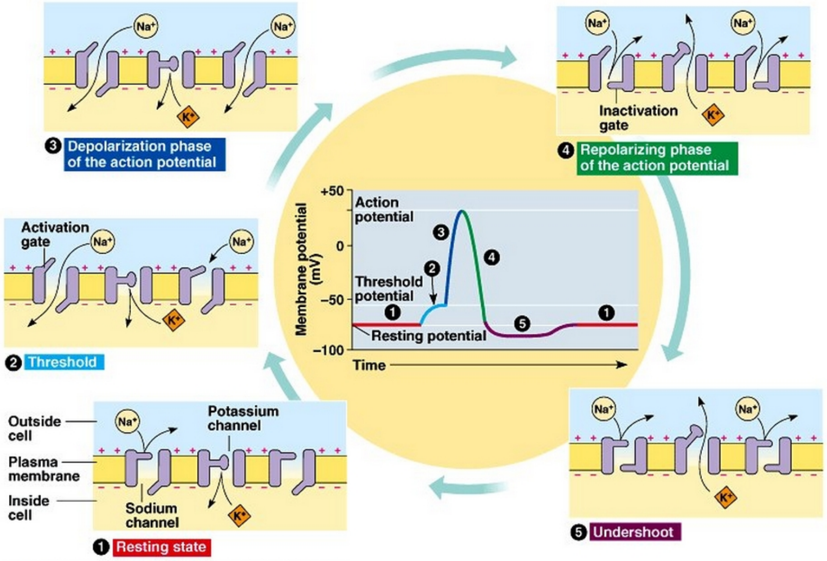


[illegible]

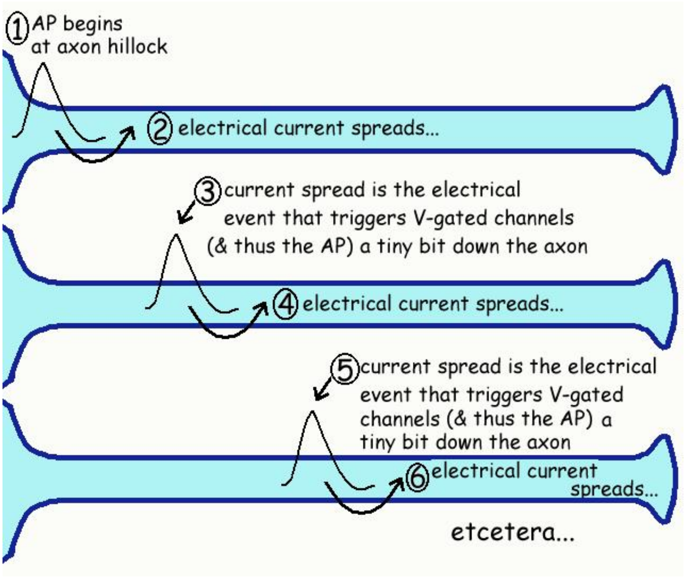
Resting potential



Action potential



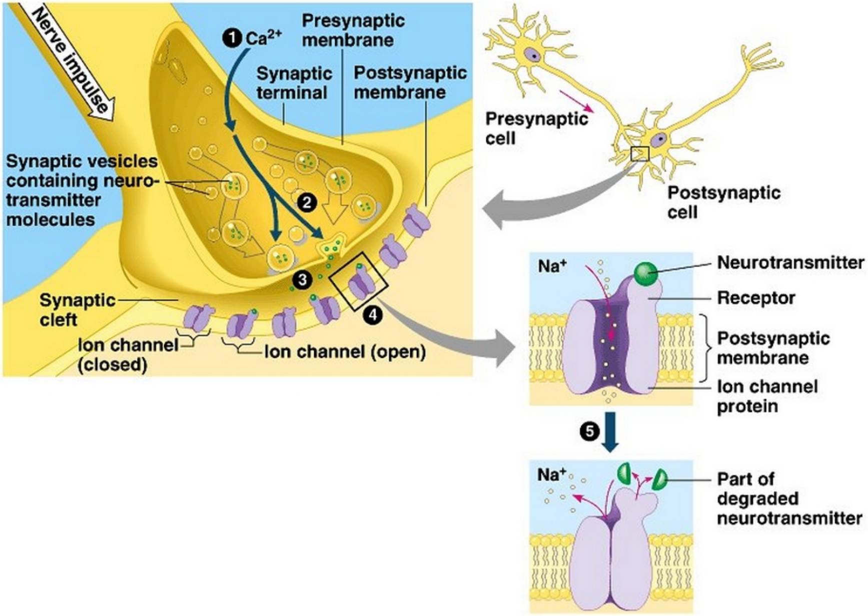
Spreading action in axon



①AP begins at axon hillock
②electrical current spreads...
③current spread is the electrical event that triggers V-gated channels (& thus the AP) a tiny bit down the axon
④electrical current spreads...
⑤current spread is the electrical event that triggers V-gated channels (& thus the AP) a tiny bit down the axon
⑥electrical current spreads...
etcetera...

