Artificial Neural Networks: Lecture 1
Simple Perceptrons for Classification

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:
- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. Simple perceptrons for classification
2. Backprop and multilayer perceptron
3. Statistical Classification by deep networks
4. Deep learning: regularization and tricks of the trade *Miniproject1*
5. Error landscape and optimization methods for deep networks
6. Convolutional networks
7. Sequence predictions and Recurrent networks *Miniproject2a*
8. Reinforcement learning1: Bellman and SARSA *Miniproject2b*
9. Reinforcement learning2: variants of SARSA
10. Reinforcement learning3: Policy Gradient
11. Deep Reinforcement learning
12. Reinforcement learning and the Brain

Miniproject1: handout March 15;
submission April 29
Miniproject 2a/2b: handout April 29;
submission: choose May27 or June 3

Weakly sessions as follows:

- 10:15 in lecture hall: Discussions with TA
- 1 Exercise of previous week discussed from 10:15 -10:40 in separate Exercise room
- break 5 min
- Lecture 1 from 10:45-11:35 (approximately)
- Break 8 min
- Lecture 2 from 11:43-12:35
- Break 5 min
- Discussion of miniproject from 12:40-12:55

Each lecture typically includes one 'in-class' exercise

TA's this year: Berfin Simsek, Nicolas El Maalouly,
Chiara Gastaldi, Florian Colombo, Bernd Illing

Previous 3 slides.
Every week the first two slides contain the contents and main objectives of the day.

Normally, the teaching term at EPFL has fourteen weeks.
However, Friday before the Easter break (Holy Friday) is a holiday.
Moreover, the class on Friday after 'Ascension' is dropped and replaced by somewhat longer sessions each week.

In total there will therefore be 12 lectures in this class. Each Friday we
- Start with exercises (exercise room)/ Discussion with TAs (main lecture hall)
- There will be two lectures of about 50 minutes each, first one starts at 10h45
- Break in between is 8 minutes (duration based on feedback from students)
- On average, one exercise is 'integrated' in the lecture (sometimes zero or two)

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Miniproject1: handout March 15;
submission April 29
Miniproject 2a/2b: handout April 15;
submission: choose May27 or June 3

Previous slide.

Results with artificial neural networks are discussed in newspaper articles and have inspired people around the world.
These years we experience the third wave of neural networks.
The first wave happened in the 1950s with the first simple computer models of neural networks, with McCulloch and Pitt and Rosenblatt's Perceptron. There was a lot of enthusiasm, and then it died.
The second wave happened in the 1980, around the Hopfield model, the BackPropagation algorithm, and the ideas of 'parallel distributed processing'. It died in the mid-nineties when statistical methods and Support Vector Machines took over.

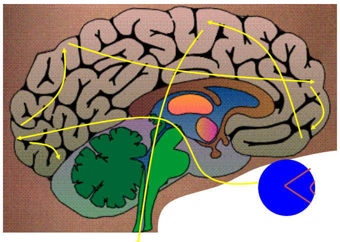
The third wave started around 2012 with larger neural networks trained on GPUs using data from big image data bases. These neural networks were able to beat the benchmarks of Computer vision and have been called 'deep networks'.

Artificial Neural Networks, how they work, and what they can do, will be in the focus of this lecture series.

The brain: Cortical Areas

motor
cortex

visual
cortex



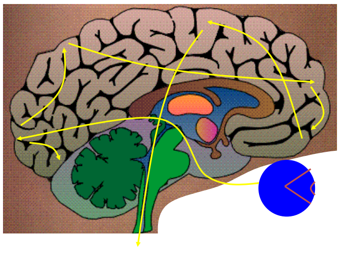
frontal cortex

to
muscles

Previous slide.
During all these waves, during 60 years of research, artificial neural networks researchers worked on building intelligent machines that learn, the way humans learn. And for that they took inspiration from the brain.

Suppose you look at an image. Information enters through the eye and then goes to the cortex.

The brain: Cortical Areas

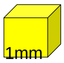


Previous slide.
Cortex is divided into different areas:
Information from the eye will first arrive at visual cortex (at the back of the head), and from there it goes on to other areas. Comparison of the input with memory is thought to happen in the frontal area (above the eyes). Movements of the arms a re controlled by motor cortex somewhere above your ears.

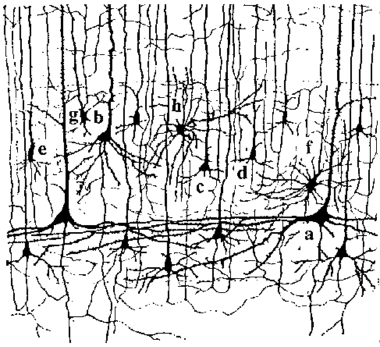
Talking about cortical areas provides a **macroscopic** view of the brain.

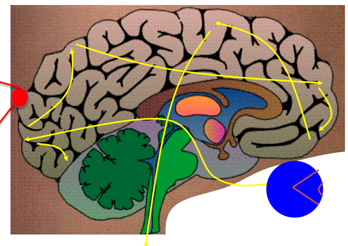
The Brain: zooming in

1mm



10 000 neurons
3 km of wire





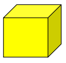
Ramon y Cajal

Previous slide.
If we zoom in and look at one cubic millimeter of cortical material under the microscope, we see a network of cells.
Each cell has long wire-like extensions.
If we counted all the cells in one cubic millimeter, we would get numbers in the range of ten thousand.
Researchers have estimated that, if you put all the wires you find in one cubic millimeter together you would find several kilometers of wire.

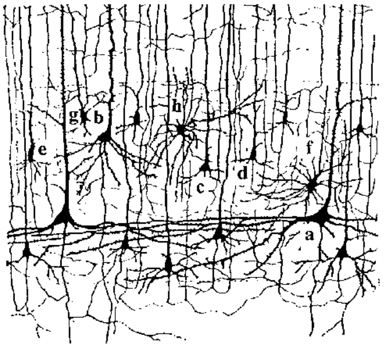
Thus, the neural network of the brain is a densely connected and densely packed network of cells.

The brain: a network of neurons

1mm

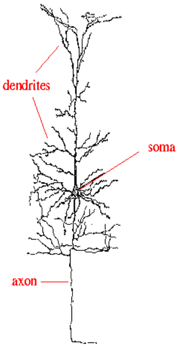


10 000 neurons
3km of wire



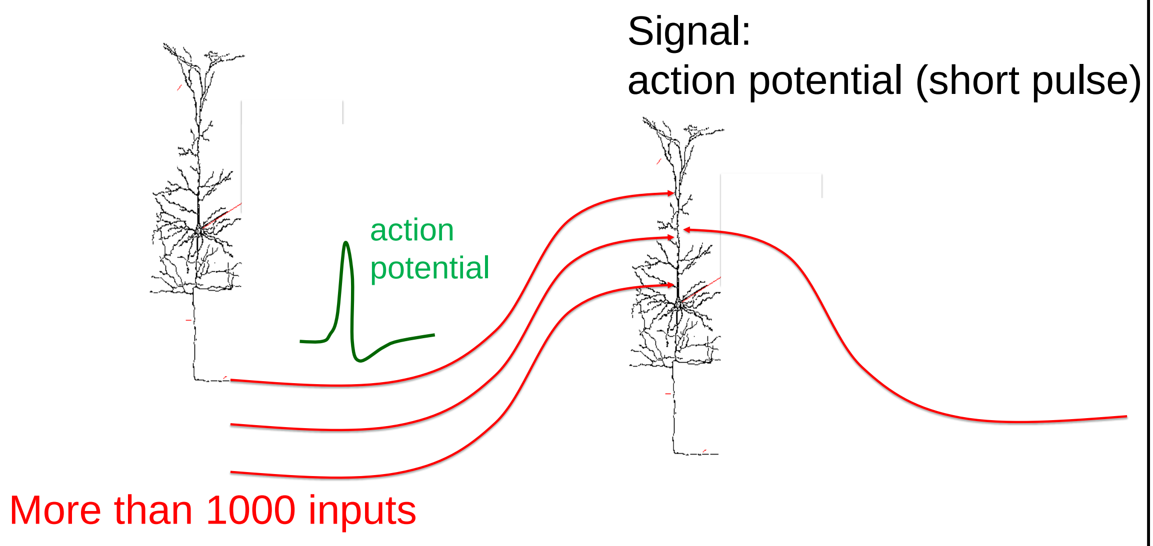
Ramon y Cajal

Signal:
Action potential (short pulse)



Previous slide.
These cells are called neurons and communicated by short electrical pulses, called action potentials, or ‘spikes’.

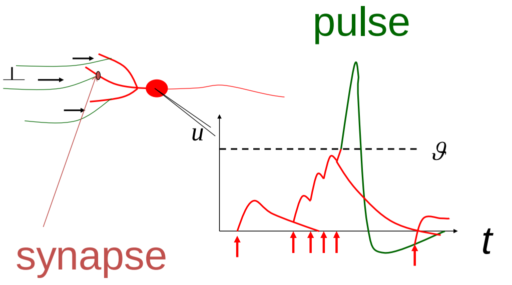
The brain: signal transmission



Previous slide.
Signals are transmitted along the wires (axons). These wires branch out to make contacts with many other neurons.

Each neuron in cortex receives several thousands of wires from other neurons that end in ‘synapses’ (contact points) on the dendritic tree.

