# 02_Neural_Networks_1

Mittwoch, 2. Mai 2018 10:15

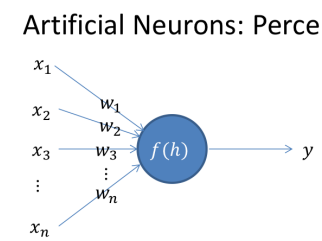
02_Neural_Networks_1

## Brain Computer Interfacing

From Neurons To Data
Session 2: Neural Networks 1
*MLP*

## Recap: Neurons

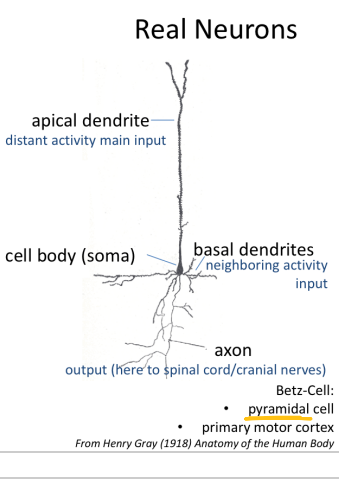
Artificial Neurons: Perceptron


$x_1$
$x_2$
$x_3$
$x_n$
$w_1$
$w_2$
$w_3$
$w_n$
$f(h)$
$y$

*Mostly logistic function:*
$f(h) = \frac{1}{1+e^{-ah}}$
… or other sigmoid function

$h = \sum_{i=1}^{n} w_i x_i - b \rightarrow y = f\left(\sum_{i=1}^{n} w_i x_i - b\right)$

Real Neurons

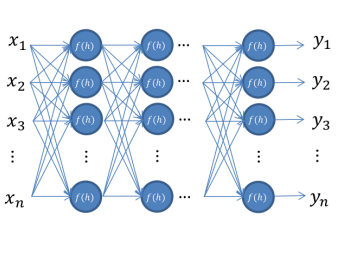

apical dendrite
distant activity main input
cell body (soma)
basal dendrites
neighboring activity input
axon
output (here to spinal cord/cranial nerves)

Betz-Cell:
- pyramidal cell
- primary motor cortex

From Henry Gray (1918) Anatomy of the Human Body

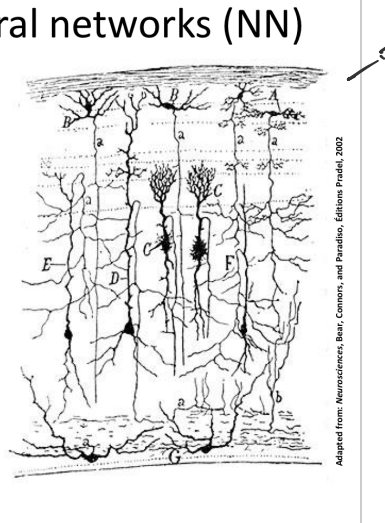
## Artificial vs. real neural networks (NN)

Straight-forward network


$x_1$
$x_2$
$x_3$
$x_n$
$f(h)$
$y_1$
$y_2$
$y_3$
$y_n$

e.g. Multi-Layer Perceptron (MLP)

- simple
- one of the most likely used



Adapted from: Neuroscience, Bear, Connors, and Paradiso, Éditions Pradel, 2002

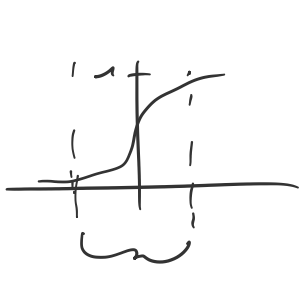
→ exemplary neurons

⇒ now shift to more complicated → MLPs hard to train with several layers

## „Calculating the output"

Artificial NN (Multi-Layer Perceptron):
- Layer-wise propagation
- Deterministic process
- Discrete calculation
- Mostly backpropagation learning

Real NN
- Forward and backward connections
- Stochastic process (that is how most people see it)
- Temporal Integration discrete time-steps
- Autonomous Learning&Forgetting? (Hebbian rule)
- Oscillations → dynamical systems with forward & backward propagation



↳ Probability of firing (interpretation)

Threshold Sites in real neuron
⇒ membran ?level? that has

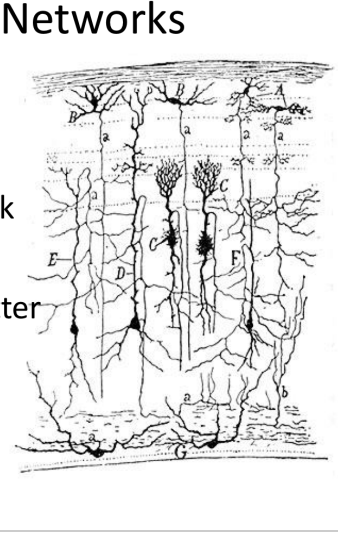
The human brain

## REAL NEURAL NETWORKS

## Real Neural Networks

to neighbours

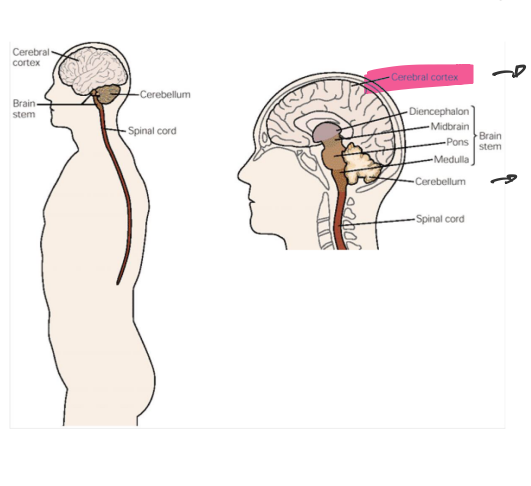
- Lateral connections
  (→lateral inhibition)
- Feedforward & Feedback
- Long range connections
- Different Neurotransmitter systems
- Specific (local) structure