Zdroj: D. A. Tamarkin; STCC Foundation Press

Chemical synapse



Summation

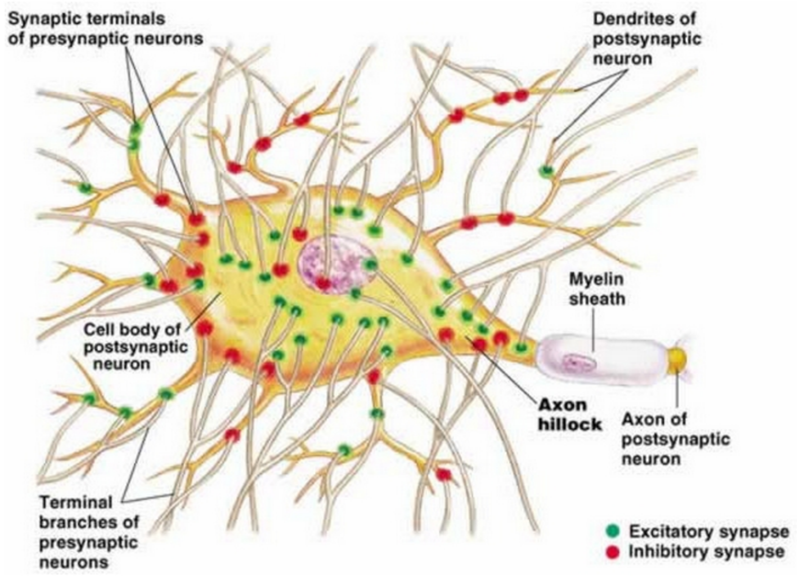
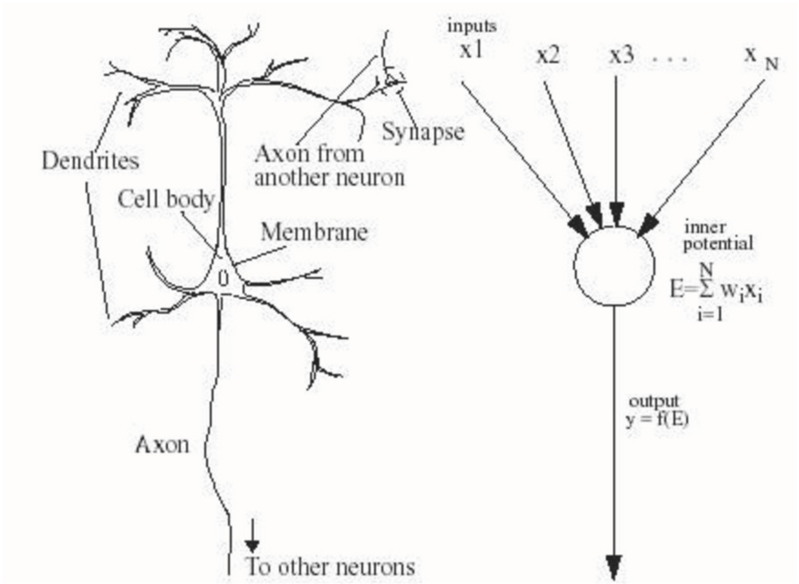
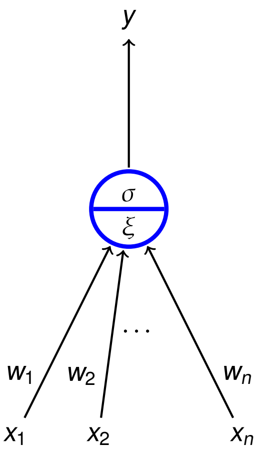

Synaptic terminals of presynaptic neurons
Dendrites of postsynaptic neuron
Myelin sheath
Cell body of postsynaptic neuron
Axon hillock
Axon of postsynaptic neuron
Terminal branches of presynaptic neurons
● Excitatory synapse
● Inhibitory synapse