The brain: neurons sum their inputs

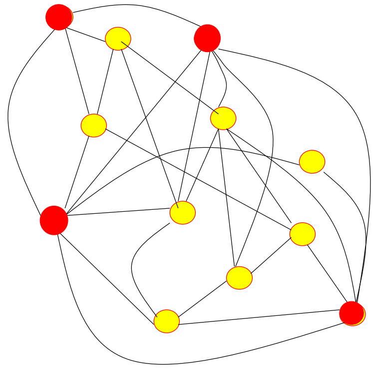


Previous slide.
If a spike arrives at one of the synapses, it causes a measurable response in the receiving neuron.

If several spikes arrive shortly after each other onto the same receiving neuron, the responses add up.

If the summed response reaches a threshold value, this neuron in turn sends out a spike to yet other neurons (and sometimes back to the neurons from which it received a spike).

Summary: the brain is a large network of neurons



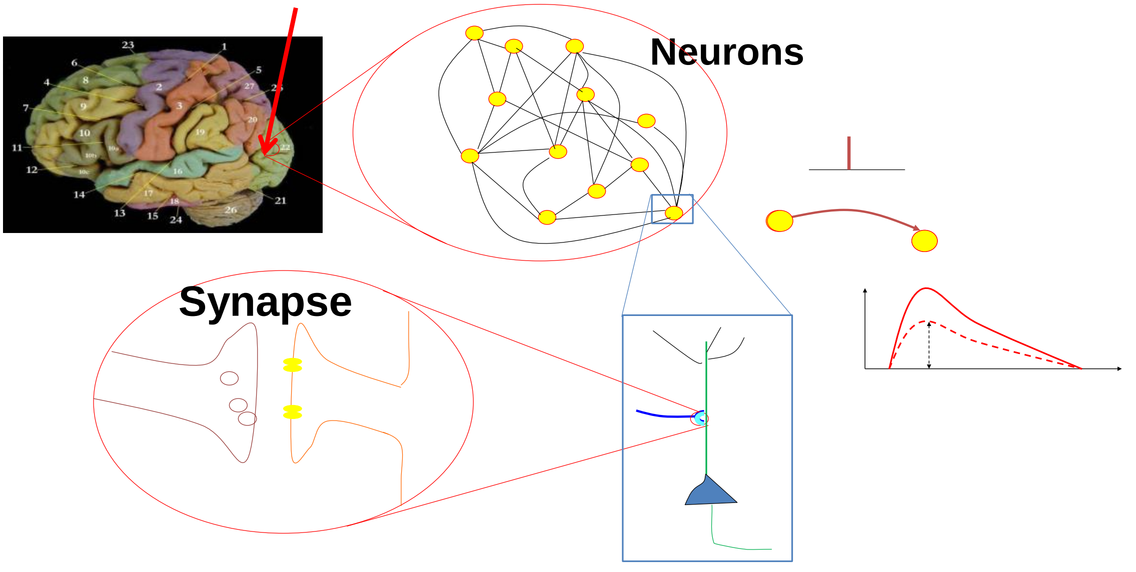
- Active neuron

Previous slide.

Thus, signals travel along the connections in a densely connected network of neurons.

Sometimes I draw an active neuron (that is a neuron that currently sends out a spike) with a filled red circle, and an inactive one with a filled yellow circle.

Learning in the brain: changes between connections



learning = change of connection

Previous slide.

Synapses are not jut simple contact points between neurons, but they are crucial for learning.

Any change in the behavior of an animal (or a human, or an artificial neural network) is thought to be linked to a change in one or several synapses.

Synapses have a ‘weight’. Spike arrival at a synapse with a large weight causes a strong response; while the same spike arriving at a synapses with a small weight would cause a low-amplitude response.

All Learning corresponds to a change of synaptic weights. For example, forming new memories corresponds to a change of weights. Learning new skills such as table tennis corresponds to a change of weights.

Artificial Neural Networks

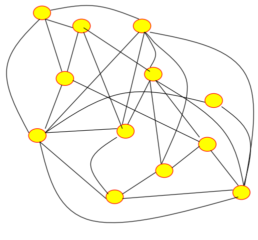
Wulfram Gerstner
EPFL, Lausanne, Switzerland

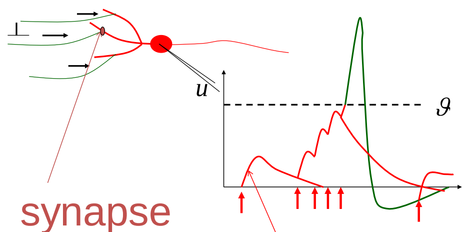
1. The brain
2. **Artificial Neural Networks**

Previous slide.

After this super-short overview of the brain, we now turn to artificial neural networks: highly simplified models of neurons and synapses.

Modeling: artificial neurons





-responses are added
-pulses created at threshold
-transmitted to other

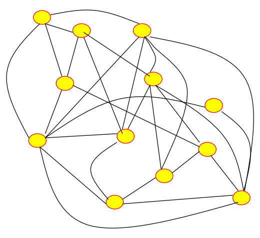
→ Mathematical description

Previous slide.
In the previous part we have seen that response are added and compared with a threshold.

This is the essential ideal that we keep for the abstract mathematical model in the following.

We drop the notion of pulses or spikes and just talk of neurons as active or inactive.

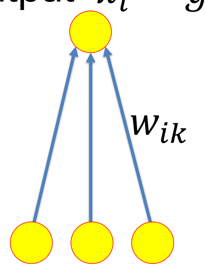
Modeling: artificial neurons



forget spikes: continuous activity x
forget time: discrete updates

nonlinearity/threshold

activity of output $x_i = g\left(\sum_k w_{ik}\, x_k\right)$ ϑ



weights = adaptive parameters

activity of inputs x_k

Previous slide.

The activity of inputs (or input neurons) is denoted by x_k

The weight of a synapse is denoted by w_{ik}

The nonlinearity (or threshold function) is denoted by g

The output of the receiving neuron is given by

$$x_i = g\left(\sum_k w_{ik}\, x_k\right)$$

Learning of connections in biology

Where do the connection weights come from?

Hebbian Learning

When an axon of cell **j** repeatedly or persistently takes part in firing cell **i**, then j's efficiency as one of the cells firing i is increased

Hebb, 1949

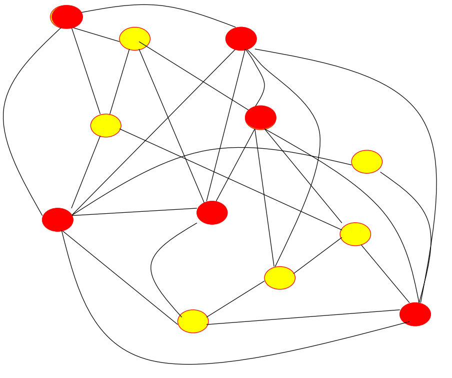
- local rule
- simultaneously active neurons

Previous slide.

As mentioned earlier, weights can by weak or strong – but which 'rule' sets the weights?

In biology a basic idea is that joint activity of two neurons can change the weight that connects those two neurons (Hebb rule).

Hebbian Learning of Associations



Previous slide.

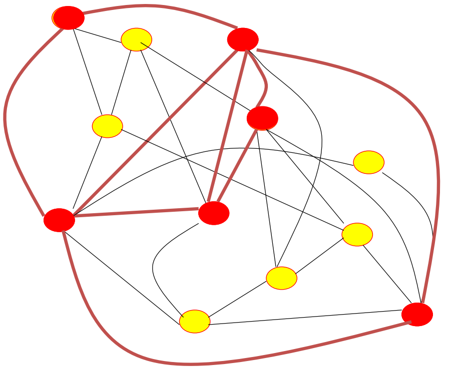
Why is the Hebb rule useful?

Suppose you see (for the first time in your life!) an apple.

Some neurons will be activated because the apple is red, others because it has a round shape, or because it has a certain odor.

If the brain implements the Hebb rule, the result of this co-activation of different neurons is that the connections between the active neurons are strengthened.

Hebbian Learning of Associations

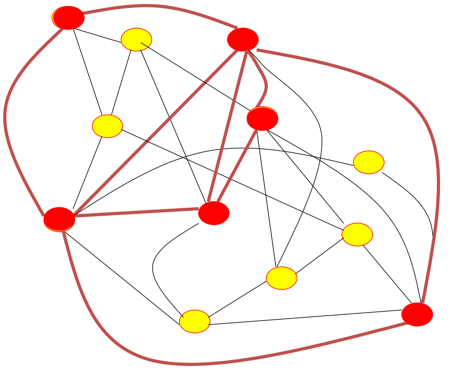


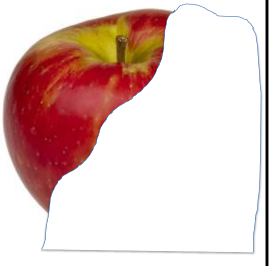
item memorized by change of synaptic weights

Previous slide.
We claim that this change of weights is important to 'memorize' the item 'apple'.

Hebbian Learning: Associative Recall

Recall: Partial info





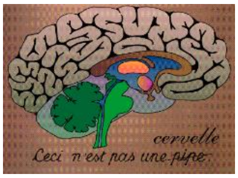
item recalled

Previous slide.
To check this claim, we now show a partial picture of an apple. Only part of the visual information is available, and no odor information.

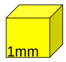
But because the stimulus is sufficient to reactivate some of the same neurons as during the first exposure to the apple, the strong connections now enable the neurons that encode the missing information to also become active.

The item is recalled and the missing information is associated with the partical stimulus: Ah, I remember how this apple smelled.