Adapted from: Neuroscience, Bear, Connors, and Paradiso, Éditions Pradel, 2002

→ already in the Retina is lateral connection

## Whole Brain & Central Nervous System

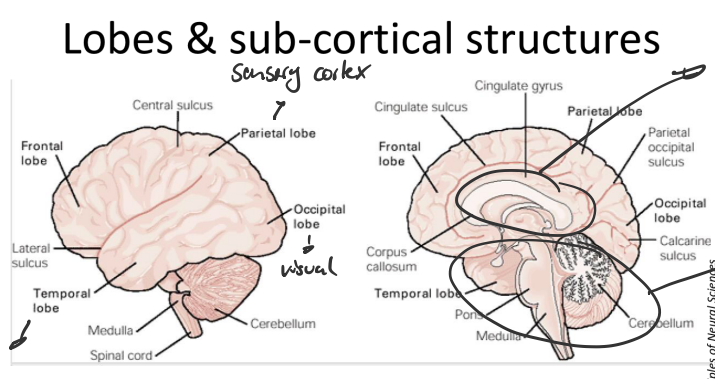

Cerebral cortex
Brain stem
Cerebellum
Spinal cord
Cerebral cortex
Diencephalon
Midbrain
Pons
Medulla
Brain stem
Cerebellum
Spinal cord

From Kandel (2000) Principles of Neural Sciences

→ We are mainly talking about that part

→ strongly specialised with motor & movement

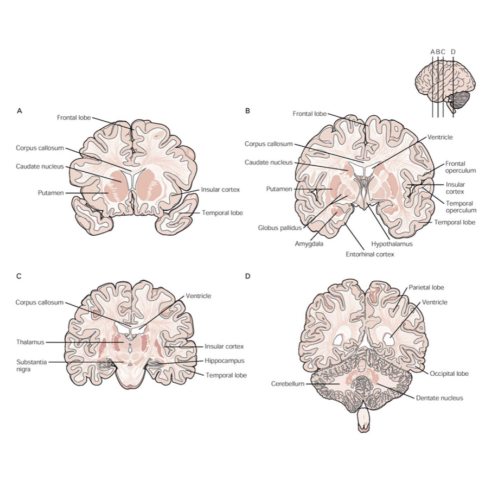
## Lobes & sub-cortical structures


Central sulcus
Parietal lobe
Frontal lobe
Occipital lobe
Lateral sulcus
Temporal lobe
Medulla
Spinal cord
Cerebellum
Cingulate sulcus
Cingulate gyrus
Parietal lobe
Parietal occipital sulcus
Frontal lobe
Occipital lobe
Corpus callosum
Calcarine sulcus
Temporal lobe
Pons
Medulla
Cerebellum
sensory cortex
visual
fear, emotion
brain stem
movement, basic stuff
memory, emotion, auditive processing

Principles of Neural Sciences

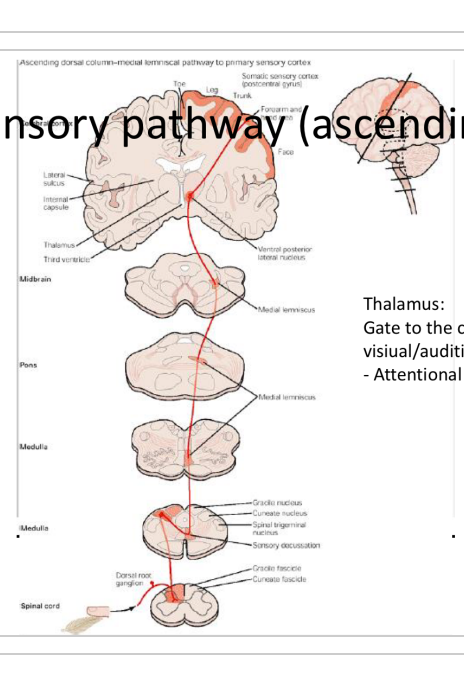
| Rough cortilcal division | |
|---|---|
| Frontal lobe: | executive functions, motor areas |
| Parietal lobe: | secondary visual perception, sensory areas |
| Temporal lobe: | memory (hippocampus), emotion (amygdala), auditive processing |
| Occipital lobe: | primary visual perception |

## Lobes & sub-cortical structures



From Kandel (2000) Principles of Neural Sciences

## Sensory pathway (ascending)

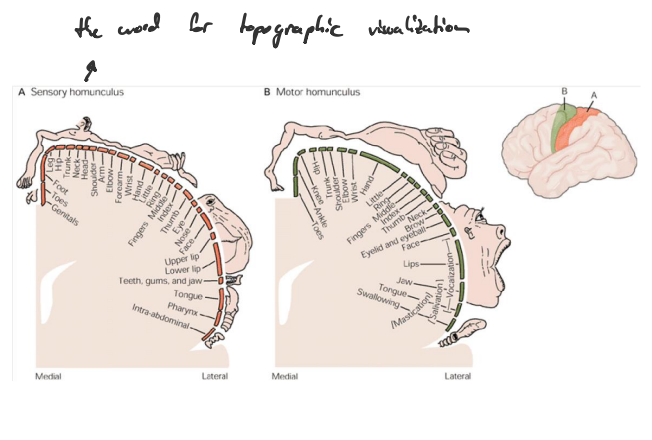


Thalamus:
Gate to the cortex (also in visiual/auditive perception)
- Attentional modulation

From Kandel (2000) Principles of Neural Sciences

## Homunculi

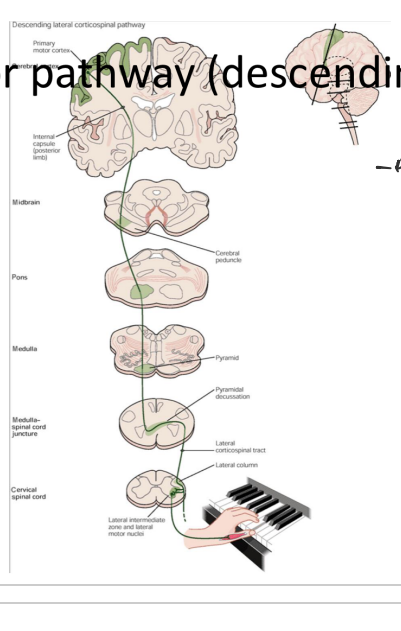
the word for topographic visualisation


A Sensory homunculus
B Motor homunculus
Medial
Lateral
Medial
Lateral

From Kandel (2000) Principles of Neural Sciences

20% brain activity is a rumor... we use always a lot of our brain.