Figure 48.11(a), page 972, Campbell's *Biology, 5th Edition*

Biological and Mathematical neurons



Formal neuron (without bias)

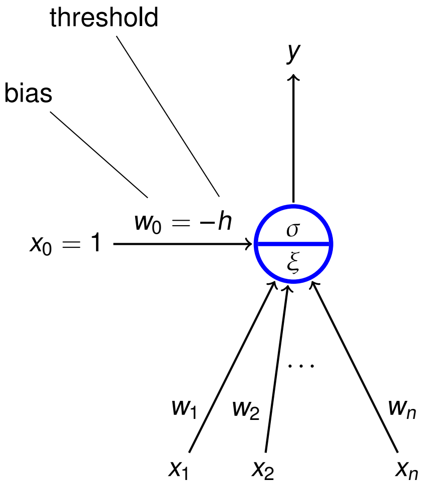


- $x_1, \dots, x_n \in \mathbb{R}$ are **inputs**
- $w_1, \dots, w_n \in \mathbb{R}$ are **weights**
- ξ is an **inner potential**;
  almost always $\xi = \sum_{i=1}^{n} w_i x_i$
- y is an **output** given by $y = \sigma(\xi)$
  where σ is an **activation function**;
  e.g. a *unit step function*

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq h\,; \\ 0 & \xi < h. \end{cases}$$

where $h \in \mathbb{R}$ is a *threshold*.

Formal neuron (with bias)



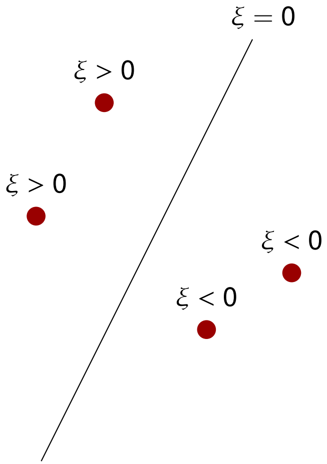
- $x_0 = 1, x_1, \dots, x_n \in \mathbb{R}$ are **inputs**
- $w_0, w_1, \dots, w_n \in \mathbb{R}$ are **weights**
- ξ is an **inner potential**; almost always $\xi = w_0 + \sum_{i=1}^{n} w_i x_i$
- y is an **output** given by $y = \sigma(\xi)$ where σ is an **activation function**;

  e.g. a *unit step function*

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0\,; \\ 0 & \xi < 0. \end{cases}$$

(The threshold h has been substituted with the new input $x_0 = 1$ and the weight $w_0 = -h$.)

Neuron and linear separation



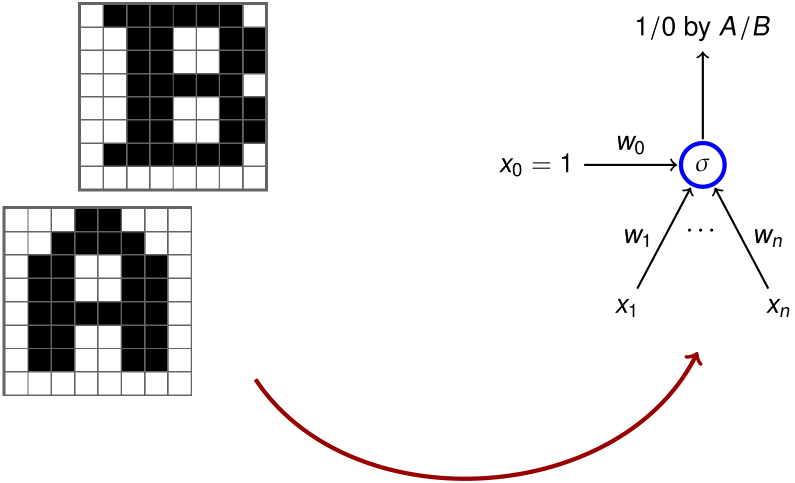
- inner potential

$$\xi = w_0 + \sum_{i=1}^{n} w_i x_i$$

determines a separation hyperplane in the *n*-dimensional **input space**

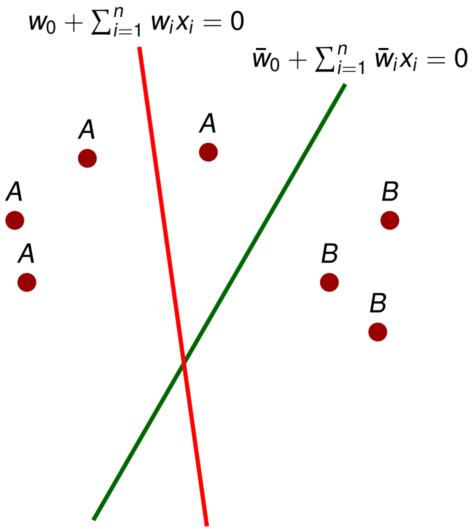
  - in 2d line
  - in 3d plane
  - $\cdots$

