Stochastic optimization algorithms
Lecture 9, 20200918

Neural networks and data analysis

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Note!

- The deadline for the home problem (HP1) is at 23.59.59 on Tuesday the 22nd.
- Read carefully the FAQ and the *checklist for home problem submission* (on the course web page) before submitting your solutions.
- Check also the comments for the IPP, and make any required adjustments.
- Penalties for delays, see the checklist.

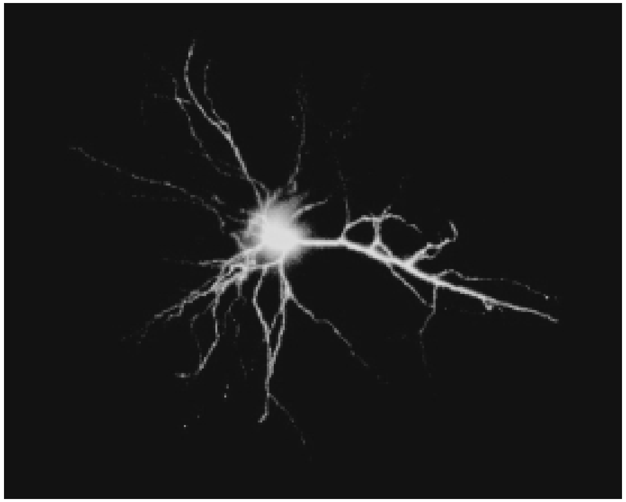
- Next week, there is no lecture on Wednesday (23rd).

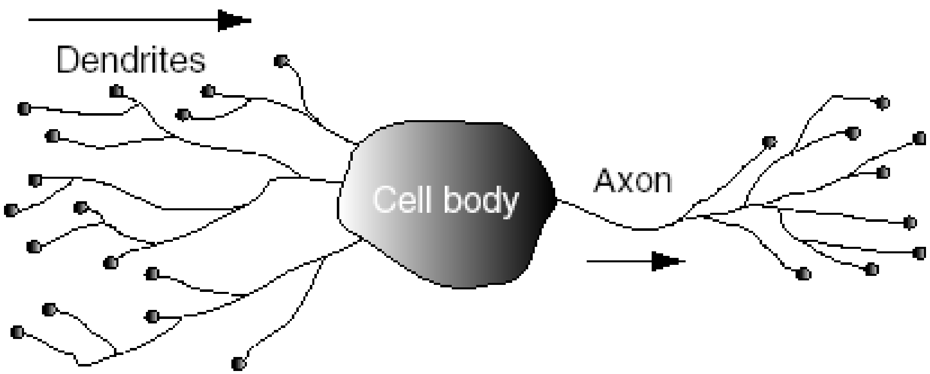
Today's learning goals

- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.

Biological neurons

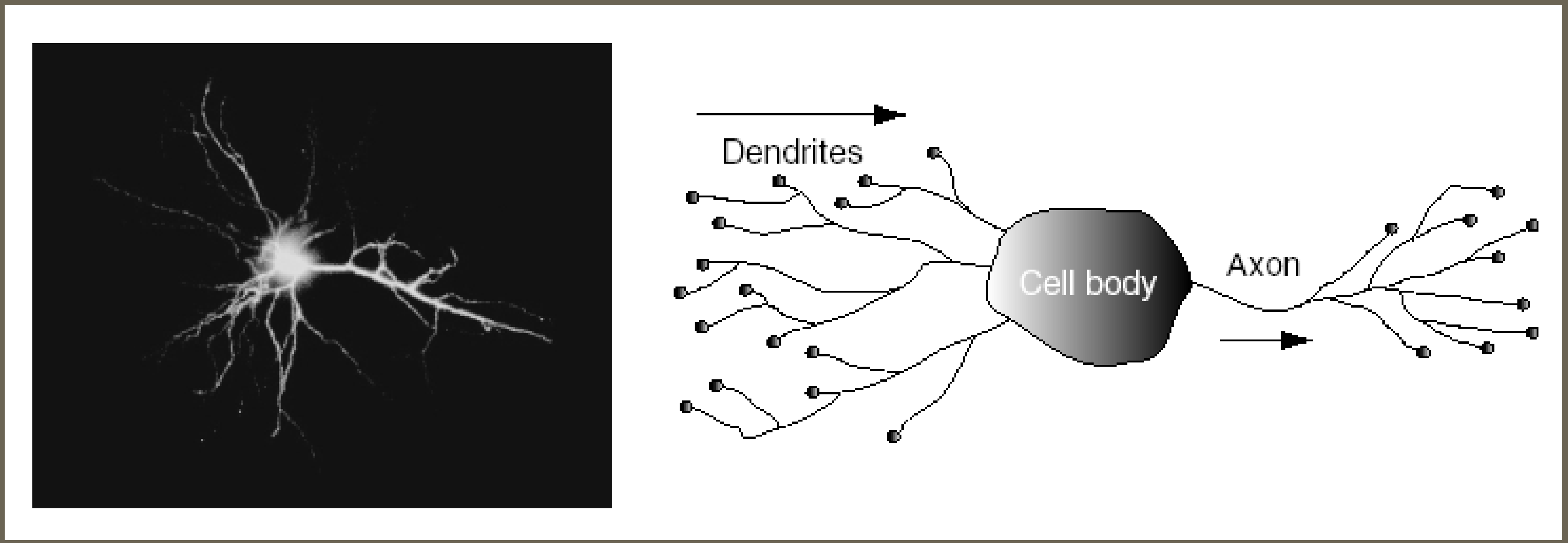
- The brains of animals consists of many (billions, in the case of humans and other mammals) **neurons**.





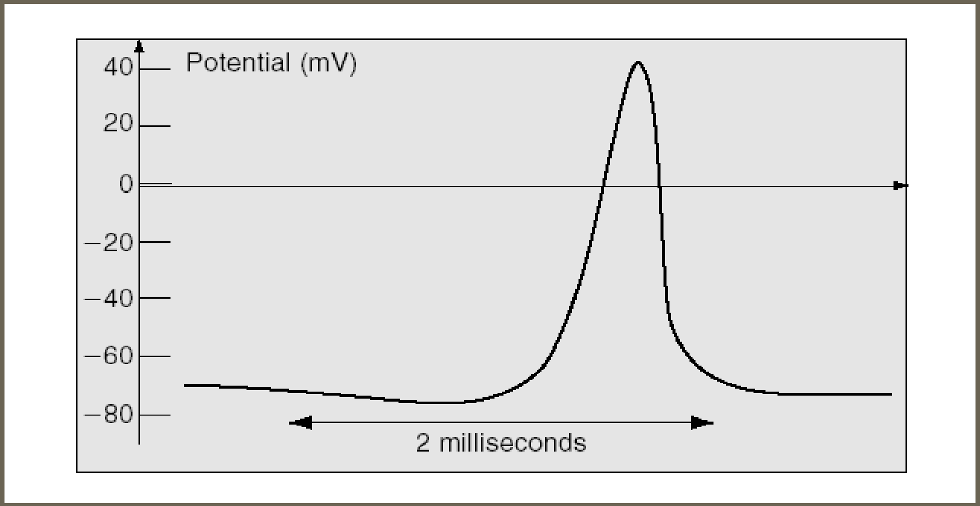
Biological neurons

- Each **neuron** is connected to many (typically thousands) of other neurons via (mostly chemical) **synapses.**



Biological neurons

- The signal flow within a neuron is electrical. The neuron makes a binary decision whether or not to fire a **spike**.



- Synapses can be either **excitatory** or **inhibitory**.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^{3} Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^3 Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^{3} Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^{3} Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^{3} Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Biological neurons

- There is a refractory period limiting the firing frequency to around 1 kHz.
- How can brains then carry out such complex tasks?
  - Parallel computation!
  - Humans: $\sim 10^{12}$ neurons, $\sim 10^{15}$ synapses.
  - 10^{12} neurons x 10^{3} Hz => 10^{15} operations/s.
- As we shall see, the same principle, i.e. using many simple, interconnected computational units, is used in artificial neural networks.

Today’s learning goals

- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.

Learning

- The information in neural networks is (mostly) stored in the connections (synapses) between neurons.
- Learning = modification of synapse strength.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Learning

- The information in neural networks is (mostly) stored in the connections (synapses) between neurons.
- Learning = modification of synapse strength.

Learning

- The information in neural networks is (mostly) stored in the connections (synapses) between neurons.
- Learning = modification of synapse strength.
- Hebbian learning:

$$\frac{dw_{ij}}{dt} = \eta x_i x_j$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Learning

- The information in neural networks is (mostly) stored in the connections (synapses) between neurons.
- Learning = modification of synapse strength.
- Hebbian learning:

$$\frac{dw_{ij}}{dt} = \eta x_i x_j$$

- Modified hebbian learning:

$$\frac{dw_{ij}}{dt} = \eta (x_i - \overline{x_i})(x_j - \overline{x_j})$$

Habituation and sensitization

- Habituation: Decreased response to neutral stimuli.
- Sensitization: Increased response to neutral stimuli following an aversive stimulus.
- Studied extensively by Kandel *et al.* (Nobel Prize 2000)
- Kandel studied the sea slug Aplysia, due to
  - ...its very thick neurons (easier to measure) and,
  - ...its rather small number of neurons (around 20000).