## Motor pathway (descending)



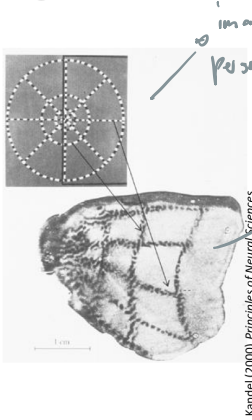
From Kandel (2000) Principles of Neural Sciences

→ brain is sleeping
↳ focus on how new sth. is. known things are handled stereotyped

## Retiontropic mapping in V1

- Proximal areas in the retina

→proximal activation in V1



From Kandel (2000) Principles of Neural Sciences

image that the person saw

processing in the cortex

## Cortical organization

The cortical distance of activation is of meaning.
Neighbouring activity can correspond to:
- Sensory mapping
- Retinotropy mapping (visual)
- Spatial mapping (orientation)
- Frequency mapping (auditory)
- Similarity of higher-level objects
- …

## Reccurrence and Oscillations

↳ loops

- The human cortex is highly reccurrent
- There are different oscillations in the brain connected to activation & attention, but also disfunction (epilepsy, migraine) and probably also metabolism/idling

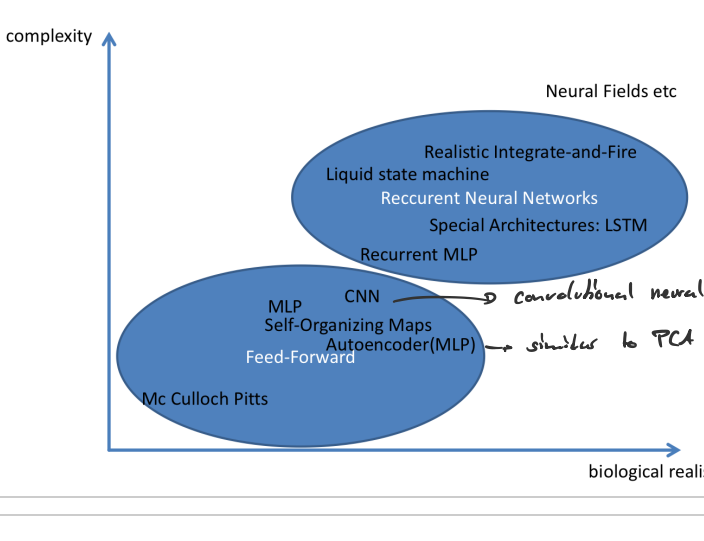
→ Neural Oscillations part of next lecture

## Associative parallel connections

- A single part of a engramme (memory) facilitates all associated/connected neurons to fire (across senses) grandma example: cake, meat smell
- Depictive examples: → everything is connected somehow
  - Lip-reading
  - A smell activating a whole visual memory
  - Musicians mentally playing instruments while listening → hand area is active while a piano player listens to piano music
- This is a basic concept of perception sorting all possible stimuli according to their statistical probability: → context-sensitivity

## ARTIFICIAL NEURAL NETWORKS

## Other artificial network models


complexity
Neural Fields etc
Realistic Integrate-and-Fire
Liquid state machine
Reccurent Neural Networks
Special Architectures: LSTM
Recurrent MLP
MLP
CNN
Self-Organizing Maps
Autoencoder(MLP)
Feed-Forward
Mc Culloch Pitts
biological realism

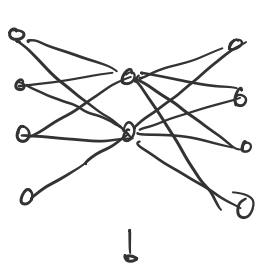
→ we see cortex as field from the distance

→ convolutional neural network

→ similar to PCA →

(Non-linear PCA)

Autoencoder → dimensionality reduction



they have to get the gist of the data

## What makes them different

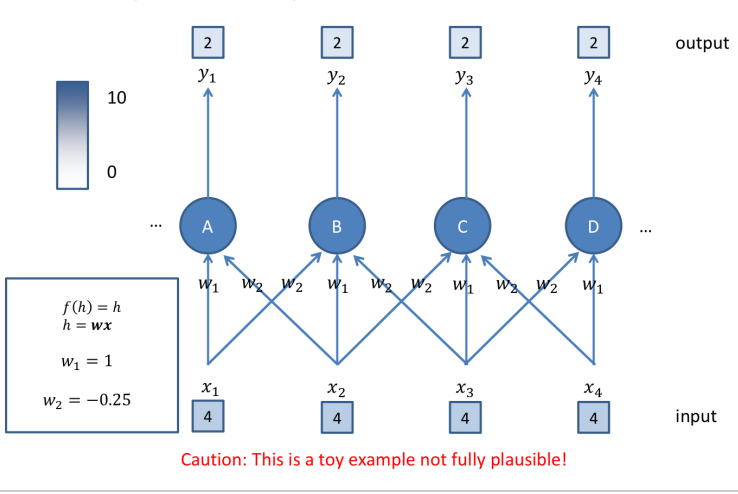
- Connections:
  - Feedforward only (e.g. MLP)
  - Feedback also (recurrent NNs)
- Activation function:
  - Sigmoid
  - Radial basis function
  - …
- Input/Output:
  - Real-valued
  - Logic [0,1]
- Designated types of learning:
  - Supervised/Reinforcement vs unsupervised
  - Complex Functions vs. Dimensionality reduction/Clustering
  - …