Neuron and linear separation



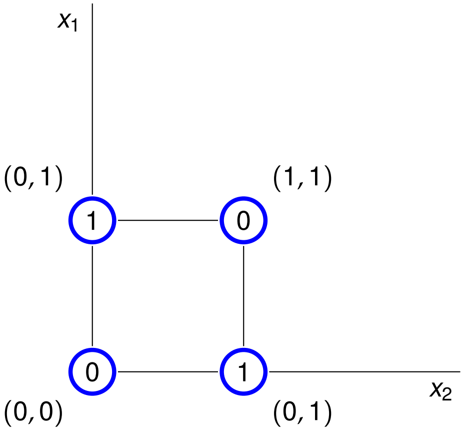
$n = 8 \cdot 8$, i.e. the number of pixels in the images. Inputs are binary vectors of dimension n (black pixel ≈ 1, white pixel ≈ 0).

Neuron and linear separation


$w_0 + \sum_{i=1}^{n} w_i x_i = 0$
$\bar{w}_0 + \sum_{i=1}^{n} \bar{w}_i x_i = 0$
A
A
A
A
B
B
B

- Red line classifies incorrectly
- Green line classifies correctly (may be a result of a correction by a learning algorithm)

Neuron and linear separation (XOR)



- No line separates ones from zeros.

Neural networks

Neural network consists of formal neurons interconnected in such a way that the output of one neuron is an input of several other neurons.

In order to describe a particular type of neural networks we need to specify:

- Architecture

  How the neurons are connected.
- Activity

  How the network transforms inputs to outputs.
- Learning

  How the weights are changed during training.

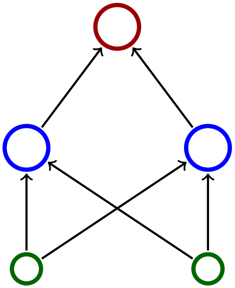
Architecture

Network architecture is given as a digraph whose nodes are neurons and edges are connections.

We distinguish several categories of neurons:

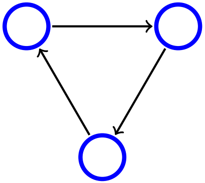
- Output neurons
- Hidden neurons
- Input neurons

(In general, a neuron may be both input and output; a neuron is hidden if it is neither input, nor output.)

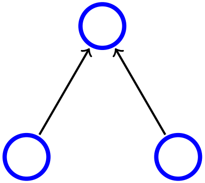


Architecture – Cycles

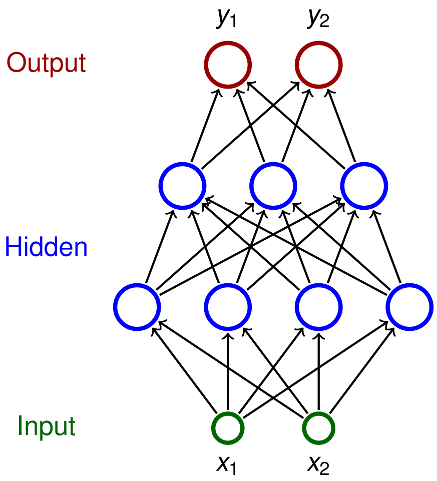
- A network is **cyclic** (recurrent) if its architecture contains a directed cycle.



- Otherwise it is **acyclic** (feed-forward)



Architecture – Multilayer Perceptron (MLP)



- Neurons partitioned into **layers**; one input layer, one output layer, possibly several hidden layers
- layers numbered from 0; the input layer has number 0
  - E.g. three-layer network has two hidden layers and one output layer
- Neurons in the *i*-th layer are connected with all neurons in the $i + 1$-st layer
- Architecture of a MLP is typically described by numbers of neurons in individual layers (e.g. 2-4-3-2)

Activity

Consider a network with n neurons, k input and ℓ output.

- **State** of a network is a vector of output values of all neurons.