Habituation and sensitization

- Habituation: Decreased response to neutral stimuli.
- Sensitization: Increased response to neutral stimuli following an aversive stimulus.
- Studied extensively by Kandel *et al.* (Nobel Prize 2000)
- Kandel studied the sea slug Aplysia, due to
  - ...its very thick neurons (easier to measure) and,
  - ...its rather small number of neurons (around 20000).

Habituation and sensitization

- Habituation: Decreased response to neutral stimuli.
- Sensitization: Increased response to neutral stimuli following an aversive stimulus.
- Studied extensively by Kandel *et al.* (Nobel Prize 2000)
- Kandel studied the sea slug Aplysia, due to
  - ...its very thick neurons (easier to measure) and,
  - ...its rather small number of neurons (around 20000).

Habituation and sensitization

- Eric Kandel's Nobel lecture (highly recommended!):
  - http://nobelprize.org/mediaplayer/?id=898

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Habituation and sensitization

- Basic conclusions from Kandel's work
  - Short-term learning depends on changes in the release of neurotransmitters.
  - Long-term learning depends on changes in gene regulation that, in turn, modify synapses (more or less) permanently.

Habituation and sensitization

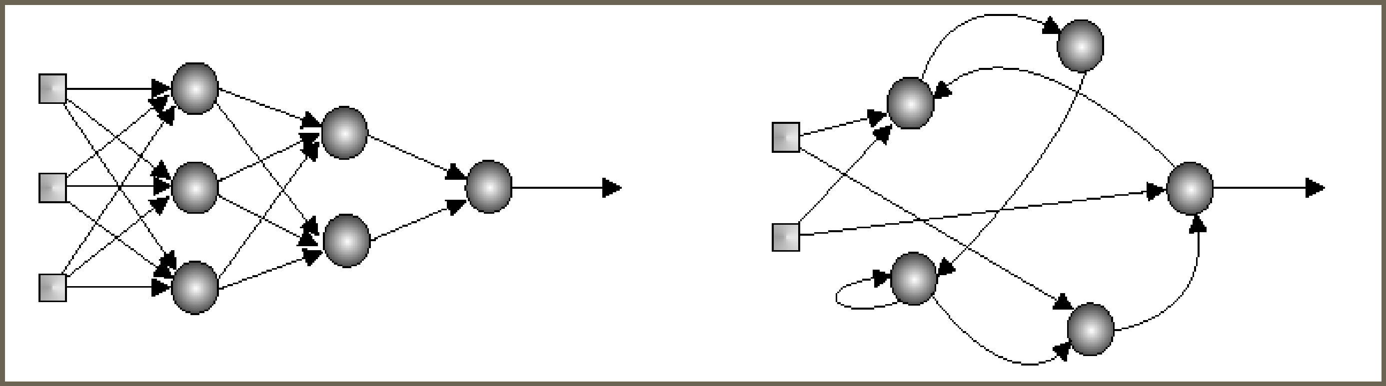
- Basic conclusions from Kandel's work
  - Short-term learning depends on changes in the release of neurotransmitters.
  - Long-term learning depends on changes in gene regulation that, in turn, modify synapses (more or less) permanently.

Today’s learning goals

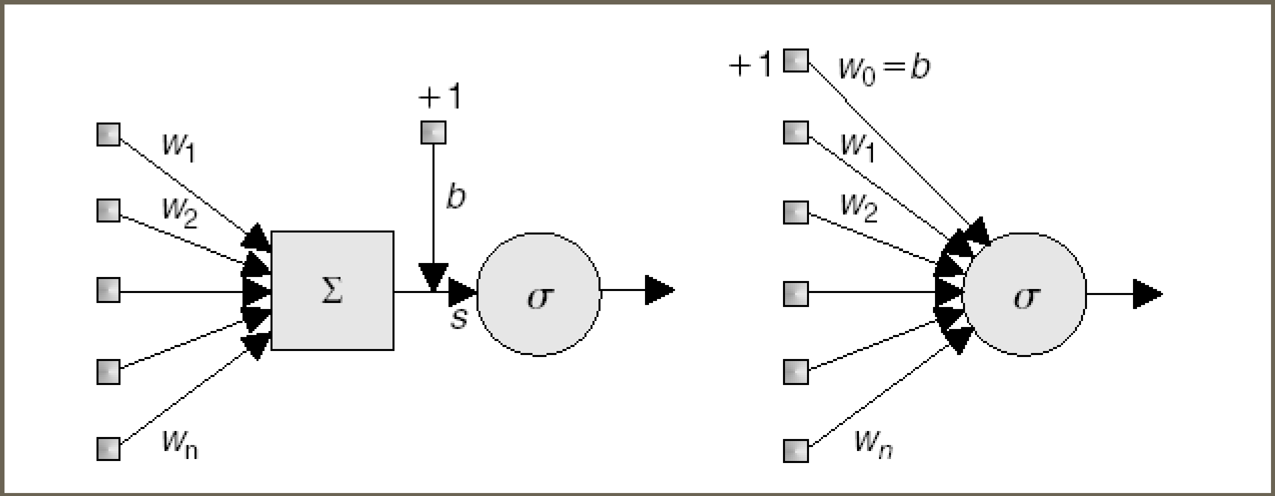
- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.

Artificial neural networks

- Artificial neural networks (ANNs) consist of simple, interconnected computational units.
- Two kinds:
  - Feedforward neural networks (FFNNs)
  - Recurrent neural networks (RNNs)

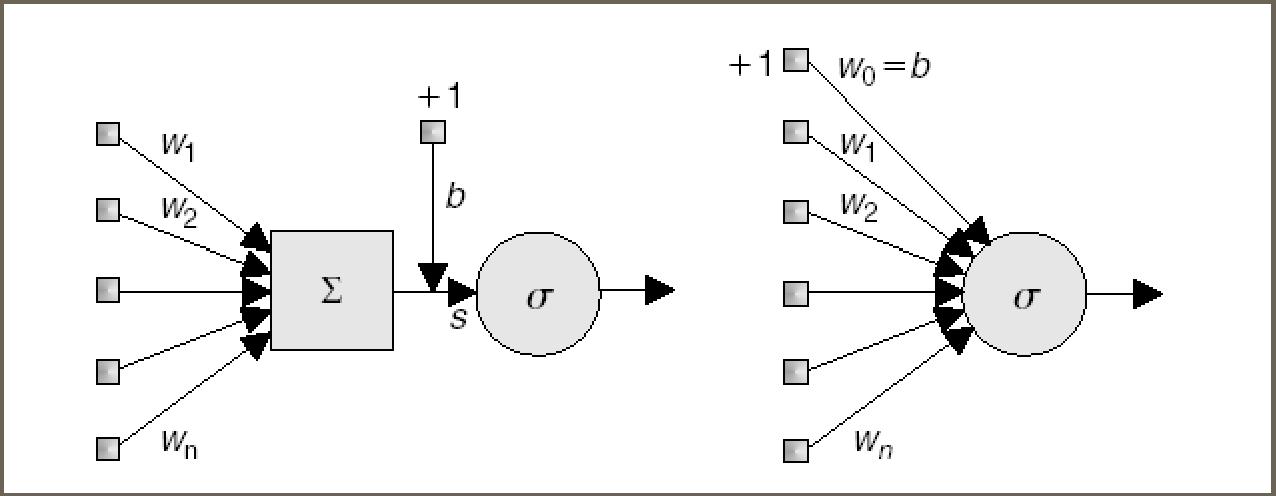


Neurons (in ANNs)



- Output: $y = \sigma\left(\sum_{j=1}^{n} w_j x_j + b\right) \equiv \sigma\left(\sum_{j=0}^{n} w_j x_j\right)$

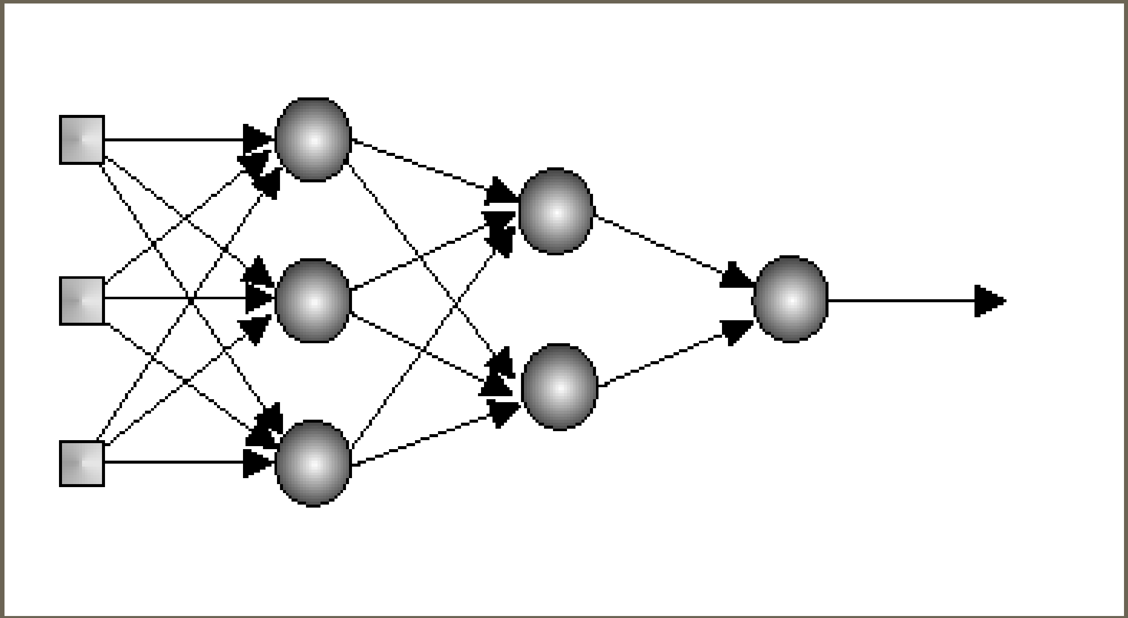
Neurons (in ANNs)