Lateral feedforward inhibition of the retina

## A SIMPLE EXAMPLE OF NETWORK FUNCTIONALITY

(seitlich) (Hemmung)

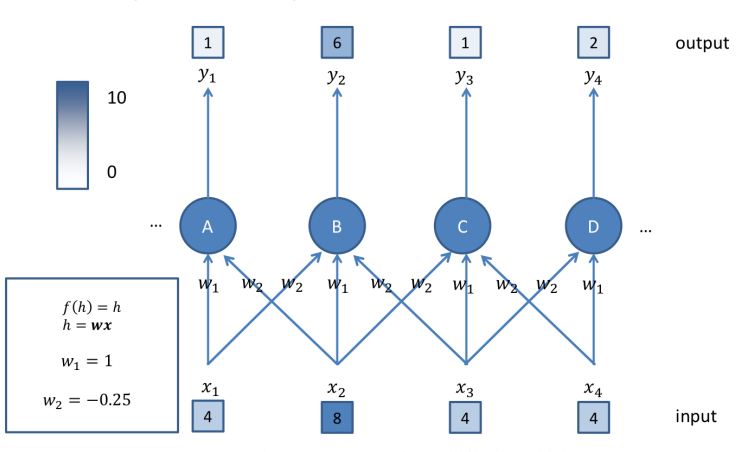
## Simple example: lateral inhibition 1D


2
2
2
2
output
$y_1$
$y_2$
$y_3$
$y_4$
10
0
A
B
C
D
$w_1$
$w_2$
$x_1$
$x_2$
$x_3$
$x_4$
4
4
4
4
input

$f(h) = h$
$h = \boldsymbol{wx}$
$w_1 = 1$
$w_2 = -0.25$

Caution: This is a toy example not fully plausible!

## Simple example: lateral inhibition 1D

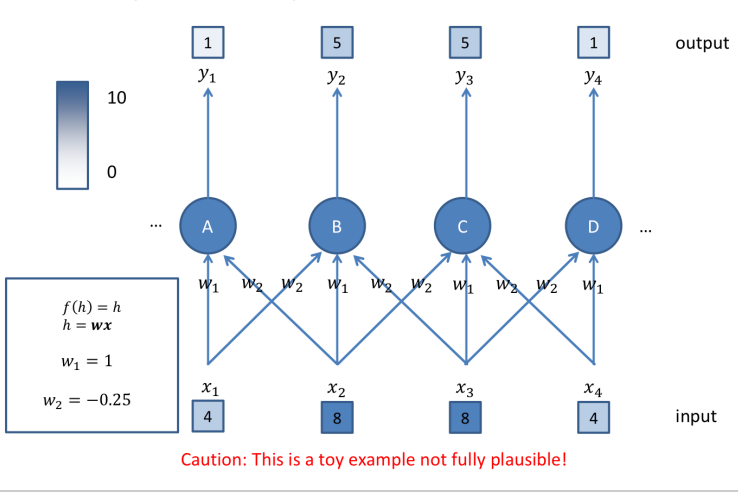

1
6
1
2
output
$y_1$
$y_2$
$y_3$
$y_4$
10
0
A
B
C
D
$w_1$
$w_2$
$x_1$
$x_2$
$x_3$
$x_4$
4
8
4
4
input

$f(h) = h$
$h = \boldsymbol{wx}$
$w_1 = 1$
$w_2 = -0.25$

Caution: This is a toy example not fully plausible!

↳ more light on this sensory cell

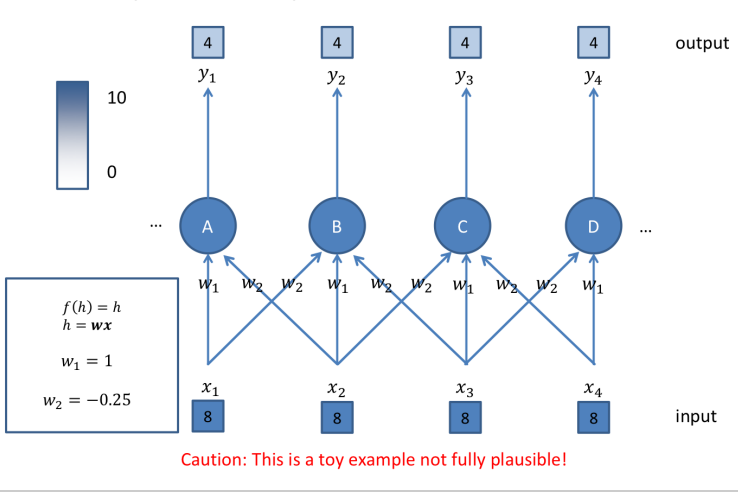
## Simple example: lateral inhibition 1D


1
5
5
1
output
$y_1$
$y_2$
$y_3$
$y_4$
10
0
A
B
C
D
$w_1$
$w_2$
$x_1$
$x_2$
$x_3$
$x_4$
4
8
8
4
input

$f(h) = h$
$h = \boldsymbol{wx}$
$w_1 = 1$
$w_2 = -0.25$

Caution: This is a toy example not fully plausible!

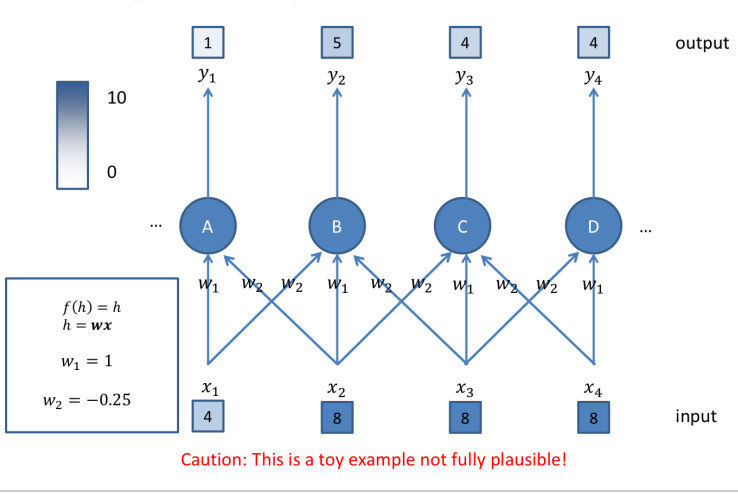
## Simple example: lateral inhibition 1D


4
4
4
4
output
$y_1$
$y_2$
$y_3$
$y_4$
10
0
A
B
C
D
$w_1$
$w_2$
$x_1$
$x_2$
$x_3$
$x_4$
8
8
8
8
input

$f(h) = h$
$h = \boldsymbol{wx}$
$w_1 = 1$
$w_2 = -0.25$

Caution: This is a toy example not fully plausible!