Neurons and Synapses form a big network



Brain



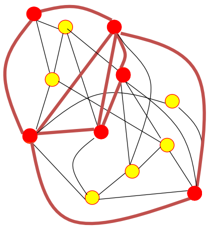
1mm

10 000 neurons
3 km of wire

10 billions neurons

10 000 connexions/neurons

memory in the connections



Distributed Architecture

No separation of processing and memory

Previous slide.
Even though we are not going to work with the Hebb rule during this class, the above example still shows that

- Memory is located in the connections
- Memory is largely distributed
- Memory is not separated from processing

(as opposed to classical computing architectures such as the van Neumann architecture or the Turing machine)

Quiz: biological neural networks

[ ] Neurons in the brain have a threshold.
[ ] Learning means a change in the threshold.
[ ] Learning means a change of the connection weights
[ ] The total input to a neuron is the weighted sum of individual inputs
[ ] The neuronal network in the brain is feedforward: it has no recurrent connections

Your notes.

Artificial Neural Networks

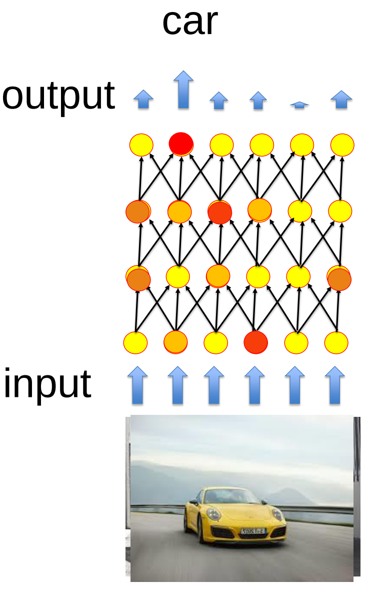
Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The brain
2. **Artificial Neural Networks**
   - artificial neurons
   - artificial neural networks for classification

Previous slide.
Now that we know about artificial neurons and synaptic weights, let us construct a useful network.
The first task we study is classification

Artificial Neural Networks for classification

feedforward network



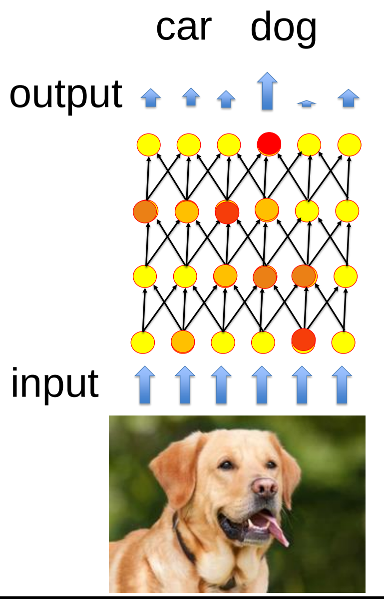
Previous slide.
An input is presented at the bottom of the network.
It passes through several layers of neurons.
All connections are directed from the bottom to the next layer further up: this architecture is called a feedforward network.

The output is a set of neurons that correspond to different 'classes'.

An ideal network should respond with activating the neuron corresponding to 'car', if the input image shows a car.

Artificial Neural Networks for classification

Aim of learning:
Adjust connections such that output class is correct (for each input)



Previous slide.
The aim of learning is to adjust the connection weights such that, for each input, the output class is correct.
If the input is a dog, the 'dog'-neuron should respond.
If the input is a car, the 'car'-neuron should respond.

In the first half of the semester, we focus on the task of building and training artificial neural networks for classification.

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

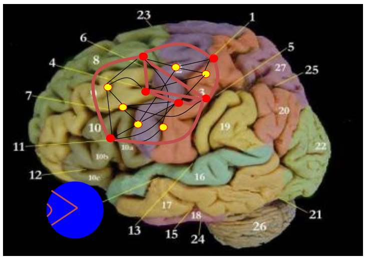
1. The brain
2. Artificial Neural Networks
   - artificial neurons
   - Neural networks for classification
   - **Neural networks for action learning**

Previous slide.
However, classification is not the only task we are interested in.

Artificial Neural Networks for action learning

Coactivation of 2 neurons:
- strengthens connection
- Facilitates to repeat same action

Even the mistakes?



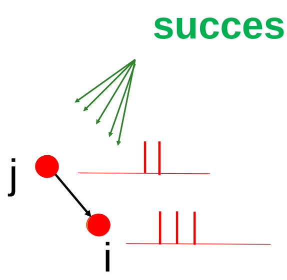
Missing:
Value of action
- ‘goodie’ for dog
- ‘success’
- ‘compliment’

Previous slide.
Let us go back for a moment to the brain, and how humans or animals learn.

We learn actions by trial and error exploiting rather general feedback: reward or praise on one side, pleasure and pain on the other side.

In other words, the mere co-activation of two neurons (Hebb rule) is not enough to explain human learning. Important is the notion of value of an action.
Learning actions or sequences of actions is very different from classification.

Modeling – the role of reward



Three factors for changing a connection
- activity of neuron j
- activity of neurone i
- success

Barto 1985, Schultz et al. 1997; Waelti et al., 2001; Reynolds and Wickens 2002; Lisman et al. 2011

Reinforcement learning = learning based on reward

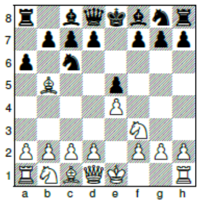
Previous slide.
Reward-based learning has been studied in psychology and biology.

At the level of synapses, reward-based learning can be seen as a generalization of the Hebb rule:
To implement a change of synaptic weights, three factors are needed:
- the activity of the sending neuron;
- the activity of the receiving neuron;
- and a broadcast signal that transmit the information: this was successful (because it led to a reward).

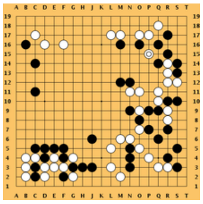
Learning based on rewards is at the center of reinforcement learning.

Deep reinforcement learning

Chess



Go



Artificial neural network (*AlphaZero*) discovers different strategies by playing against itself.

In Go, it beats Lee Sedol

Previous slide.
The same kind of ideas have also been implemented in artificial neural networks that are trained by reinforcement learning.

In a game such a Chess or Go, the reward signal is only given once at the very end of the game: positive reward if the game is won, and negative reward if it is lost.

This rather sparse reward information is sufficient to train an artificial neural network to a level where it can win against grand masters in chess or Go.

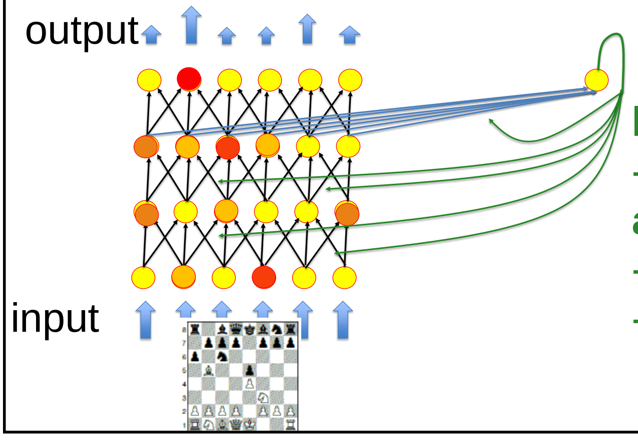
To improve performance, each network plays against a copy of itself. By doing so it discovers good strategies (such as openings in chess).

Deep reinforcement learning

Network for choosing action

action:
Advance king

2^e output for **value** of action:
probability to win



learning:
- change connections

aim:
- Predict value of position
- Choose next action to win

Previous slide.
Schematically, the artificial neural network takes the position of chess as input.
There are two types of outputs:
- The main outputs are the actions such as ‘move king to the right’
- An auxiliary output predicts the ‘value’ of each state. It can be used to explore possible next positions so as to pick the one with the highest value.
- The value can be interpreted as the probability to win (given the position)

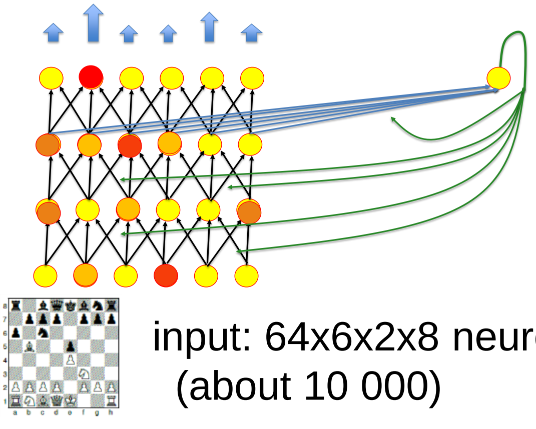
In the theory of reinforcement learning, positions are also called ‘states’.

Deep reinforcement learning (alpha zero)

Silver et al. (2017) , Deep Mind

output: 4672 actions

advance king



input: 64x6x2x8 neuronss
(about 10 000)

Training 44Mio
games (9 hours)

Planning:
potential sequences
(during 1s before playing next action)

Previous slide.
Since there are many different positions, the number of input neurons is in the range of ten thousand:

On each of 64 positions there can be one of 6 different 'figures' (king, horse) of 2 different colors.
To avoid repetitions the 8 last time steps are used as input.

Training is done by playing against itself in 44 million games.

The allotted computer time for planning the next action is 1s.