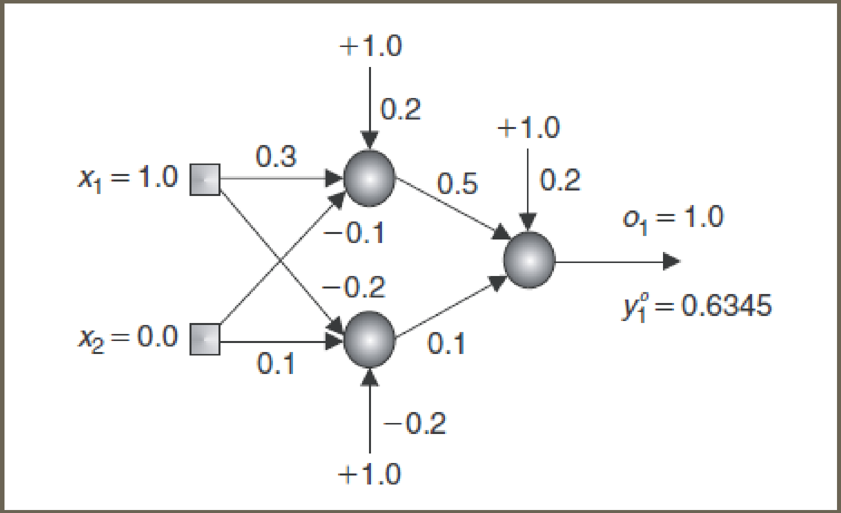
- Output: $y = \sigma\left(\sum_{j=1}^{n} w_j x_j + b\right) \equiv \sigma\left(\sum_{j=0}^{n} w_j x_j\right)$

For example the logistic sigmoid $\sigma(z) = 1/(1 + e^{-cz})$, where c is a constant.

Artificial neural networks



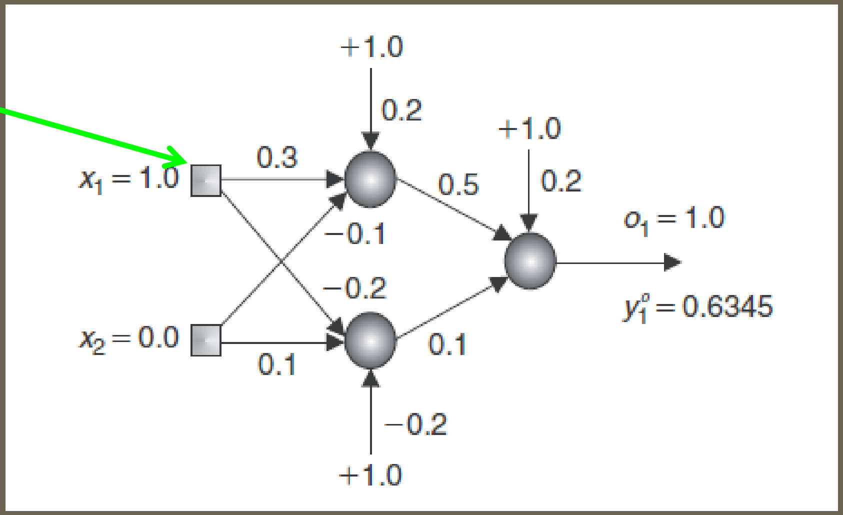
Example A.1



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example A.1

Input unit (simply mediates signal – no other computation).

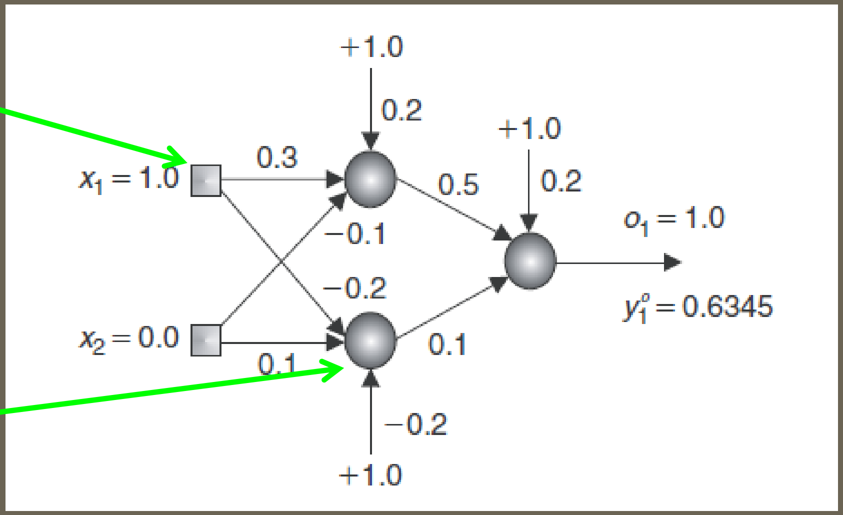


Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example A.1

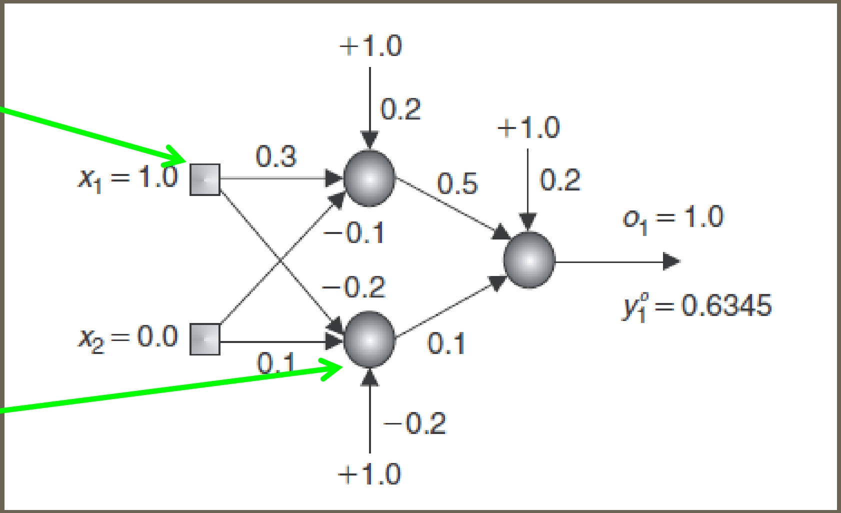
Input unit (simply mediates signal – no other computation).

Neuron – computation as in the previous slide.



Example A.1

Input unit (simply mediates signal – no other computation).

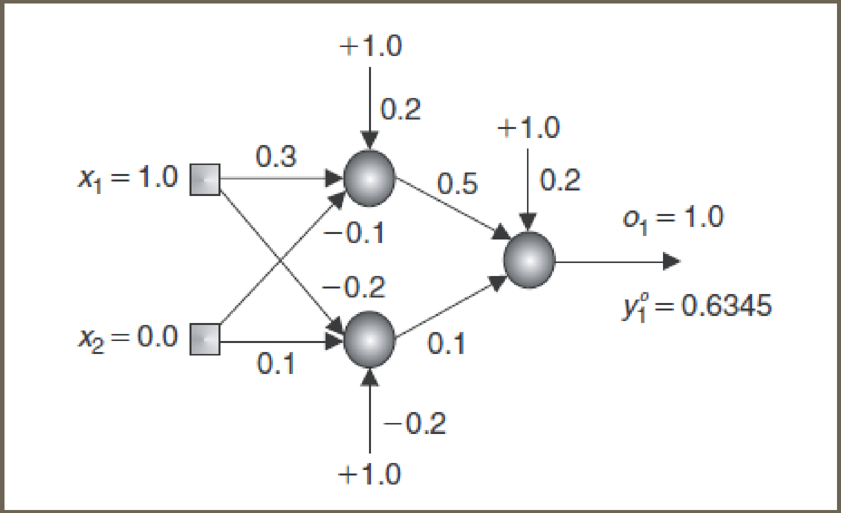


Neuron – computation as in the previous slides.

- You should know how to compute the output of an FFNN, but you do **<u>not</u>** need to know backpropagation (for this course, at least).

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

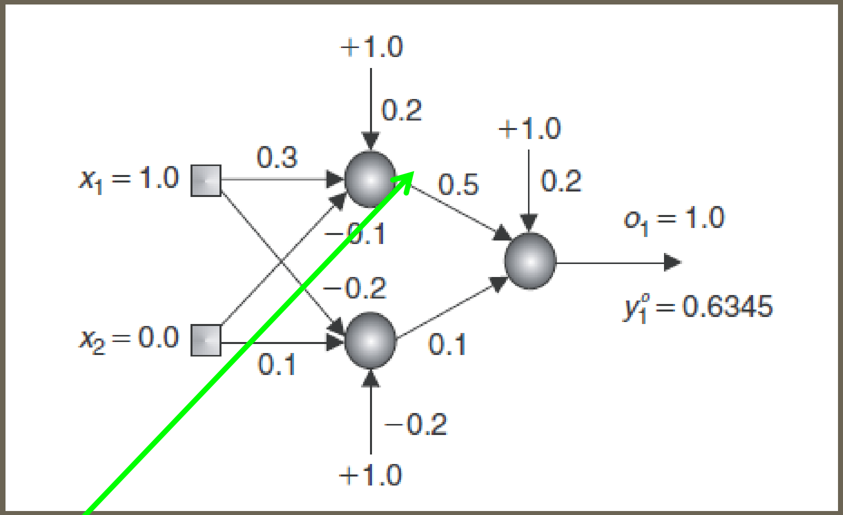
Example A.1



- You should know how to compute the output of an FFNN, but you do **not** need to know backpropagation (for this course, at least).