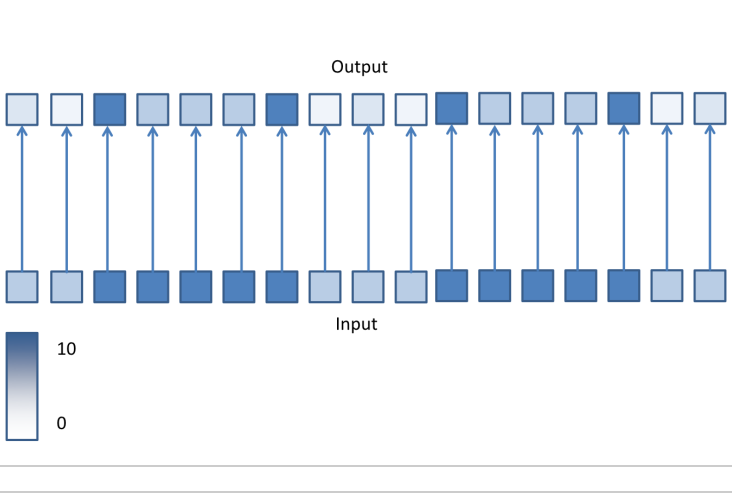
## Simple example: lateral inhibition 1D


1
5
4
4
output
$y_1$
$y_2$
$y_3$
$y_4$
10
0
A
B
C
D
$w_1$
$w_2$
$x_1$
$x_2$
$x_3$
$x_4$
4
8
8
8
input

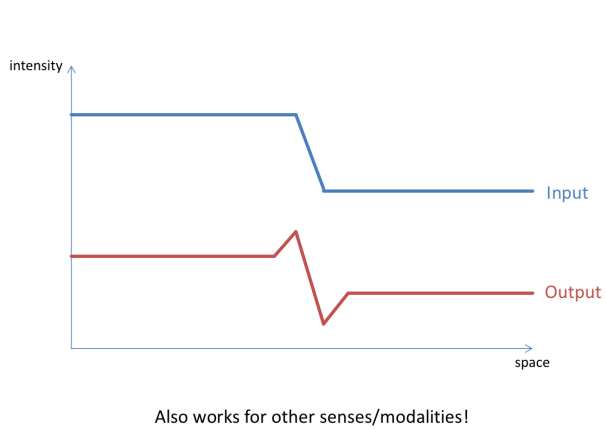
$f(h) = h$
$h = \boldsymbol{wx}$
$w_1 = 1$
$w_2 = -0.25$

Caution: This is a toy example not fully plausible!

## „Edge Detection"


Output
Input
10
0

## Contrasting of brightness

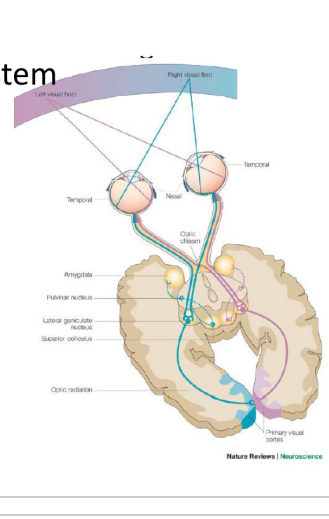


Also works for other senses/modalities!

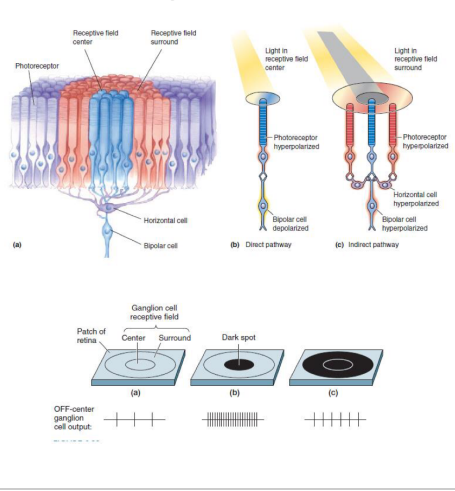
main information is where does the input change → edges