  (States of a network with n neurons are vectors of $\mathbb{R}^n$)

- **State-space** of a network is a set of all states.

- **Network input** is a vector of k real numbers, i.e. an element of $\mathbb{R}^k$.
- **Network input space** is a set of all network inputs.

  (sometimes we restrict ourselves to a proper subset of $\mathbb{R}^k$)

- **Initial state**

  Input neurons set to values from the network input
  (each component of the network input corresponds to an input neuron)

  Values of the remaining neurons set to 0.

Activity – computation of a network

- **Computation** (typically) proceeds in discrete steps.
  In every step the following happens:
  1. A set of neurons is selected according to some rule.
  2. The selected neurons change their states according to their inputs (they are simply evaluated).
     (If a neuron does not have any inputs, its value remains constant.)

  A computation is **finite** on a network input $\vec{x}$ if the state changes only finitely many times (i.e. there is a moment in time after which the state of the network never changes).
  We also say that the network **stops on** $\vec{x}$.

- **Network output** is a vector of values of all output neurons in the network (i.e. an element of $\mathbb{R}^{\ell}$).
  Note that the network output keeps changing throughout the computation!

MLP uses the following selection rule:

In the i-th step evaluate all neurons in the i-th layer.

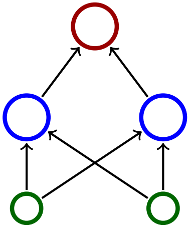
Activity – semantics of a network

Definition

Consider a network with n neurons, k input, ℓ output.
Let $A \subseteq \mathbb{R}^k$ and $B \subseteq \mathbb{R}^\ell$. Suppose that the network stops on every input of A.
Then we say that the network computes a function *$F : A \to B$ if for every network input $\vec{x}$ the vector $F(\vec{x}) \in B$ is the output of the network after the computation on $\vec{x}$ stops.*

Example 1

This network computes a function from $\mathbb{R}^2$ to $\mathbb{R}$.



Activity – inner potential and activation functions

In order to specify activity of the network, we need to specify how the inner potentials ξ are computed and what are the activation functions σ.

We assume (unless otherwise specified) that

$$\xi = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$$

here $\vec{x} = (x_1, \ldots, x_n)$ are inputs of the neuron and $\vec{w} = (w_1, \ldots, w_n)$ are weights.

There are special types of neural network where the inner potential is computed differently, e.g. as a "distance" of an input from the weight vector:

$$\xi = \left\| \vec{x} - \vec{w} \right\|$$

here $\|\cdot\|$ is a vector norm, typically Euclidean.

Activity – inner potential and activation functions

There are many activation functions, typical examples:

- Unit step function

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0\,; \\ 0 & \xi < 0. \end{cases}$$

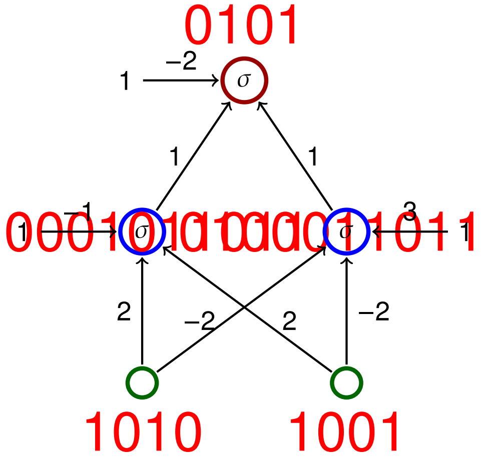
- (Logistic) sigmoid

$$\sigma(\xi) = \frac{1}{1 + e^{-\lambda \cdot \xi}}$$ here $\lambda \in \mathbb{R}$ is a *steepness* parameter.

- Hyperbolic tangens

$$\sigma(\xi) = \frac{1 - e^{-\xi}}{1 + e^{-\xi}}$$

Activity – XOR



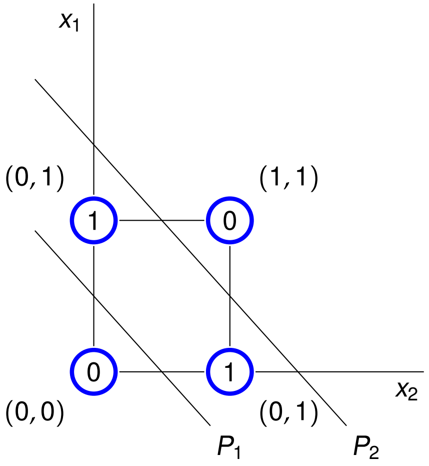
- Activation function is a unit step function