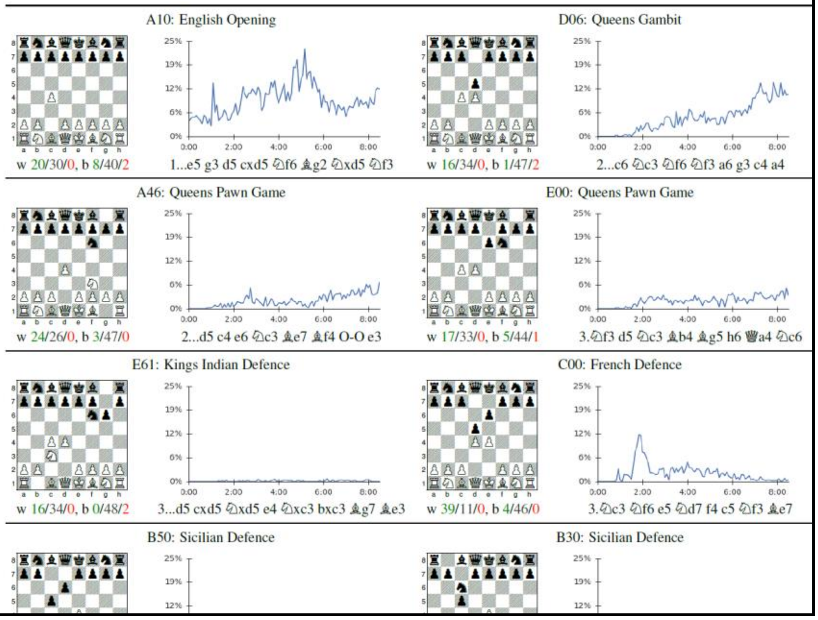
Deep reinforcement learning (alpha zero)

Silver et al. (2017)

Chess:
-discovers classic openings

-beats best human players

-bets best classic AI algorithms



Previous slide.
After training for 44 Milllion self-play games, the algorithm matches or beats classical AI algorithms for chess.

Interestingly, it ‘discovers’ well-known strategies for openings, corresponding closely to well known openings in textbooks on chess.

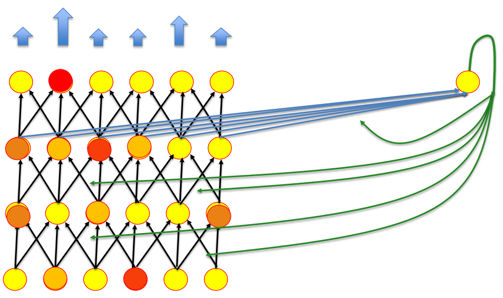
When trained on go it beats the world champions.

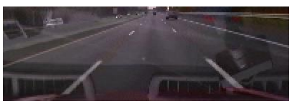
Self-driving cars

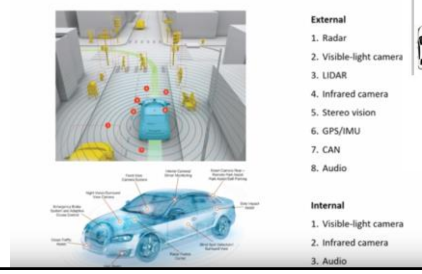
https://selfdrivingcars.mit.edu/ *Lex Friedman, MIT*

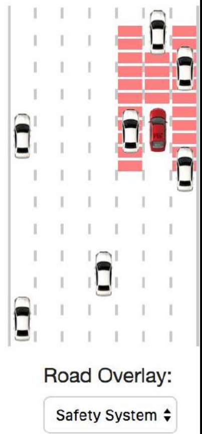
advance and accerate

Value: security, duration of travel









Previous slide.
Similar reinforcement learning algorithms are also used to train selfdriving cars.
There is a nice series of video lectures by Lex Friedman on the WEB.
Inputs are video images as well as distance sensors.

The value is security (top priority) combined with duration of travel.

Artificial Neural Networks

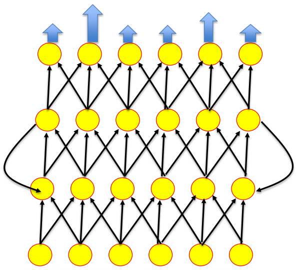
Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The brain
2. Artificial Neural Networks
   - for classification
   - for action learning
   - **for sequences (music, translation, speech)**

Previous slide.
The third task we will consider in this class is sequence learning.

Deep networks with recurrent connections

'a man sitting on a couch with a dog'



Network desribes the image with the words:

'a man sitting on a couch with a dog'

(Fang et al. 2015)

Previous slide.
An amazing example is this network which looks at static image and outputs the spoken sentence:
'A man is sitting on a couch with a dog'.

Sequence learning requires recurrent connections (feedback connections), in contrast to the feedforward architecture that we have seen so far.

Quiz: Classification versus Reinforcement Learning

[ ] Classification aims at predicting the correct category such as ‘car’ or ‘dog’
[ ] Classification is based on rewards
[ ] Reinforcement learning is based on rewards
[ ] Reinforcement learning aims at optimal action choices

Your notes.

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The brain
2. Artificial Neural Networks
   - for classification
   - for action learning
   - for sequences
3. **Overview of class**

Previous slide/next slide
All three tasks:
-classification (5 lessons)
-sequence learning (1 lesson)
-reinforcement learning (5 lessons)
will be covered.
Plus an extra session for convolutional networks.

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. Simple perceptrons for classification
2. Backprop and multilayer perceptron
3. Statistical Classification by deep networks
4. Deep learning: regularization and tricks of the trade miniproject1
5. Error landscape and optimization methods for deep networks
6. Convolutional networks
7. Sequence predictions and Recurrent networks ← miniproject2a
8. Reinforcement learning1: Bellman and SARSA ← miniproject2b
9. Reinforcement learning2: variants of SARSA
10. Reinforcement learning3: Policy Gradient
11. Deep Reinforcement learning
12. Reinforcement learning and the Brain

Previous slide.
Overall there will be 12 sessions for 14 weeks.

Friday after ascension is no class.
Instead each week there will be 97 minutes of lectures.

Every student has to do two miniprojects.

Miniprojects (MPs): we use software package 'Keras'
- hand in 2 (not 3) out of 3 projects
- graded on a scale of 1-6
- average grade of MPs counts 30% toward final grade
- we do **fraud detection interviews**
- you get **4 weeks** for each MP
- MP done in groups of two students (not alone)
- interview for final MP is in first week after end of classes

→ **plan ahead!!**

Written exam:
- counts 70 percent toward final grade
- no tools allowed (no calculator, no cell phone, no paper, no book)
- 'mathy', similar to exercises

Previous slide.
Everybody does the first miniproject.
For the second miniproject you choose between 2a (sequences) or 2b (reinforcement learning).

The average grade of the 2 miniprojects counts 30 percent toward the final grade.

Exercise sessions as follows:

- hand-out of exercise sheet ***n*** Friday of week ***n***
- You work on it at home on your own
- Solutions posted at noon, Monday, week ***n+1***
- Friday week ***n+1*** by 10:15 am: the most difficult exercise is explained by a TA on the blackboard
- Friday week ***n+1*** at 10:15am. Main lecture hall for discussions with Tas.
- Friday week **n+1 at 10:45 am class/lecture**
- Friday week **n+1** at 12:40 individual Q&A to miniprojects

Previous slide.

Exercise sessions follow a special model, very different from the standard EPFL way of doing it.

Weakly sessions as follows:

- Meet at 10:15 in lecture hall
- Exercise discussed from 10:15 -10:40
- break 5 min
- Lecture 1 from 10:45-11:35
- Break 8 min
- Lecture 2 from 11:43-12:35
- Break 5 min
- Discussion of miniproject from 12:40-12:55

Each lecture typically includes one ‘in-class’ exercise

Previous slide.

For the in-class exercises it is important that you really try to solve them. No problem if you fail (some are harder than others). But important is that you start to think about how you would solve them.

The results of in-class exercises are needed so as to understand the rest of the lecture.

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

- The math is developed on the blackboard
- There are no written course notes!!
- All of the contents are **standard textbook material**

Choose a textbook that you like! I recommend

For **first half** of class:

- *Pattern Recognition and Machine Learning*, C.M Bishop, 2006
- *Neural Networks for Pattern Recognition*, C.M. Bishop, 1995
- *Deep Learning*, Ian Goodfellow et al., 2017 (also online)

For **second half** of class:

- *Reinforcement learning,* R. Sutton+ A. Barto (2nd ed, online)

Also good: *Neural networks and learning machines*, S. Haykin

Previous slide.

Work with a textbook that you like.

The books of
Goodfellow et al.
Sutton and Barto
are the basis of the class. Both are available online in pdf format as preprints for free.

Artificial Neural Networks

Prerequisits:
CS433, Machine Learning
(Profs Jaggi+Urbanke)

Rules:
If you have taken this class: please ask many questions

If you have not taken this class: please do not complain

Previous slide.

The overlap with the class of Jaggi+Urbanke is minimal (main overlap for 'regularization'). But we need quite a few of their results as a basis!

Some students have taken a very similar class and then this is also fine.

Students who did not take the above class (or something very similar) are not admitted to the class 'Artificial Neural Networks'. If they attend, it is at their own risk; they should not ask questions, but fill the knowledge gaps on their own. They should not complain if they find the class too hard.

Artificial Neural Networks

Learning outcomes:

- apply learning in deep networks to real data
- assess/evaluate performance of learning algorithms
- Elaborate relations between different mathematical concepts of learning
- judge limitations of learning algorithms
- propose models for learning in deep networks

Transversal skills:

Access and evaluate appropriate sources of information

Manage priorities

work through difficulties, write a technical report

Previous slide.

Access and evaluate appropriate sources of information
→ this means: you should learn to read textbooks. It is not sufficient to just look at slides.

Manage priorities
→ this means: the two miniprojects together only count 30 percent. Don't write a program with bells and whistles, but really focus on the things you are asked to do.

work through difficulties,
→ this means: some things will look hard at the beginning, be it in the miniproject or in the mathematical calculations. That's normal, but you have to work through this.

write a technical report
→ this means: we would like to receive a readable technical report for the miniprojects. Concise, to the point, not too long.