## Receptive fields

- Here: the early visual system
- Also in other senses

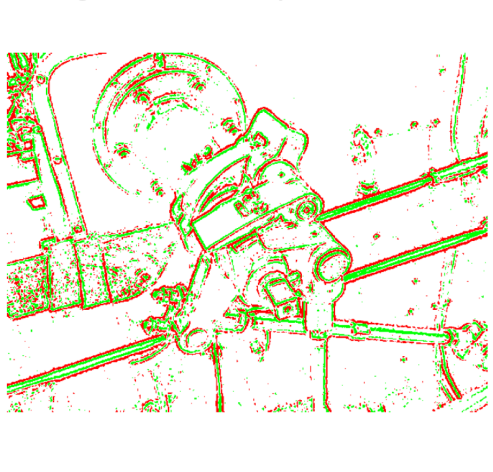


## Visual System: Retina

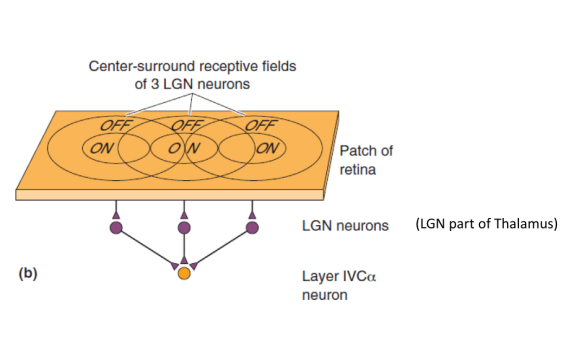


Adapted from: Neuroscience, Bear, Connors, and Paradiso, Éditions Pradel, 2002

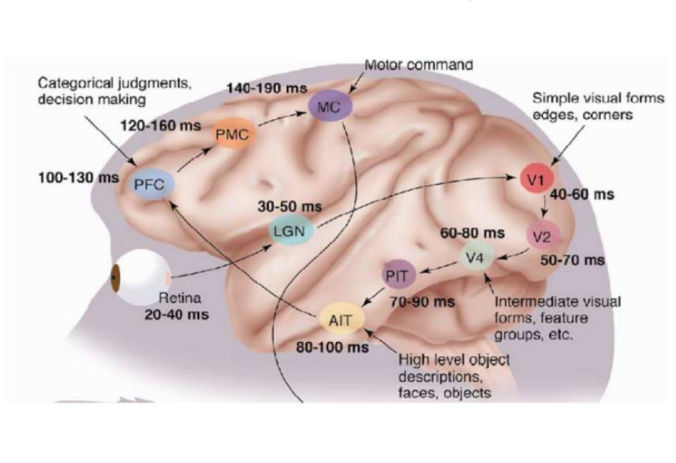
## Edge/Saliency detection



## Further processing in primary visual cortex via Thalamus


Center-surround receptive fields of 3 LGN neurons
OFF
ON
OFF
ON
OFF
ON
Patch of retina
LGN neurons
(LGN part of Thalamus)
(b)
Layer IVCα neuron

## Visual Pathway


Categorical judgments, decision making
140-190 ms
Motor command
120-160 ms
PMC
MC
Simple visual forms edges, corners
100-130 ms
PFC
V1
40-60 ms
30-50 ms
LGN
60-80 ms
V2
V4
50-70 ms
PIT
Retina 20-40 ms
70-90 ms
AIT
Intermediate visual forms, feature groups, etc.
80-100 ms
High level object descriptions, faces, objects

Adapted from: Neuroscience, Bear, Connors, and Paradiso, Éditions Pradel, 2002

## Take Home Message: visual system

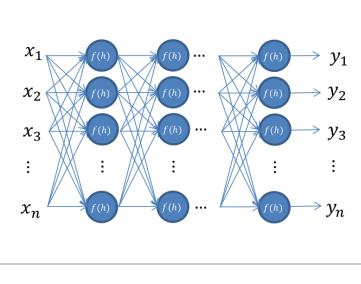
- Cortex not randomly organized
- Edges or differences between neighbors lead to activation:
  - The information in sensory input lie within comparison of the neighborhood
  - Deviation from neighborhood can also be interpreted as a deviation from expectation (change in the image)

Tools for the exercises

## HOW TO TRAIN ARTIFICIAL NEURONS

## Multi-Layer Perceptron

For the exercise, we will mainly focus on the Multi-Layer Perceptron (MLP):
- Reasonable complexity: easy to learn
- Biological interpretability



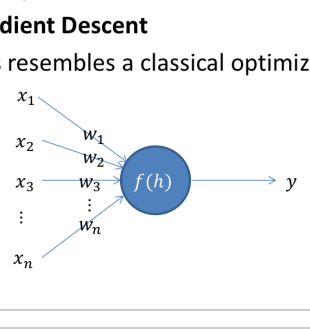
$y = f\left(\sum_{i=1}^{n} w_i x_i - b\right)$

## How can we teach a single neuron?

If we have a target function (supervised learning):
- Measure the error $E(w, b)$
- Build the gradient towards the parameters (weights $w$ and bias $b$) and move in the direction where the error decreases

→ **Gradient Descent**
- This resembles a classical optimization problem

Classical optimization problem.

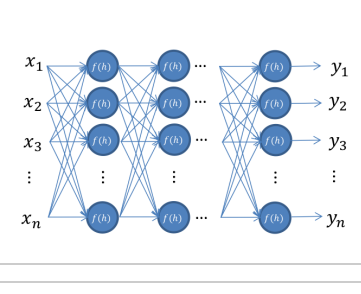


$y = f\left(\sum_{i=1}^{n} w_i x_i - b\right)$

## What with a network?

- Do gradient descent for the last layer neurons
- Propagate the error from the last layer step-by-step to the front

→Backpropagation algorithm



## One of the simplest training algorithm: Backpropagation

- Supervised learning algorithm: desired output vector **o**
- Is a result of the chain rule!

$$\frac{df(g(x))}{dx} = \frac{df(x)}{dg(x)}\frac{dg(x)}{dx}$$

- We start with the desired (training) outputs $\mathbf{o} = \begin{pmatrix} o_1 \\ \vdots \\ o_N \end{pmatrix}$

and measure e.g. the sum-of-square error $E$ in the outputs $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$

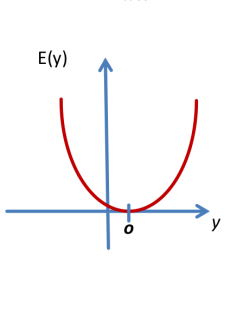
$$E(\mathbf{y}) = \frac{1}{2}\sum_i (y_i - o_i)^2 = \frac{1}{2}|\mathbf{y} - \mathbf{o}|^2$$

$L_2$-norm

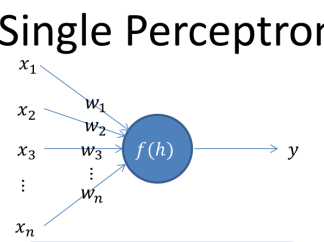
The factor $\frac{1}{2}$ is easier for derivation and not functionally essential