CHALMERS

Example A.1



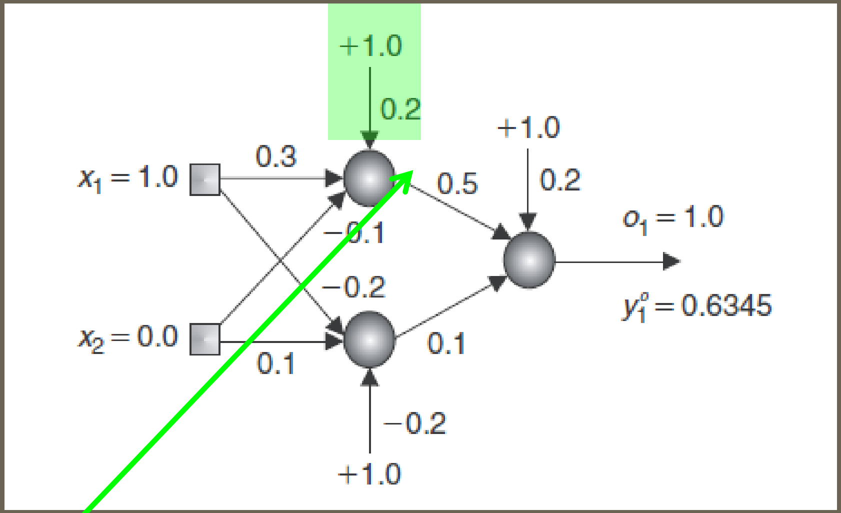
Output from the first neuron in the hidden layer:

$$y_1^H = \sigma\left(\sum_{p=0}^{2} w_{1p}^{I\to H} y_p^I\right) = \sigma(0.2 \times 1.0 + 0.3 \times 1.0 - 0.1 \times 0)$$

$$= \sigma(0.5) = \frac{1}{1 + e^{-0.5}} = 0.6225,$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

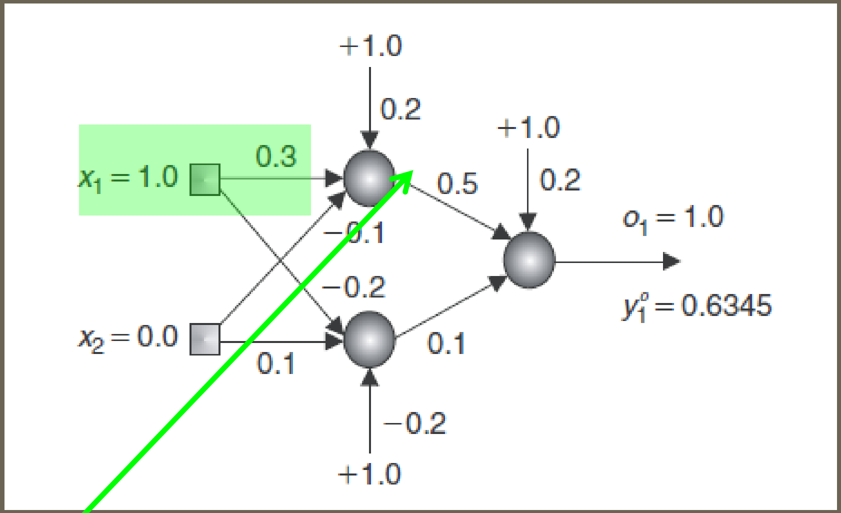
Example A.1



Output from the first neuron in the hidden layer:

$$y_1^H = \sigma\left(\sum_{p=0}^{2} w_{1p}^{I\to H} y_p^I\right) = \sigma(0.2 \times 1.0 + 0.3 \times 1.0 - 0.1 \times 0)$$
$$= \sigma(0.5) = \frac{1}{1+e^{-0.5}} = 0.6225,$$

Example A.1



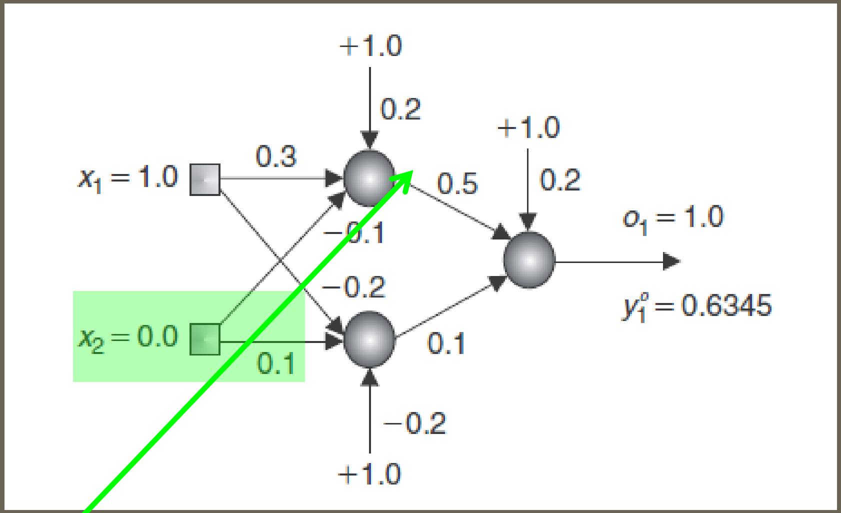
Output from the first neuron in the hidden layer:

$$y_1^H = \sigma\left(\sum_{p=0}^{2} w_{1p}^{I\to H} y_p^I\right) = \sigma(0.2 \times 1.0 + 0.3 \times 1.0 - 0.1 \times 0)$$

$$= \sigma(0.5) = \frac{1}{1 + e^{-0.5}} = 0.6225,$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example A.1



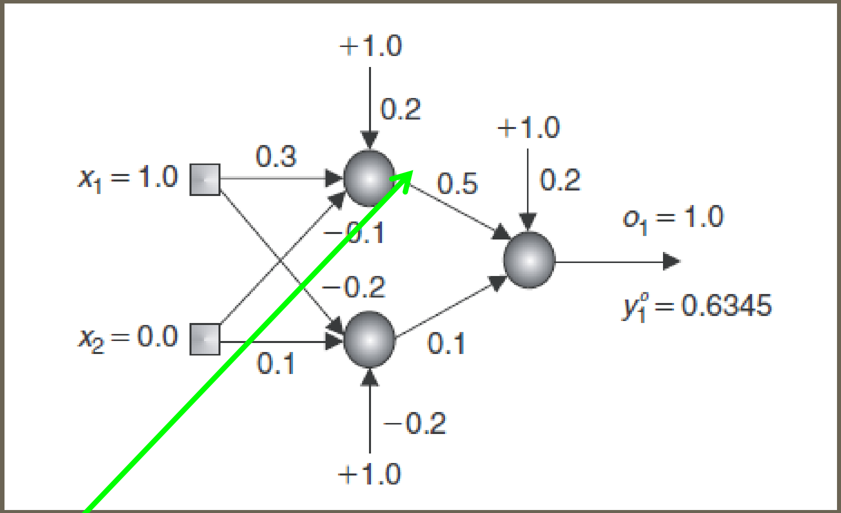
Output from the first neuron in the hidden layer:

$$y_1^H = \sigma\left(\sum_{p=0}^{2} w_{1p}^{I\to H} y_p^I\right) = \sigma(0.2 \times 1.0 + 0.3 \times 1.0 - 0.1 \times 0)$$

$$= \sigma(0.5) = \frac{1}{1+e^{-0.5}} = 0.6225,$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example A.1



Output from the first neuron in the hidden layer:

$$y_1^H = \sigma\left(\sum_{p=0}^{2} w_{1p}^{I\to H} y_p^I\right) = \sigma(0.2 \times 1.0 + 0.3 \times 1.0 - 0.1 \times 0)$$

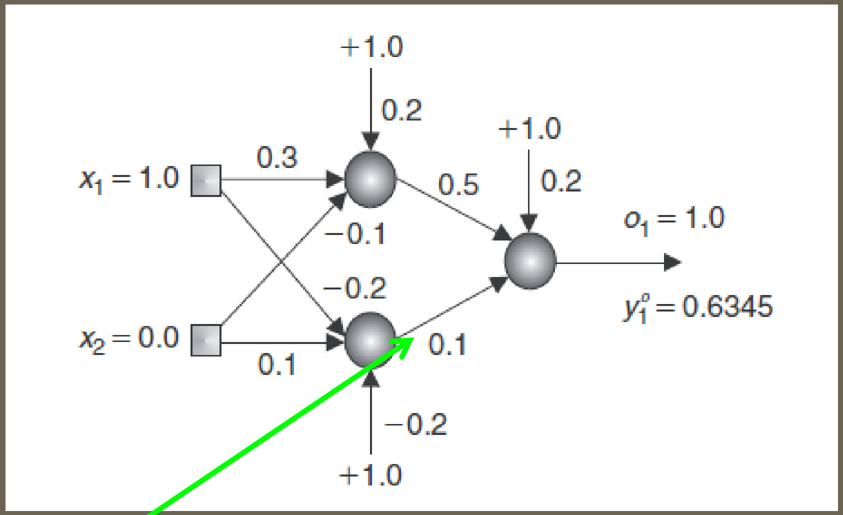
$$= \sigma(0.5) = \frac{1}{1+e^{-0.5}} = 0.6225,$$