Artificial Neural Networks

Work load:

4 credit course → 6 hours per week for 18 weeks

(1 ECTS = 27 hours of work)

Previous slide.

18 weeks for 12 weeks of lectures:
The week of ascension, and easter, and exam preparation time counts as well.

Two ways to study for this class

A: Self-paced self-study
1. Read slides 1+2
   (objectives and reading)
2. Start exercise n.
3. If stuck, read book chapter
   Return to 2.
4. n←n+1
5. Compare with solutions
6. Do quizzes in slides
   (yellow pages)

Hand-in two miniprojects.

Note: Slides are not meant for self-study. Use textbook!

B: Lecture-based weekly
1. Follow lecture
   - annotate slides
   - participate in quizzes
   - try to solve in-class exercises
2. At home do other exercises
3. Compare with solutions.
4. If stuck, vote for this exercise to be explained.

Hand-in two miniprojects.

Note: Do not forget to annotate slides so that you can use them.

Previous slide.

You don't need to come to class, since all material is textbook material. But then you really have to study the textbooks!
Slides are not meant to replace textbooks.

Questions?

… before we start

TA's this year:

- Berfin Simsek
- Nicolas El Maalouly
- Chiara Gastaldi
- Florian Colombo
- Bernd Illing

Previous slide.

Your Semester planning

Wulfram Gerstner
EPFL, Lausanne, Switzerland

The course **'Deep Learning'** (Fleuret)
and the course **'Artificial Neural Networks'** (Gerstner)
have about 20-30 percent overlap.
You can take either one or the other or both (OR), students consider the course of Prof. Fleuret as 'more practical coding-oriented' than this one here.

The course **'Unsupervised and Reinforcement L.'** (not given)
and the course **'Artificial Neural Networks'** (Gerstner)
have about 20-30 percent overlap (three weeks)
I suggest to take either one or the other or both;
The other course is more 'biological' than this one here

Previous slide.
The class 'Deep learning' also treats backpropagation, tricks of the trade, convolutional networks. It does not contain any reinforcement learning.

The class 'Unsupervised and Reinforcement learning' also treats Reinforcement learning. It does not contain any supervised learning for classification, no backpropagation. Reinforcement learning is discussed with a biological focus. The class is not given in 2019.

The class 'Artificial Neural Networks' is planned for IC students who have already taken the class 'Machine Learning' by Jaggi-Urbanke.

The class 'Deep Learning' is planned for STI students and does not have any prerequisits (except engineering bachelor)

Your semester planning

Wulfram Gerstner
EPFL, Lausanne, Switzerland

The course **'Deep Learning'** (Fleuret)
and the course **'Unsupervised and Reinforcemnt L.'** (Gewaltig)
have less than 5 percent overlap.
You can take one or the other or both (OR).

+The course **'Unsupervised and Reinforcement L.'** (Gewaltig)
is oriented towards biological questions, aimed at SV students
+The course **'Deep Learning'** (Fleuret) is an applied course.
It has no prerequisits and does not cover reinforcement learning.
Aimed at STI students
+ The course 'Artificial Neural Networks' (Gerstner) is a course
aimed at IC students. **Prerequisit: Machine Learning** (Jaggi)

Previous slide.
The course starting now is aimed for IC students.

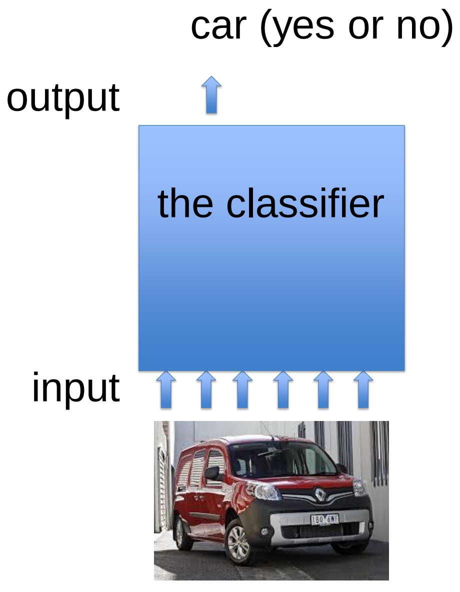
Artificial Neural Networks: Lecture 1
Simple Perceptrons for Classification

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:
- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons

Previous slide.
… and now we really start.

1. The problem of Classification



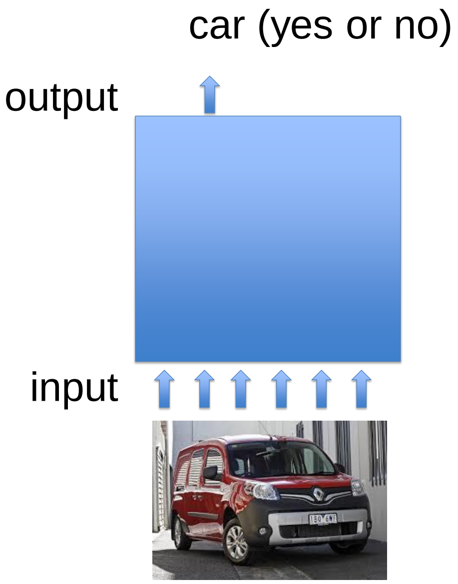
Previous slide.
We focus on the class of classification.

To be concrete we consider images. The task of the classifier is to say: yes or no.

In the concrete example: yes means, there is a car on the image

1. The problem of Classification

Blackboard 1:
from images to vector



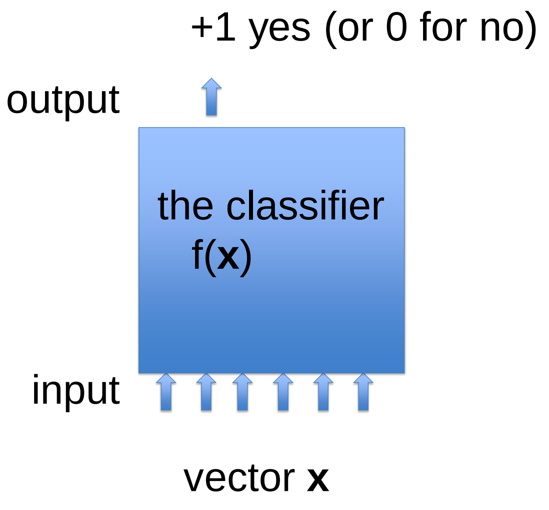
Previous slide.

Even though we visualize the input as a two-dimensional image, the input to the networks really just is a vector **x** of pixel values.

Blackboard 1:
from images to vector

Your notes.

1. The problem of Classification



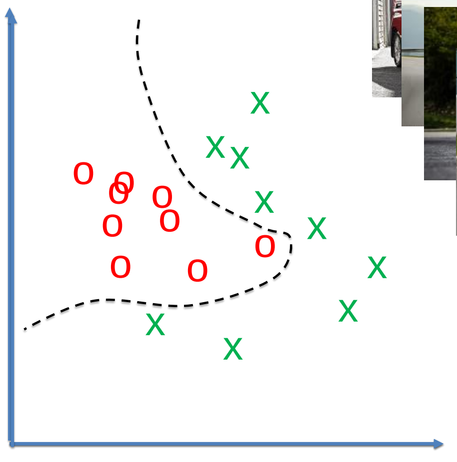
Previous slide.

The input is a vector **x**

The classifier is a function f(**x)**
that maps the input to the output

The output is binary: +1 or 0.

1. Classification as a geometric problem



Blackboard 2:
from vectors to classification

Previous slide.
Classification means: assigning a +1 to some inputs (e.g. cars) and 0 to other inputs (not cars).

Classification corresponds to a separating (by some surface) the positive examples (green crosses) from the negative ones (red circles).

The space is the space of input vectors **x**

Blackboard 2:
from vectors to classification

Your notes.

1. Classification as a geometric problem

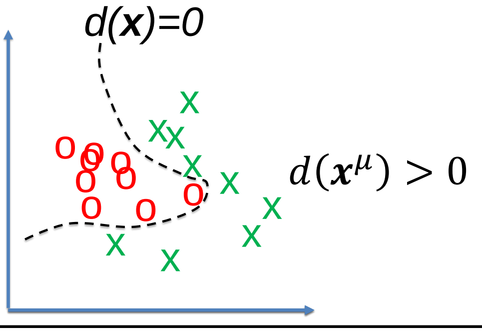
Task of Classification

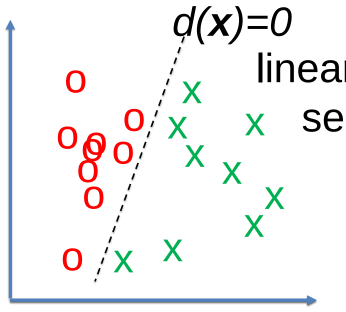
= find a **separating surface** in the high-dimensional input space

Classification by **discriminant function** *d(**x**)*

→ *d(**x**)=0* on this surface*; d(**x**)>0 for all positive examples **x***

*d(**x**)<0 for all counter examples **x***





linearly separable problem

Previous slide.