## Backpropagation

- Gradient descent:
  - Gradient: $\frac{dE(y(w))}{dw} = \frac{d\frac{1}{2}|y-o|^2}{dy}\frac{dy}{dw} = (y-o)\frac{dy}{dw}$
  - Descent: $w_{t+1} = w_t - \eta\frac{dE(y)}{dw}$
    - w: weight
    - y: output
    - o: training (target) output
    - t: discrete time point
    - η: learning rate



## Single Perceptron



*Mostly logistic function:*
$f(h) = \frac{1}{1+e^{-ah}}$

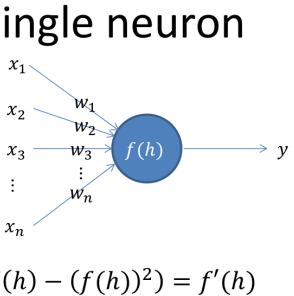
$h = \sum_{i=1}^{n} w_i x_i - b \rightarrow y = f\left(\sum_{i=1}^{n} w_i x_i - b\right)$

$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ $\quad$ $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ $\quad$ $\mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ $\quad$ $\mathbf{y} = f(\mathbf{w} \cdot \mathbf{x} - \mathrm{b})$

From now on: bold variables are vectors!

## Error gradient in a single neuron

$y = f(h) = f\left(\sum_{i=1}^{n} w_i x_i - b\right) = f(\mathbf{w} \cdot \mathbf{x} - b)$



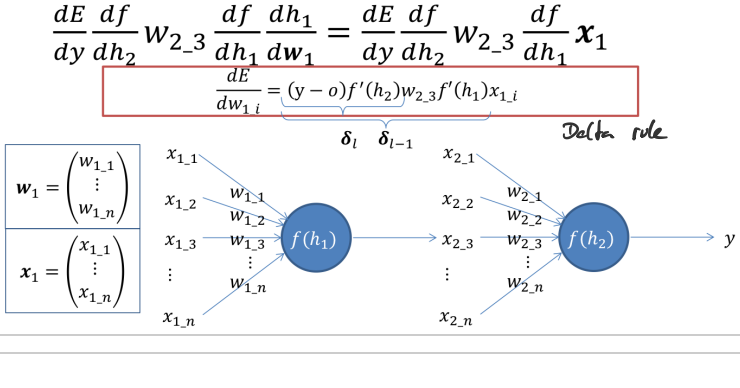
- $\frac{dE}{dw_i} = \frac{dE}{dy}\frac{dy}{dw_i} = \frac{dE}{dy}\frac{df}{dh}\frac{dh}{dw_i}$
- $\frac{df}{dh} = \frac{d}{dh}\frac{1}{1+e^{-ah}} = a\frac{e^{-ah}}{(1+e^{-ah})^2} = a(f(h) - (f(h))^2) = f'(h)$
- $\frac{dE}{d\mathbf{w}} = \boxed{\frac{dE}{dy}}\boxed{\frac{df}{dh}}\boxed{\frac{dh}{d\mathbf{w}}} = \boxed{(y-o)}\boxed{f'(h)}\boxed{\mathbf{x}}$
- $\frac{dE}{db} = \boxed{\frac{dE}{dy}}\boxed{\frac{df}{dh}}\boxed{\frac{dh}{db}} = \boxed{(y-o)}\boxed{f'(h)}\boxed{-1}$ → equals constant input

## Error backpropagation two neurons

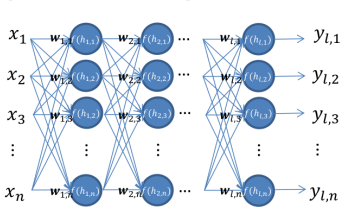
- $\frac{dE}{d\mathbf{w}_1} = \frac{dE}{dy}\frac{dy}{d\mathbf{w}_1} = \frac{dE}{dy}\frac{df}{dh_2}\frac{dh_2}{d\mathbf{w}_1} =$
$\frac{dE}{dy}\frac{df}{dh_2}\frac{d(w_{2\_3}x_{2\_3})}{d\mathbf{w}_1} = \frac{dE}{dy}\frac{df}{dh_2}w_{2\_3}\frac{df(h_1)}{d\mathbf{w}_1} =$
$\frac{dE}{dy}\frac{df}{dh_2}w_{2\_3}\frac{df}{dh_1}\frac{dh_1}{d\mathbf{w}_1} = \frac{dE}{dy}\frac{df}{dh_2}w_{2\_3}\frac{df}{dh_1}\mathbf{x}_1$

$$\boxed{\frac{dE}{dw_{1\_i}} = \underbrace{\underbrace{(y-o)f'(h_2)}_{\delta_l}w_{2\_3}f'(h_1)}_{\delta_{l-1}}x_{1\_i}}$$

Delta rule


$\mathbf{w}_1 = \begin{pmatrix} w_{1\_1} \\ \vdots \\ w_{1\_n} \end{pmatrix}$
$\mathbf{x}_1 = \begin{pmatrix} x_{1\_1} \\ \vdots \\ x_{1\_n} \end{pmatrix}$
$x_{1\_1}$
$x_{1\_2}$
$x_{1\_3}$
$x_{1\_n}$
$w_{1\_1}$
$w_{1\_2}$
$w_{1\_3}$
$w_{1\_n}$
$f(h_1)$
$x_{2\_1}$
$x_{2\_2}$
$x_{2\_3}$
$x_{2\_n}$
$w_{2\_1}$
$w_{2\_2}$
$w_{2\_3}$
$w_{2\_n}$
$f(h_2)$
$y$