In this example, $\sigma(z) = 1/(1+e^{-cz})$, with $c = 1$.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Example A.1



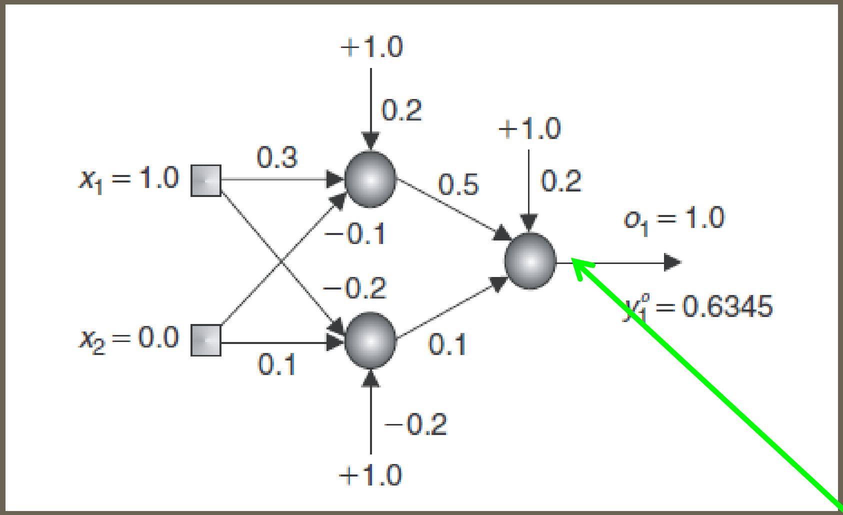
Output from the second neuron in the hidden layer:

$$y_2^H = \sigma\left(\sum_{p=0}^{2} w_{2p}^{I\to H} y_p^I\right) = \sigma(-0.2 \times 1.0 - 0.2 \times 1.0 + 0.1 \times 0)$$

$$= \sigma(-0.4) = \frac{1}{1 + e^{0.4}} = 0.4013.$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

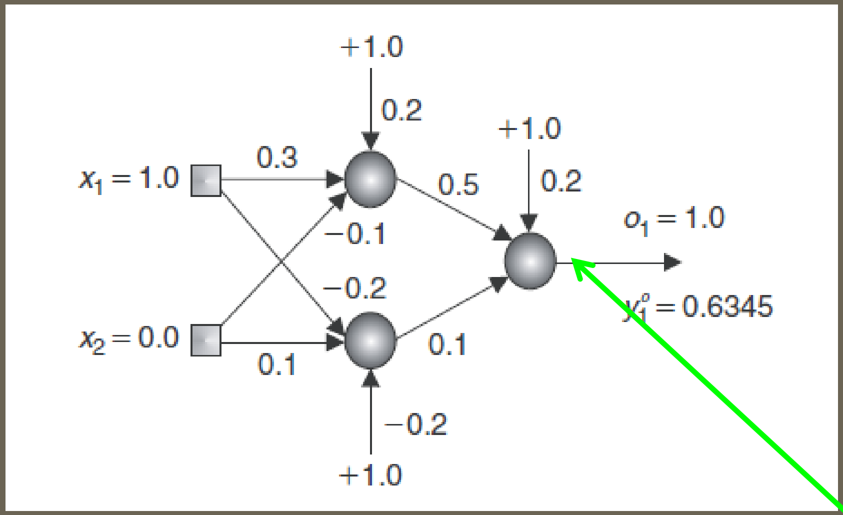
Example A.1



Output from the neuron in the output layer:

$$
\begin{aligned}
y_1^O &= \sigma\left(\sum_{s=0}^{2} w_{1s}^{H\to O} y_s^H\right) = \sigma(0.2 \times 1.0 + 0.5 \times 0.6225 + 0.1 \times 0.4013) \\
&= \sigma(0.5514) = 0.6345.
\end{aligned}
$$

Example A.1



Output from the neuron in the output layer:

$$y_1^O = \sigma\left(\sum_{s=0}^{2} w_{1s}^{H\to O} y_s^H\right) = \sigma(0.2 \times 1.0 + 0.5 \times 0.6225 + 0.1 \times 0.4013)$$

$$= \sigma(0.5514) = 0.6345.$$

y_1^H y_2^H

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.

Training methods

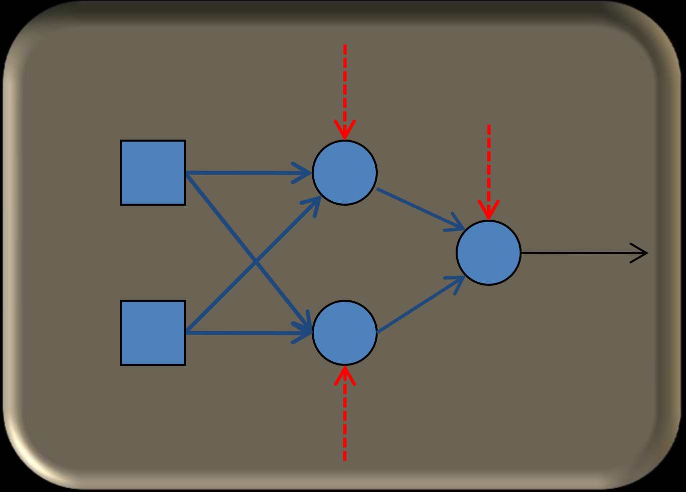
- **Strongly guided learning**: input-output pairs available, feedback (error signal) given (to the network) for every input-output pair.
- **Weakly guided learning**: No input-output pairs. Instead, the feedback is given at (for example) the end of a lengthy simulation (see, for instance, the robot example (videos) from Lecture 1).

Using a GA to optimize an ANN

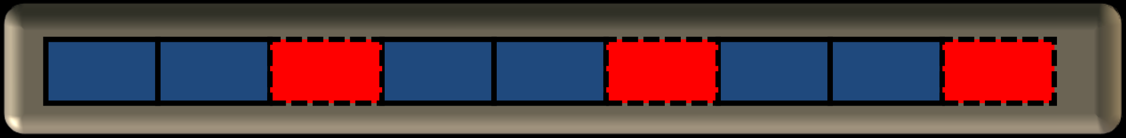
- In many cases, the data set consists of input-output pairs, such that an error signal can be formed, based on the difference between the desired and actual outputs.
- In those cases, one can apply backpropagation (which requires the gradient of the error signal).
- In other cases, one might not have a set of input-output pairs – Instead feedback (output) may only be given, say, at the end of a long evaluation => backpropagation is not suitable.
- In such cases, one can apply a GA to train the network.

Using a GA to optimize an ANN

Example: 2-2-1 network (6 weights (dark blue), 3 biases (red))



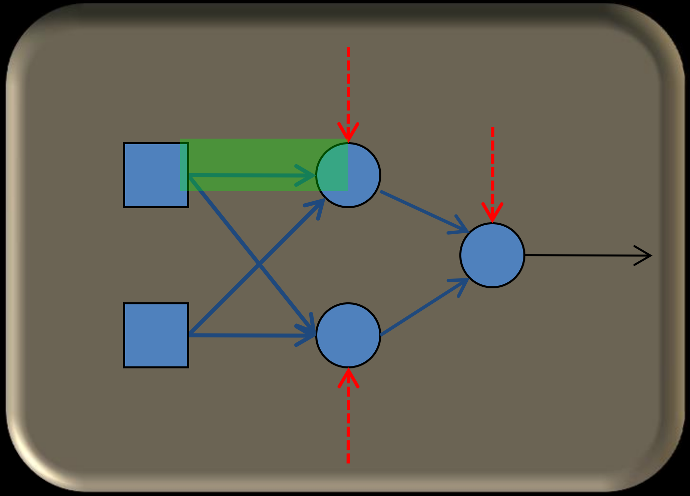
Encode in chromosomes with 9 real-valued genes:



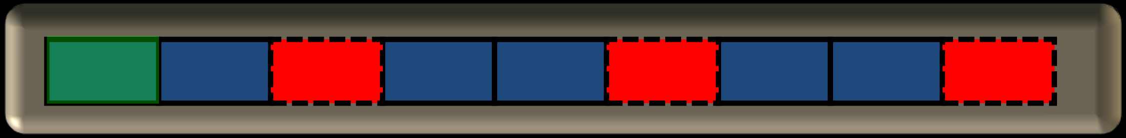
Evaluate and assign fitness values, then apply selection, crossover and mutation (i.e. as usual). The evaluation varies from case to case, of course.