The discriminant function *d(**x**)* takes inputs ***x*** and maps these to:

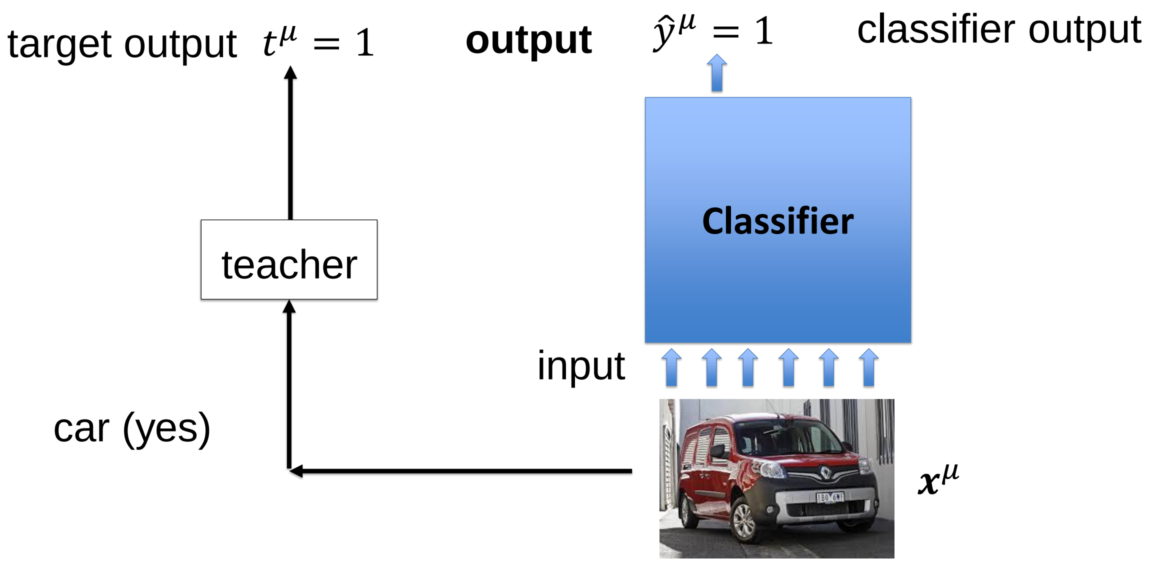
*d(**x**)>0 for all positive examples **x***

*d(**x**)<0 for all counter examples **x***

*d(**x**)=0 on the separating surface*

Solving a classification problem therefore is equivalent to finding a discriminant function.

2. Data base for Supervised learning



Previous slide.
To construct such a discriminant function we need a data base for supervised learning.
The problem is called supervise learning because we assume that a teacher has previously looked at the examples and assigned labels.

A label $t^{\mu} = 1$ for an input vector $\boldsymbol{x}^{\mu}$ means that this input pattern belongs to the class (positive example)

A label $t^{\mu} = 0$ for an input vector $\boldsymbol{x}^{\mu}$ means that this input pattern does not belong to the class (counter-example)

2. Data base for Supervised learning

P data points $\{ \ (\boldsymbol{x}^{\mu}, t^{\mu}) \ , \quad 1 \leq \mu \leq P \ \};$

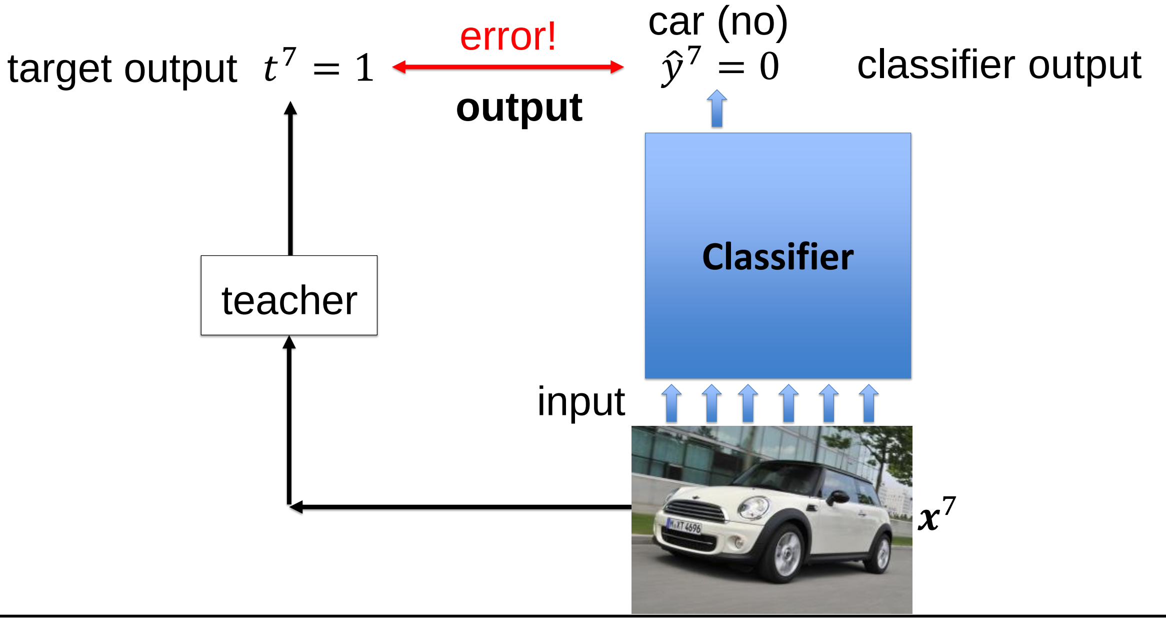
input target output

$t^{\mu} = 1$ car =yes

$t^{\mu} = 0$ car =no

Previous slide.
The data base for supervised learning contains P data points, each consisting of a pair of input and target output.

2. Data base for Supervised learning



Previous slide and next slide.

The basic idea of supervised learning is that the actual output of the classifier is compared with the target output. If there is a mismatch, then the error can be used to optimize the function f(x) of the classifier.

2. Data base for Supervised learning

P data points $\{ \ (\boldsymbol{x}^{\mu}, t^{\mu}) \ , \quad 1 \leq \mu \leq P \ \};$

input target output

for each data point $\boldsymbol{x}^{\mu}$, the classifier gives an output y^{μ}

→ use errors $\hat{y}^{\mu} \neq t^{\mu}$ **for optimization of classifier**

Remark: for multi-class problems $\mathbf{y}$ and $\boldsymbol{t}$ are vectors

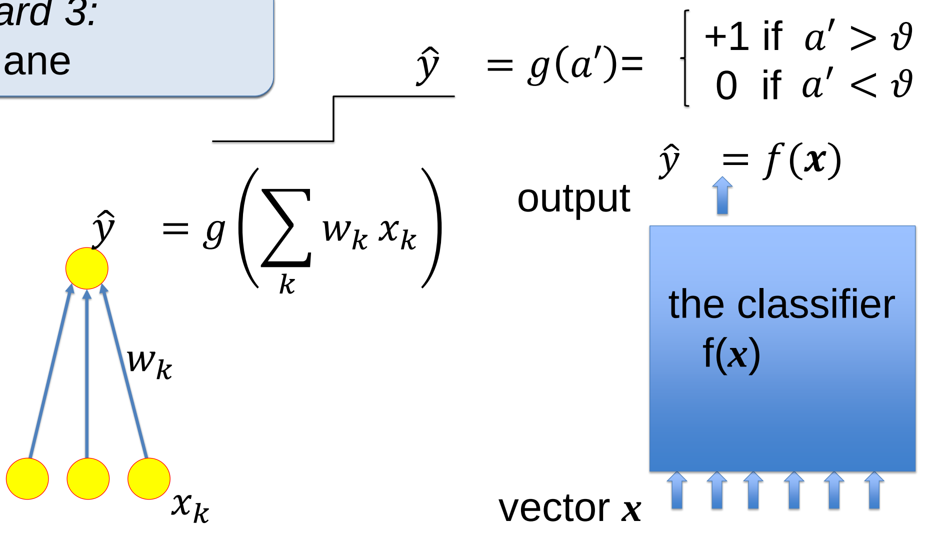
Previous slide.

A single-class classifier has a single binary target output $t^{\mu} = 0 \ or \ 1$.

For a multi-class classifier the target output is a vector.

3. Single-Layer networks: simple perceptron

Blackboard 3:
hyperplane



Previous slide.
So far we have not specified the function f(x) of the classifier.

Now we assume that the classifier consists of a single artificial model neuron.

Each component x_k of the input vector is multiplied by a weight w_k.

The function g( ) is some nonlinear function.

Blackboard 3:
hyperplane

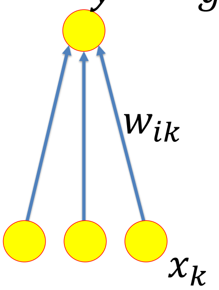
Your notes.

3. Single-Layer networks: simple perceptron

$$\hat{y}^{\mu} = 0.5[1 + sgn(\underbrace{\textstyle\sum_k w_k\, x_k - \vartheta}_{a})]$$

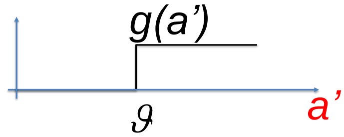
output

$$\hat{y}^{\mu} = g\left(\underbrace{\sum_k w_k\, x_k}_{a'}\right)$$



input

vector $\boldsymbol{x}$



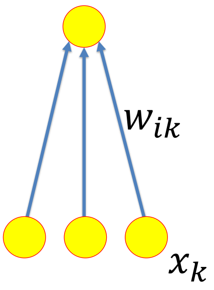
$$g(a') = \begin{cases} 1 & \text{if } a' > \vartheta \\ 0.5 & \text{if } a' = \vartheta \\ 0 & \text{if } a' > \vartheta \end{cases}$$

Previous slide.
Top line: Often we choose for g a step function with threshold ϑ .

The total effective input activation of the neuron is called a (including the threshold) or a' (before the threshold is subtracted).