## Multi-Layer Perceptron: formalism



$\mathbf{h}_l = \begin{pmatrix} h_{l,1} \\ \vdots \\ h_{l,n} \end{pmatrix}$

$\mathbf{y}_l = f(\mathbf{h}_l) = f\begin{pmatrix} h_{l,1} \\ \vdots \\ h_{l,n} \end{pmatrix} = f\begin{pmatrix} \mathbf{w}_{l,1}f(\mathbf{h}_{l-1}) \\ \vdots \\ \mathbf{w}_{l,n}f(\mathbf{h}_{l-1}) \end{pmatrix} = f(\mathbf{W}_l f(\mathbf{h}_{l-1})) = f\left(\mathbf{W}_l f\begin{pmatrix} \mathbf{w}_{l-1,1}f(\mathbf{h}_{l-2}) \\ \vdots \\ \mathbf{w}_{l-1,n}f(\mathbf{h}_{l-2}) \end{pmatrix}\right)$
$= f(\mathbf{W}_l f(\mathbf{W}_{l-1} f(\mathbf{h}_{l-2}))) = \cdots = g(\mathbf{x})$

$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$; $\mathbf{y}_l = \begin{pmatrix} y_{l,1} \\ \vdots \\ y_{l,n} \end{pmatrix} = f(\mathbf{w}_{l,i} \cdot \mathbf{y}_{l-1} - \mathbf{b}_l)$; $\mathbf{w}_{l,i} = \begin{pmatrix} w_{l,i,1} \\ \vdots \\ w_{l,i,n} \end{pmatrix}$; $\mathbf{W}_l = (\mathbf{w}_{l,1} \ \cdots \ \mathbf{w}_{l,N})^T$

## The Hadamard product

- For simpler equations, we introduce the Hadamard product, which is a simple element-wise matrix and vector operation (.* in matlab, * in python) for same dimensional A and B matrices/vectors:

$$A \circ B = (A)_{i,j}(B)_{i,j}$$

## Delta Rule for an MLP

$\boldsymbol{o}$ … desired output of layer l (=training data)

- Last layer l neuron i:

$$\frac{dE(\mathbf{y}_l)}{d\mathbf{w}_{l,i}} = (\mathbf{y}_l - \boldsymbol{o}) \circ f'(\mathbf{h}_l)\mathbf{y}_{l-1}$$

- Last hidden layer l-1 neuron k :

$$\frac{dE}{d\mathbf{w}_{l-1,k}} = (\mathbf{y}_l - \boldsymbol{o}) \circ f'(\mathbf{h}_l) \cdot [\mathbf{W}_l]_{(k)} \circ f'(\mathbf{h}_{l-1})\mathbf{y}_{l-2}$$

The kth column of $\mathbf{W}_l$

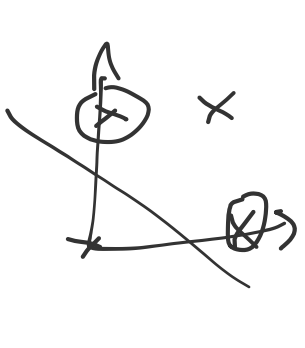
- 2nd last hidden layer l-2 neuron m :

$$\frac{dE}{d\mathbf{w}_{l-2,m}} = \underbrace{\underbrace{\underbrace{(\mathbf{y}_l - \boldsymbol{o}) \circ f'(\mathbf{h}_l)}_{\boldsymbol{\delta}_l}\mathbf{W}_l \circ f'(\mathbf{h}_{l-1})}_{\boldsymbol{\delta}_{l-1}} \cdot [\mathbf{W}_{l-1}]_{(m)} \circ f'(h_{l-2,m})}_{\delta_{l-2,m}}\mathbf{y}_{l-3}$$

Gradient Descent for any layer l, neuron i:

$$\mathbf{w}_{l,i}(t+1) = \mathbf{w}_{l,i}(t) - \eta\frac{dE}{d\mathbf{w}_i} = \mathbf{w}_{l,i}(t) - \eta\delta_{l,i}\mathbf{y}_{i-1}$$

!AND
OR
AND



## How to train a network (supervised)

- Initialize the weights (mostly random but can also be suiting your special task)
- Repeat the following until convergence:
  - Present a sample of the input
  - build the error gradient based on designated output and actual output
  - Adopt weights & bias with gradient descent:
    $w_{t+1} = w_t - \eta\frac{dE(y)}{dw}$; $b_{t+1} = b_t - \eta\frac{dE(y)}{db}$

## Summary: Delta Rule for an MLP

- Backpropagation of the error from the last layer l:
  - $\boldsymbol{\delta}_l = (\mathbf{y}_l - \boldsymbol{o}) \circ f'(\mathbf{h}_l) = \begin{pmatrix} \delta_{l\_1} \\ \vdots \\ \delta_{l\_n} \end{pmatrix}$
  - $\boldsymbol{\delta}_{l-1} = \boldsymbol{\delta}_l \mathbf{W}_l \circ f'(\mathbf{h}_{l-1}) = \begin{pmatrix} \delta_{l-1\_1} \\ \vdots \\ \delta_{l-1\_n} \end{pmatrix}$
  - $\delta_{l-k\_i} = \boldsymbol{\delta}_{l-k+1} \cdot [\mathbf{W}_{l-k}]_{(i)} f'(h_{l-k\_i})$
- Weight update rule:
  - $\mathbf{w}_{l,i}(t+1) = \mathbf{w}_{l,i}(t) - \eta\frac{dE}{d\mathbf{w}_{l,i}} = \mathbf{w}_i(t) - \eta\delta_{l-k,i}\mathbf{y}_{l-1}$
  - $\mathbf{w}_{1,i}(t+1) = \mathbf{w}_{1,i}(t) - \eta\frac{dE}{d\mathbf{w}_{1,i}} = \mathbf{w}_{1,i}(t) - \eta\delta_{1,i}\boldsymbol{x}$ for the input layer
  - Learning rate $\eta \approx \{0.01..0.1\}$
    (depending on error function shape, sample size and other aspects)

## How many iterations? Which learning rate?

- Depending on the complexity of target function, the complexity of the network and the learning rate
- If the error can be measured, a stopping criteria is mostly used:
  - If the decrease in error is low and/or
  - the overall error is low enough

  the network is expected to be converged to something meaningful.

## THANKS FOR YOUR ATTENTION