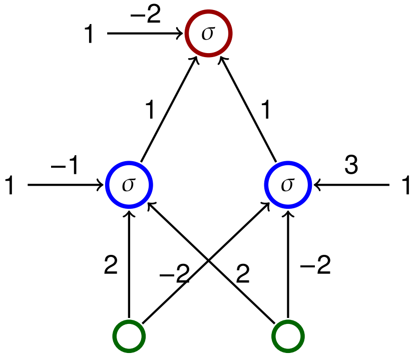
$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0\,; \\ 0 & \xi < 0. \end{cases}$$

- The network computes $XOR(x_1, x_2)$

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

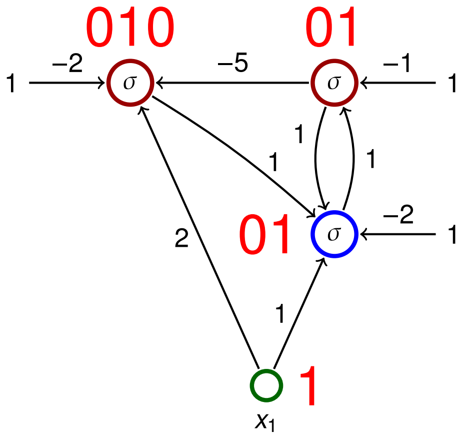
Activity – MLP and linear separation





- The line P_1 is given by $-1 + 2x_1 + 2x_2 = 0$
- The line P_2 is given by $3 - 2x_1 - 2x_2 = 0$

Activity – example



The activation function is the unit step function

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0\,; \\ 0 & \xi < 0. \end{cases}$$

The input is equal to 1

Learning

Consider a network with n neurons, k input and ℓ output.

- **Configuration** of a network is a vector of all values of weights.

  (Configurations of a network with m connections are elements of $\mathbb{R}^m$)
- **Weight-space** of a network is a set of all configurations.
- **initial configuration**

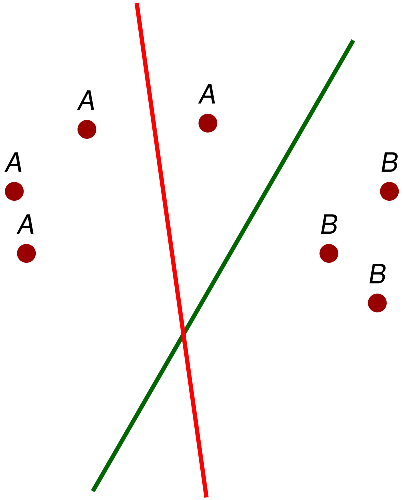
  weights can be initialized randomly or using some sophisticated algorithm

Learning algorithms

Learning rule for weight adaptation.
(the goal is to find a configuration in which the network computes a desired function)

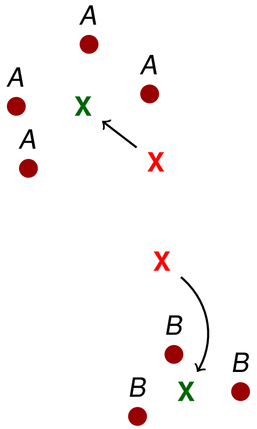
- supervised learning
  - The desired function is described using *training examples* that are pairs of the form (input, output).
  - Learning algorithm searches for a configuration which "corresponds" to the training examples, typically by minimizing an error function.
- Unsupervised learning
  - The training set contains only inputs.
  - The goal is to determine distribution of the inputs (clustering, deep belief networks, etc.)

Supervised learning – illustration



- classification in the plane using a single neuron
- training examples are of the form (point, value) where the value is either 1, or 0 depending on whether the point is either *A*, or *B*
- the algorithm considers examples one after another
- whenever an incorrectly classified point is considered, the learning algorithm turns the line in the direction of the point

Unsupervised learning – illustration



- we search for two *centres* of clusters
- red crosses correspond to potential centres before application of the learning algorithm, green ones after the application

Summary – Advantages of neural networks

- Massive parallelism
  - neurons can be evaluated in parallel
- Learning
  - many sophisticated learning algorithms used to "program" neural networks
- generalization and robustness
  - information is encoded in a distributed manned in weights
  - "close" inputs typicaly get similar values
- Graceful degradation
  - damage typically causes only a decrease in precision of results