Using a GA to optimize an ANN

Example: 2-2-1 network (6 weights (dark blue), 3 biases (red))



Encode in chromosomes with 9 real-valued genes:

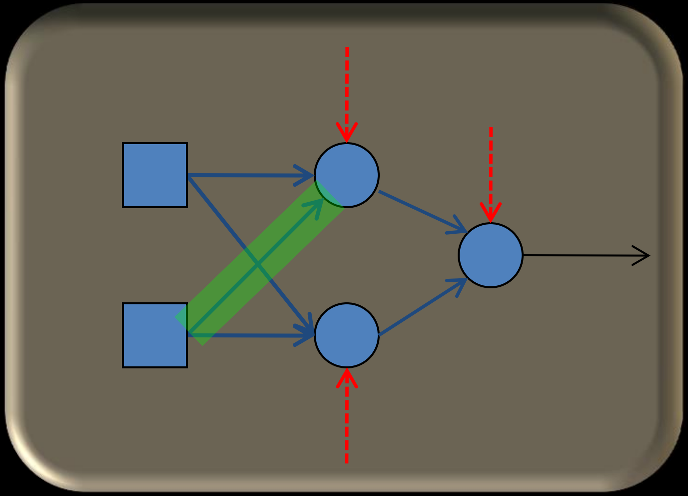


Evaluate and assign fitness values, then apply selection, crossover and mutation (i.e. as usual). The evaluation varies from case to case, of course.

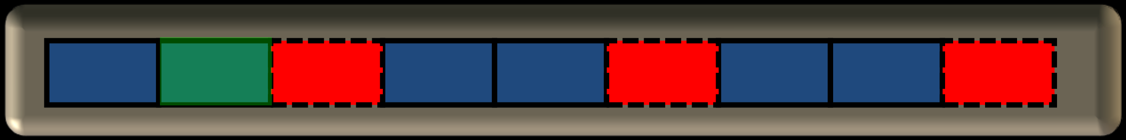
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Using a GA to optimize an ANN

Example: 2-2-1 network (6 weights (dark blue), 3 biases (red))



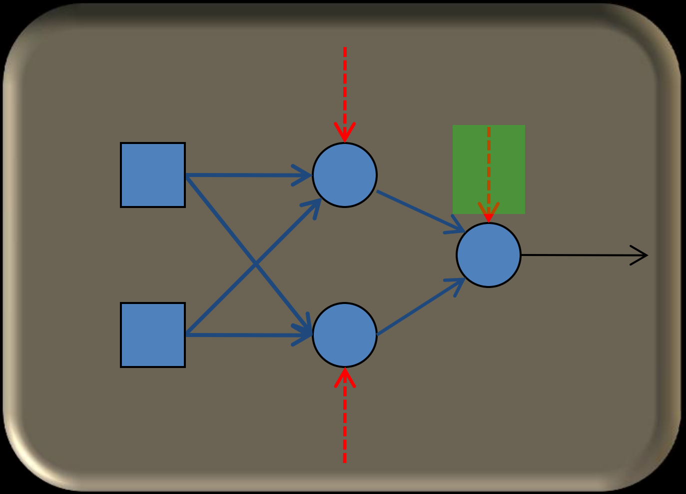
Encode in chromosomes with 9 real-valued genes:



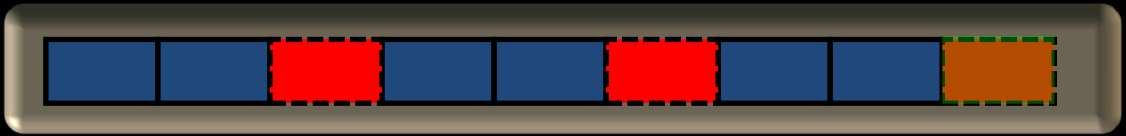
Evaluate and assign fitness values, then apply selection, crossover and mutation (i.e. as usual). The evaluation varies from case to case, of course.

Using a GA to optimize an ANN

Example: 2-2-1 network (6 weights (dark blue), 3 biases (red))



Encode in chromosomes with 9 real-valued genes:



Evaluate and assign fitness values, then apply selection, crossover and mutation (i.e. as usual). The evaluation varies from case to case, of course.

Today's learning goals

- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Experiment design and overfitting

- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  – Training
  – Validation (for determining when to quit training)
  – Test (for final evaluation, not used during training)

Experiment design and overfitting

- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  - Training
  - Validation (for determining when to quit training)
  - Test (for final evaluation, not used during training)

Experiment design and overfitting

- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  - Training
  - Validation (for determining when to quit training)
  - Test (for final evaluation, not used during training)

Experiment design and overfitting

- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  - Training
  - Validation (for determining when to quit training)
  - Test (for final evaluation, not used during training)

Experiment design and overfitting

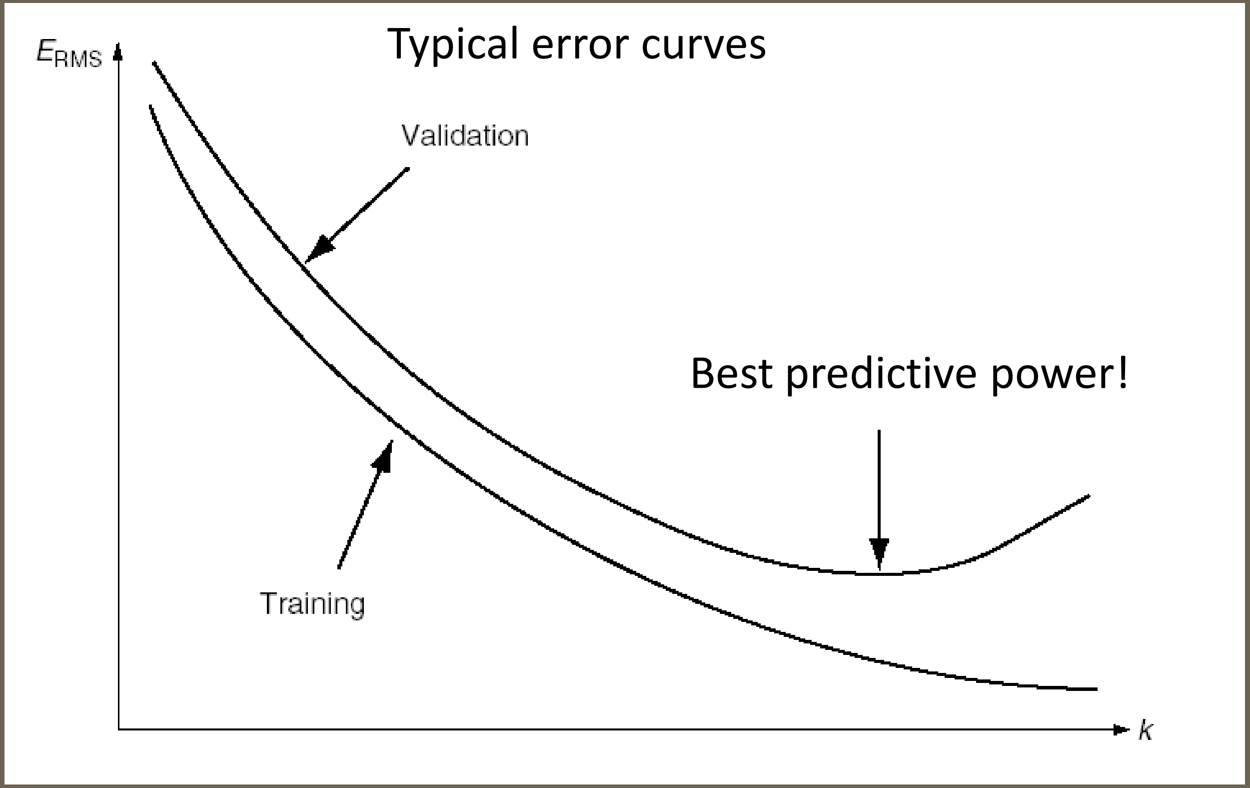
- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  - Training
  - Validation (for determining when to quit training)
  - Test (for final evaluation, not used during training)

Experiment design and overfitting

- When fitting a function (or a neural network, or some other structure) to a data set of limited size, there is a risk of fitting the noise => decrease in predictive power.
- This (fitting noise) is called **overfitting**.
- To avoid overfitting, one typically divides the available data into three sets:
  - Training
  - Validation (for determining when to quit training)
  - Test (for final evaluation, not used during training)