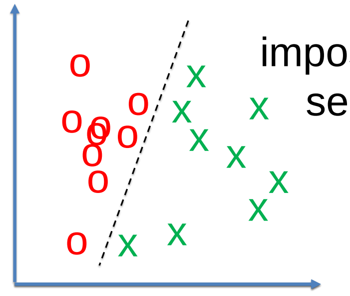
3. Single-Layer networks: simple perceptron

$$\hat{y} \quad = 0.5[1 + sgn(\sum_k w_k\, x_k - \vartheta)]$$



vector $\boldsymbol{x}$

$$d(\boldsymbol{x}) = \sum_k w_k\, x_k - \vartheta = 0$$



imposes a linear separation

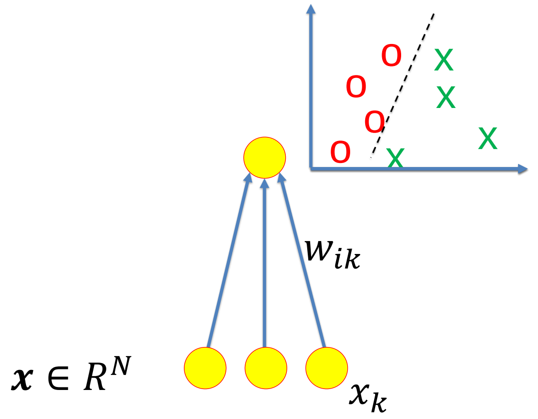
Previous slide.

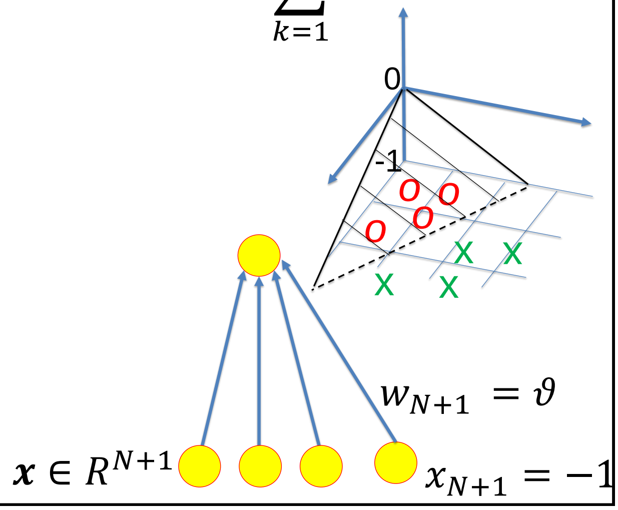
A single artificial model neuron implements a linear separation of the positive and negative examples. Thus the discriminant function is a hyperplane.

3. remove threshold: add a constant input

$$d(\boldsymbol{x}) = \sum_{k=1}^{N} w_k\, x_k - \vartheta\ = 0$$

$$d(\boldsymbol{x}) = \sum_{k=1}^{N+1} w_k\, x_k\ = 0$$





Previous slide.

The hyperplane has a distance ϑ /|w| from the origin.

Formally, we can represent the threshold by an additional weight $w_{N+1}\ = \vartheta$ which is multiplied with a constant input $x_{N+1}\ = -1$.

In this (N+1)-dimensional space, the hyperplane passes through the origin.

3. Single-Layer networks: simple perceptron

a simple perceptron

- can only solve linearly separable problems
- imposes a separating hyperplane
- for $\vartheta = 0$ hyperplane goes through origin
- threshold parameter ϑ can be removed by adding an input dimension

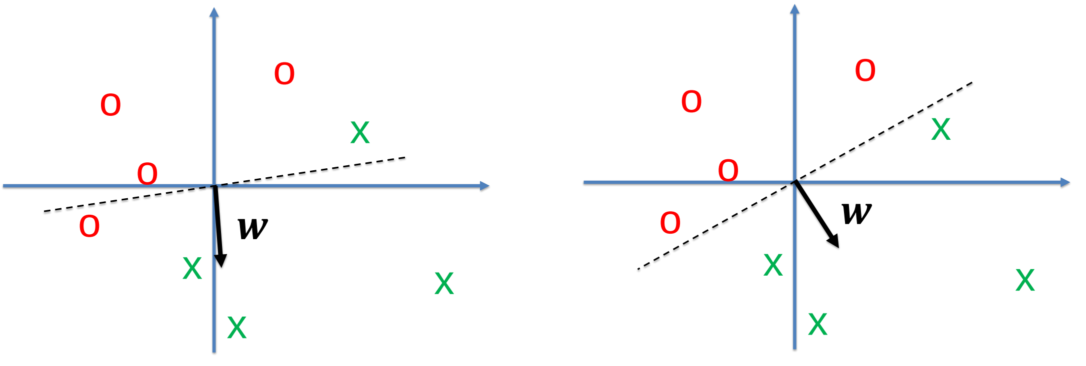
→ in ***N+1*** dimensions hyperplane always goes through origin

- We can **adapt the weight vector** to the problem

Previous slide.
Thus, a simple perceptron can only solve linearly separable problems. Important for the following is that the positioning of the hyperplane in the high-dimensional space can be changed by adapting the weight vector to the data base.

4. Perceptron algorithm: turn weight vector (in N+1 dim.)

$$hyperplane:\ d(\boldsymbol{x}) = \sum_{k=1}^{N+1} w_k\, x_k = \boldsymbol{w}^T \boldsymbol{x} \ = 0$$



Previous slide.
In the following we always work in N+1 dimensions and exploit that the hyperplane goes through the origin.

Left: one of the examples is misclassified.

Right: all examples are correctly classified.

Idea: Turn weight vector in appropriate direction to go from the situation on the left to the situation on the right.

4. Perceptron algorithm: turn weight vector

Blackboard 4:
geometry of perceptron algo $\Delta \boldsymbol{w} \sim \boldsymbol{x}^{\mu}$

Perceptron algo (in N+1 dimensions):

- set $\mu = 1$

(1) cycle many times through patterns

- choose pattern μ
- calculate output

$$\hat{y}^{\mu} = 0.5[1 + sgn(\boldsymbol{w}^T \boldsymbol{x}^{\mu})]$$

- update by

$$\Delta \boldsymbol{w} = \gamma [t^{\mu} - \hat{y}^{\mu}] \boldsymbol{x}^{\mu}$$

- iterate $\mu \leftarrow (\mu + 1) mod \boldsymbol{P}$, back to (1)

(2) stop if no changes for all ***P*** patterns

Previous slide.
A change of the weight vector (during the update step) happens only if the actual output $\hat{y}^{\mu}$ for pattern $\boldsymbol{x}^{\mu}$ is not equal to the target output t^{μ}

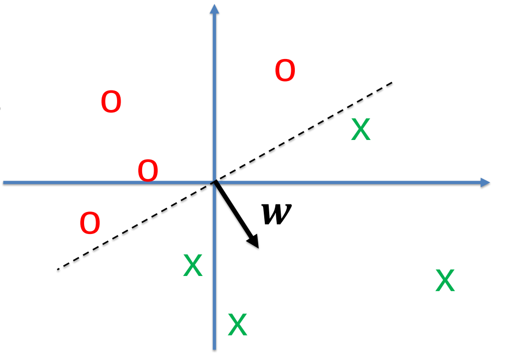
Blackboard 4:
geometry of the perc. algo

Your notes.

4. Perceptron algorithm: theoreom

If the problem is linearly separable, the perceptron algorithm converges in a finite number of steps.

Proof: in many books, e.g.,
Bishop, 1995,
Neural Networks for Pattern Recognition



Previous slide.

Important: Convergence is only guaranteed if the problem is linearly separable.

Quiz: Perceptron algorithm

The **input vector has *N* dimensions** and we apply a perceptron algorithm.

[ ] A change of parameters corresponds always to a rotation of the separating hyperplane in N dimensions.

[ ] A change of the separating hyperplane implies a rotation of the hyperplane in N+1 dimensions.

[ ] An increase of the length of the weight vector implies an increase of the distance of the hyperplane from the origin in N dimensions.

[ ] An increase of the length of the weight vector implies that the hyperplane does not change in N dimensions

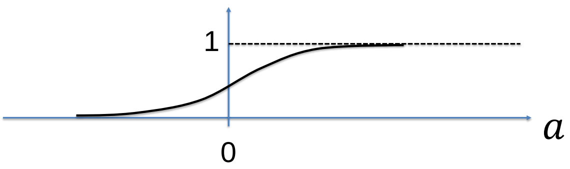
[ ] An increase of the length of the weight vector implies that the hyperplane does not change in N+1 dimensions

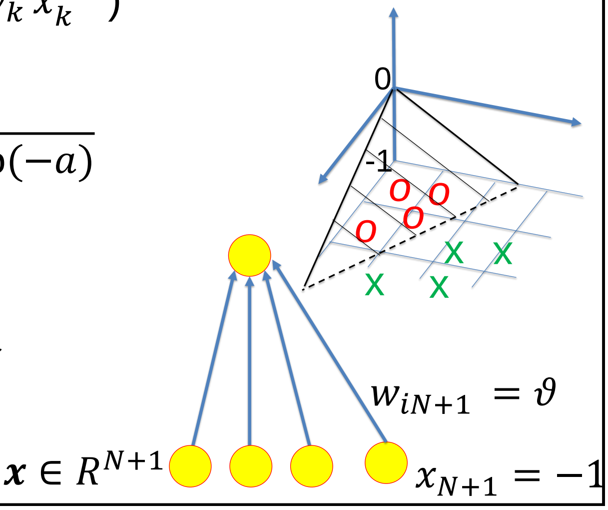
Your notes.