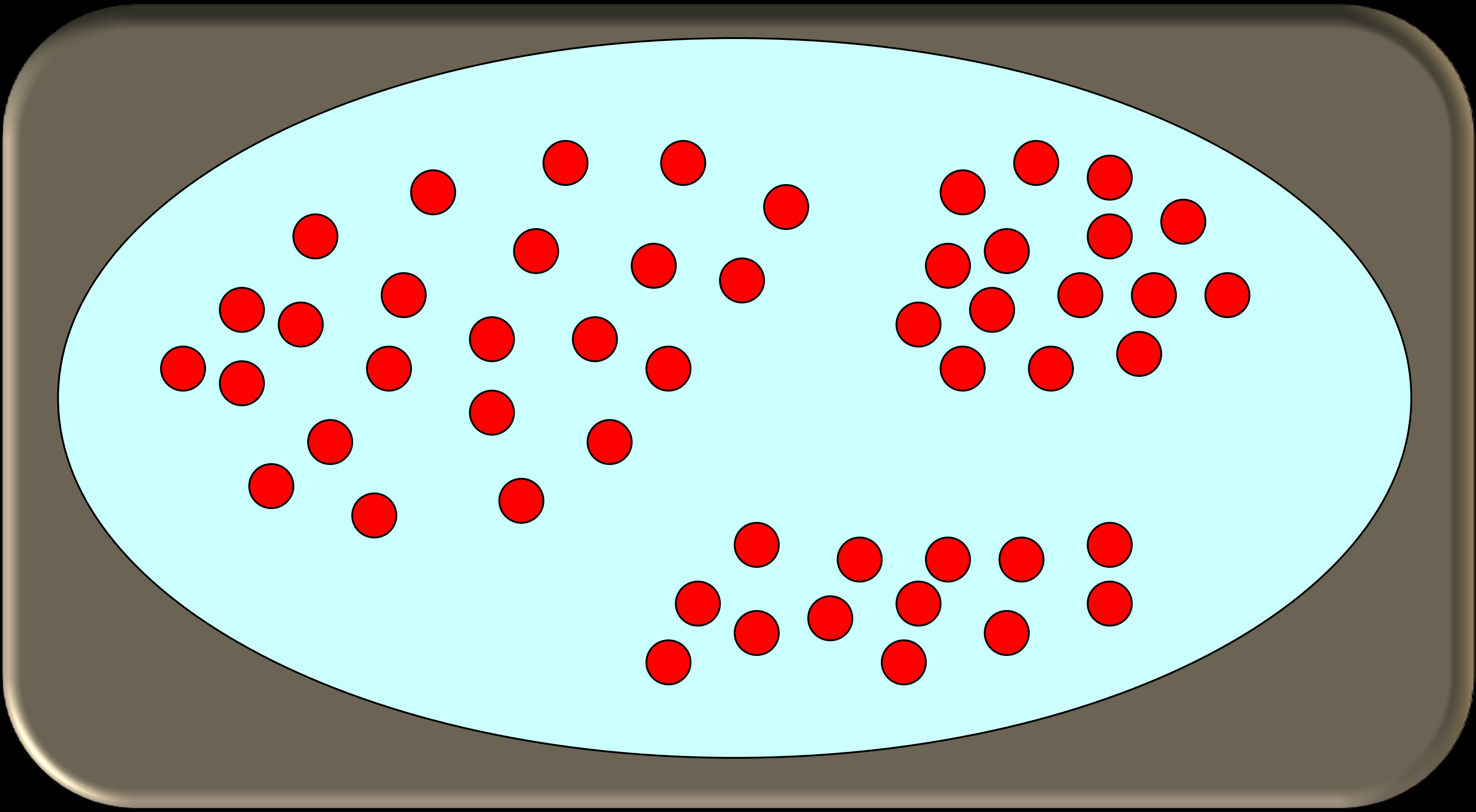
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Experiment design and overfitting



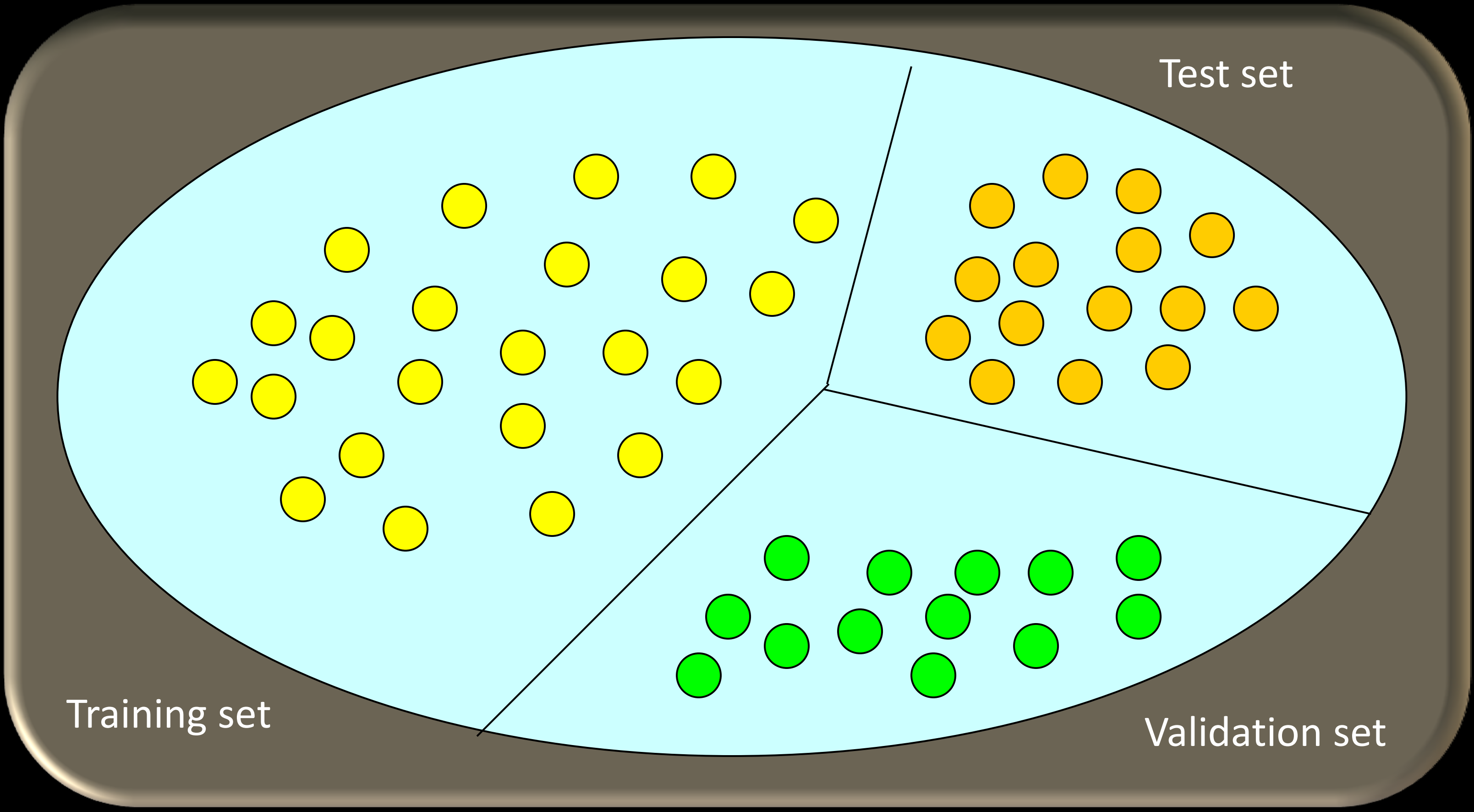
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Experiment design and overfitting



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Experiment design and overfitting



Procedure (avoiding overfitting)

- Use the **training data** to give feedback to the training algorithm.
- Use the **validation data** to determine when to stop the training – measure the error (or fitness) over the validation set, but do *not* make the result of the measurement available to the training algorithm.
- Use the (previously unused) **test data** (after completing the training) to test the results (on the selected system, i.e. the system with best validation performance).

Procedure (avoiding overfitting)

- Use the **training data** to give feedback to the training algorithm.
- Use the **validation data** to determine when to stop the training – measure the error (or fitness) over the validation set, but do *not* make the result of the measurement available to the training algorithm.
- Use the (previously unused) **test data** (after completing the training) to test the results (on the selected system, i.e. the system with best validation performance).

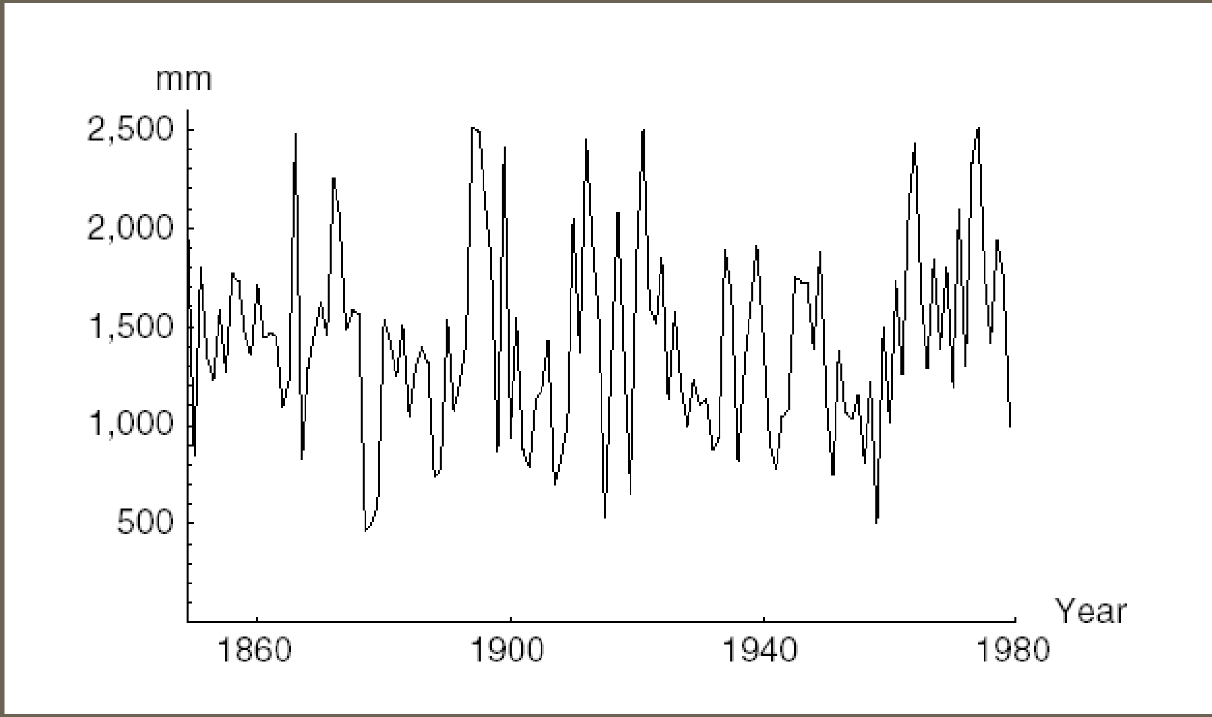
Procedure (avoiding overfitting)

- Use the **training data** to give feedback to the training algorithm.
- Use the **validation data** to determine when to stop the training – measure the error (or fitness) over the validation set, but do *not* make the result of the measurement available to the training algorithm.
- Use the (previously unused) **test data** (after completing the training) to test the results (on the selected system, i.e. the system with best validation performance).