5. Sigmoidal output unit

$$\hat{y}^{\mu} = g(\boldsymbol{w}^T \boldsymbol{x}^{\mu}) = g(\textstyle\sum_{k=1}^{N+1} w_k\, x_k^{\mu})$$

$$g(a) = \frac{\exp(a)}{1+\exp(a)} = \frac{1}{1+\exp(-a)}$$



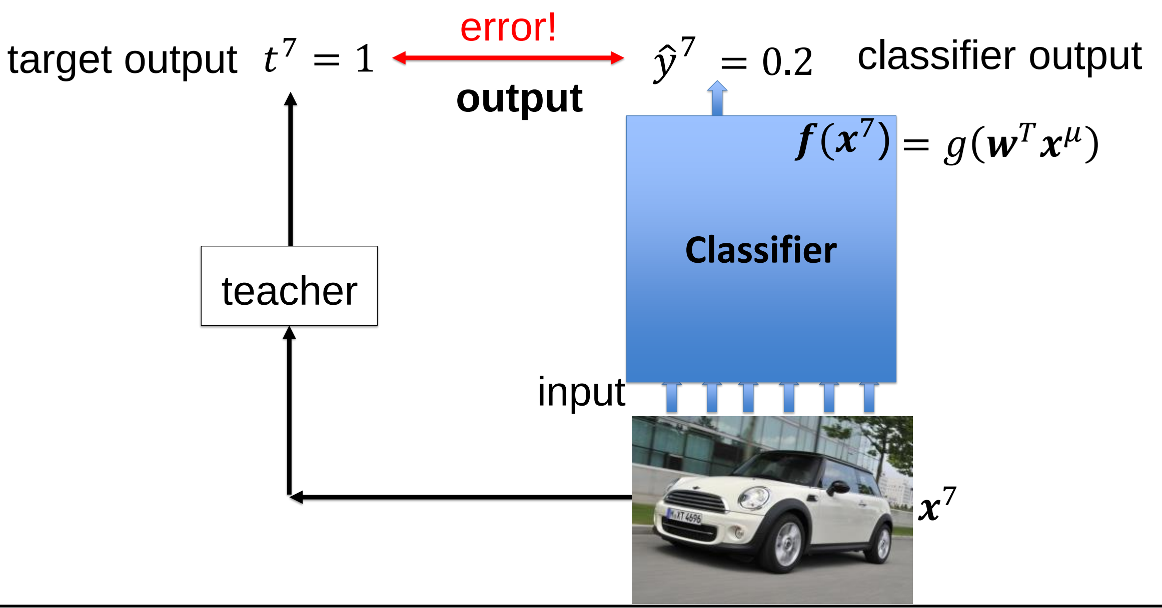


Previous slide.
We return to our choice of the nonlinear function g().
Instead of a threshold function, we can also work with a sigmoidal function.

The definition of the total input activation a is the same as before.

Instead of a step we now have a smooth transition from zero to one.

5. Supervised learning with sigmoidal output



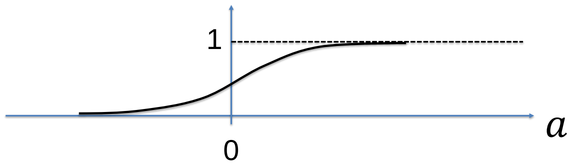
Previous slide.

The notion of mismatch in the output works for the smooth output noe analogously to the case of the binary one that we have studied before

5. Supervised learning with sigmoidal output

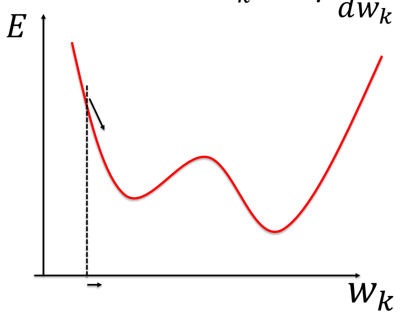
define **error**

$$E(\mathbf{w}) = \frac{1}{2}\sum_{\mu=1}^{P}\left[t^{\mu} - \hat{y}^{\mu}\right]^{2}$$

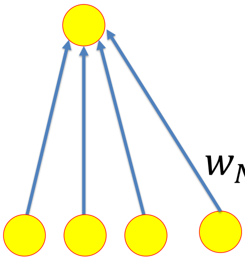


gradient descent

$$\Delta w_k = -\gamma \frac{dE}{dw_k}$$



$$\hat{y}^{\mu} = g(\mathbf{w}^{T}\mathbf{x}^{\mu})$$



Previous slide.
Since an error measure has to be always positive, we square the difference between actual output and target output.
The error function is this squared difference summed over all patterns in the data base. This error function is called the squared error or the quadratic error function. We will see other error functions in later weeks.

Most of the learning rules that we consider in this class are based on gradient descent:
The weight w_k is updated by an amount Δw_k proportional to the gradient $\frac{dE}{dw_k}$

The amplitude of the update is given by the learning rate gamma.
There is a negative sign because the aim is the REDUCE the error in each step.

With small enough learning rate, the gradient descent algorithm will end up close to a minimum of the error function. It jitters around the minimum because of the finite learning step gamma.
There is no reason that it should end up in a global minimum.

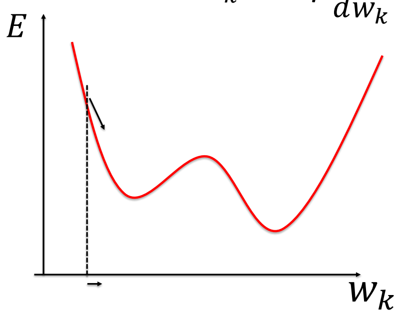
6. gradient descent

Quadratic error

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{\mu=1}^{P}\left[t^{\mu} - \hat{y}^{\mu}\right]^2$$

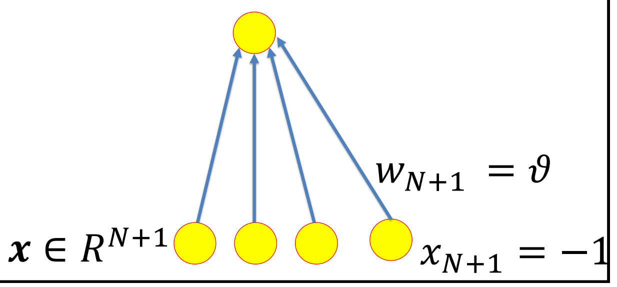
gradient descent

$$w_k = -\gamma \frac{dE}{dw_k}$$



Exercise 1.1 now:
- *calculate gradient (1 pattern)*
- *geometrical interpretation?*

$$\hat{y}^{\mu} = g(\boldsymbol{w}^T \boldsymbol{x}^{\mu})$$



Your notes.

Exercise 1.1 now:
- calculate gradient (1 pattern)
- geometrical interpretation?

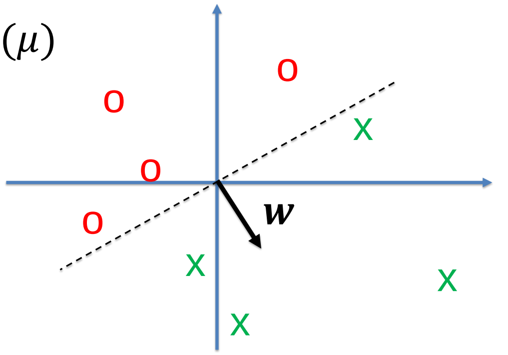
Lecture continues
at 12:02

Your notes.

6. Gradient descent algorithm

$$\Delta \boldsymbol{w} = \gamma \delta(\mu) \boldsymbol{x}^{\mu}$$

- stepsize depends on (signed) output mismatch $\delta(\mu)$ for this data point
- change implemented even if 'correctly' classified
- compare with perceptron algorithm



Previous slide.
Gradient descent can be done in two different modes:
Batch algorithm: we keep the sum over all patterns →
one update step after all patterns have been seen.
updates are repeated several times.
Online algorithm: one update step after each single pattern.
(patterns can be chosen stochastically or cyclically:
the online algo is also called stochastic gradient descent)
one 'epoche' = all patterns seen once.
repeated for many epochs until convergence

Structure of online algorithm similar to perceptron algorithm.

Main difference: the mismatch $\delta(\mu)$ is smooth here

Similar to perceptron, if a positive example is misclassified, the weight vector turns in direction of this input pattern.
The geometric picture is hence the same as for the Perceptron algorithm.

Learning outcome and conclusions for today:

- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons

Install software NOW! (Exercise Session)

Previous slide.

- Classification is equivalent to finding a separating surface in the high-dimensional input space
- This surface can be defined by the condition d(x)=0 where d is the discriminant function
- A generic data base for supervised learning requires a nonlinear discriminant function
- A simple perceptron can only implement a linear discriminant function: the separating hyperplane
- The perceptron algorithm turns the separating hyperplane in N+1 dimensions
- A quadratic error function gives rise to a stochastic gradient descent algorithm
- Geometrically the stochastic gradient descent algorithm also turns the hyperplane in N+1 dimensions, very similar to the perceptron algorithm

Reading for this week:

Bishop, Ch. 4.1.7 of
Pattern recognition and Machine Learning
or
Bishop, Ch. 3.1-3.5 of
Neural networks for pattern recognition

Motivational background reading:

Silver et al. 2017, Archive
Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

Goodfellow et al., Ch. 1 of
Deep Learning

The END

Previous slide.

The suggested reading is important, in particular if you are not able to attend the class in a given week.

In all the following weeks, the suggested reading will always be listed on slide 2, at the beginning of the lecture, so that it is easy to find.