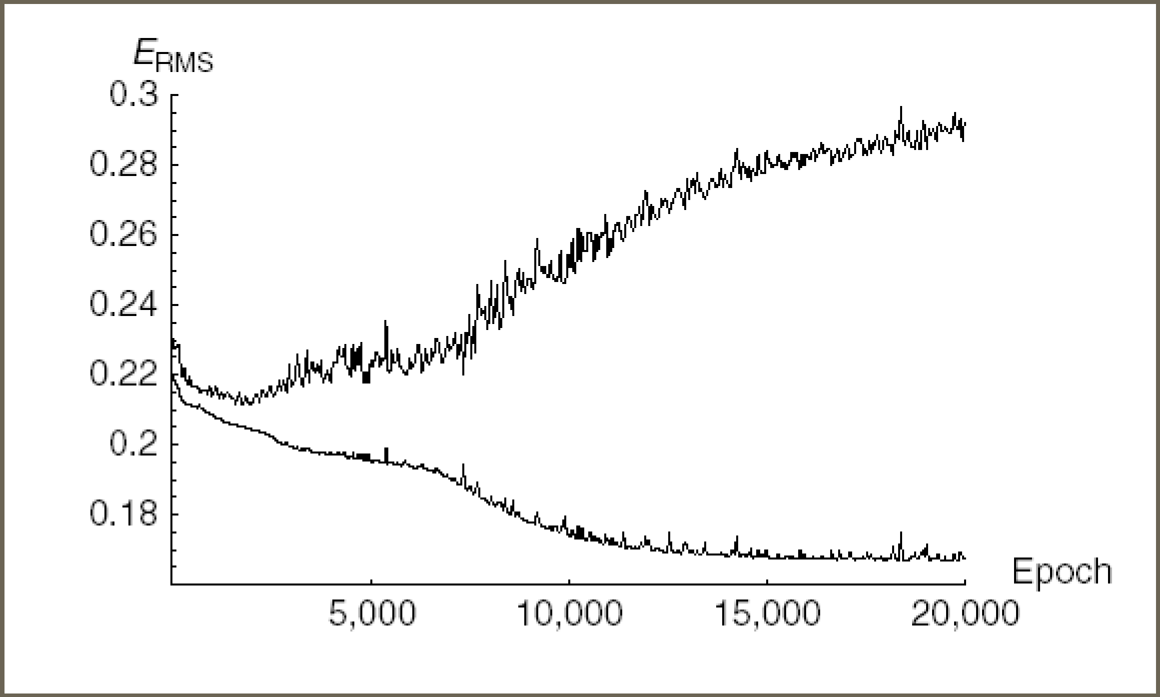
Example: Time series prediction

- Example C.4: Using a neural network to predict annual rainfall.
- In this case, each data point consists of an input-output pair:
  - Input: Rainfall in years T-1, T-2, T-3, T-4, T-5
  - Output: Rainfall in year T.
- A total of 90 training data points and 31 validation data points.
- Objective: Find an FFNN whose output differs as little as possible from the actual output.

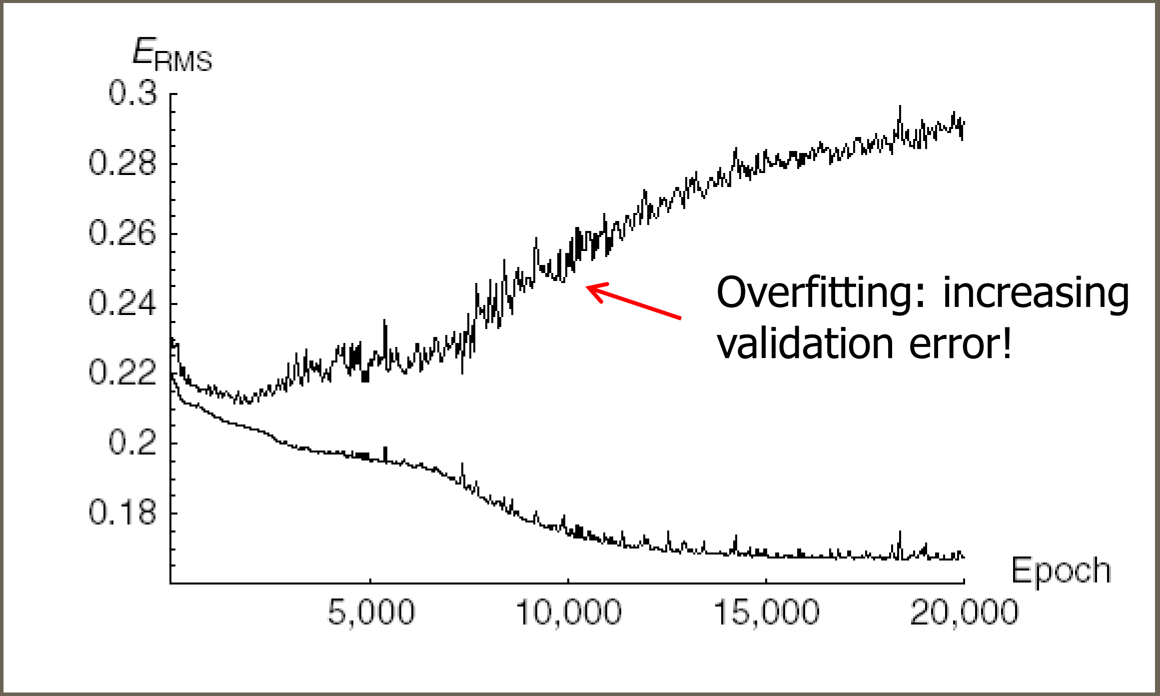
Example: Time series prediction



Example: Time series prediction

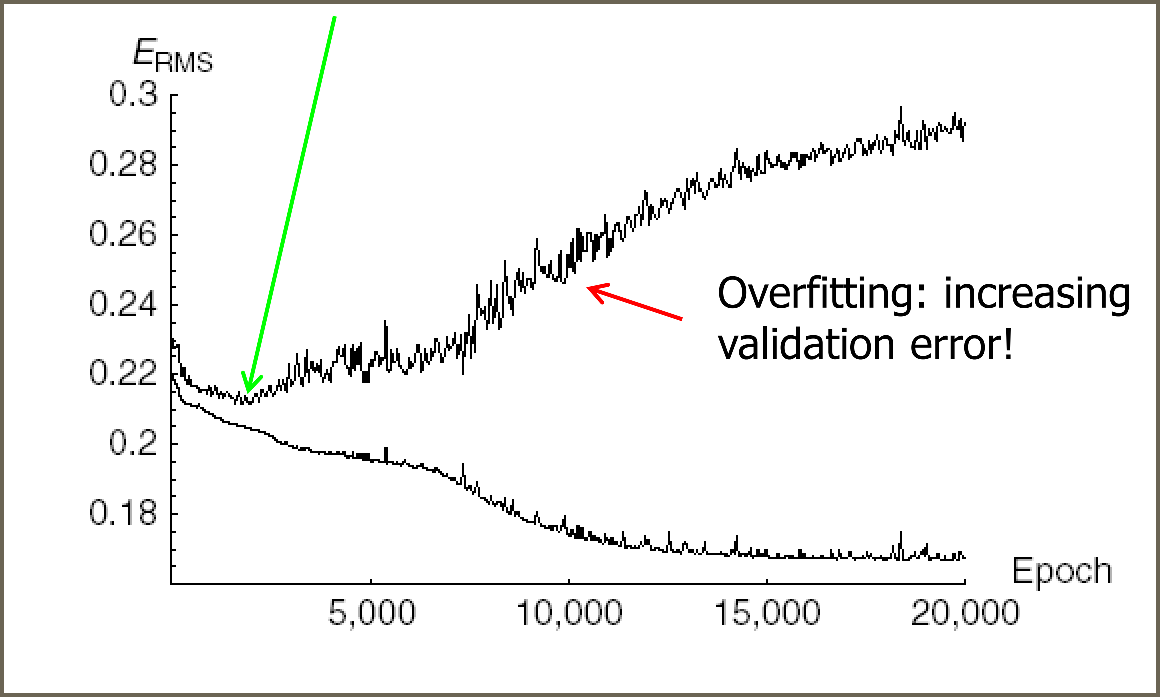


Example: Time series prediction



Example: Time series prediction

Best validation performance



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Describe the biological background of neural networks.
  - Describe basic learning: Habituation and sensitization.
  - Compute the output of a feedforward neural network.
  - Apply a GA to optimize an artificial neural network.
  - Describe and discuss the concept of overfitting, as well as methods for avoiding it.