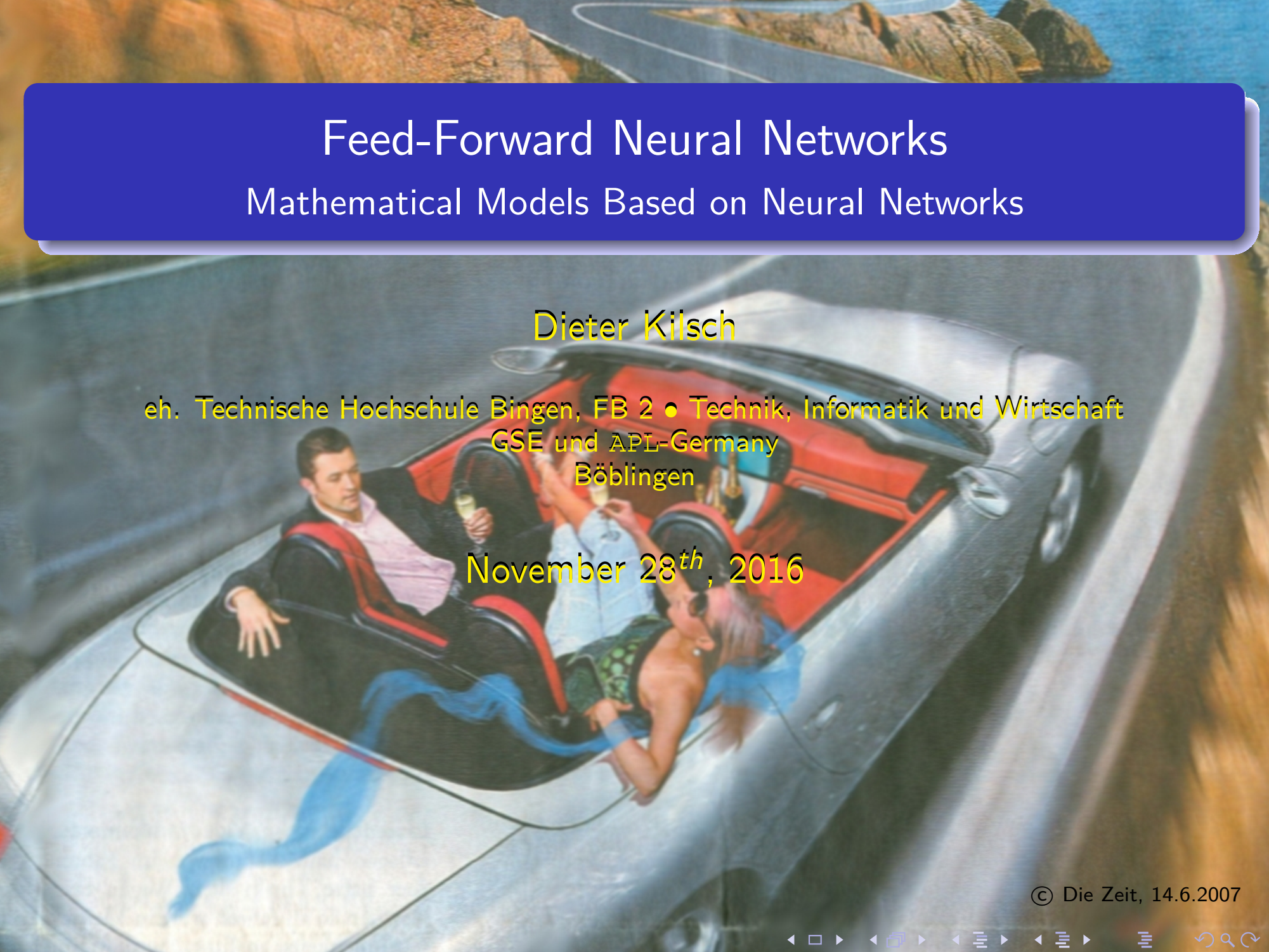
Feed-Forward Neural Networks

Mathematical Models Based on Neural Networks

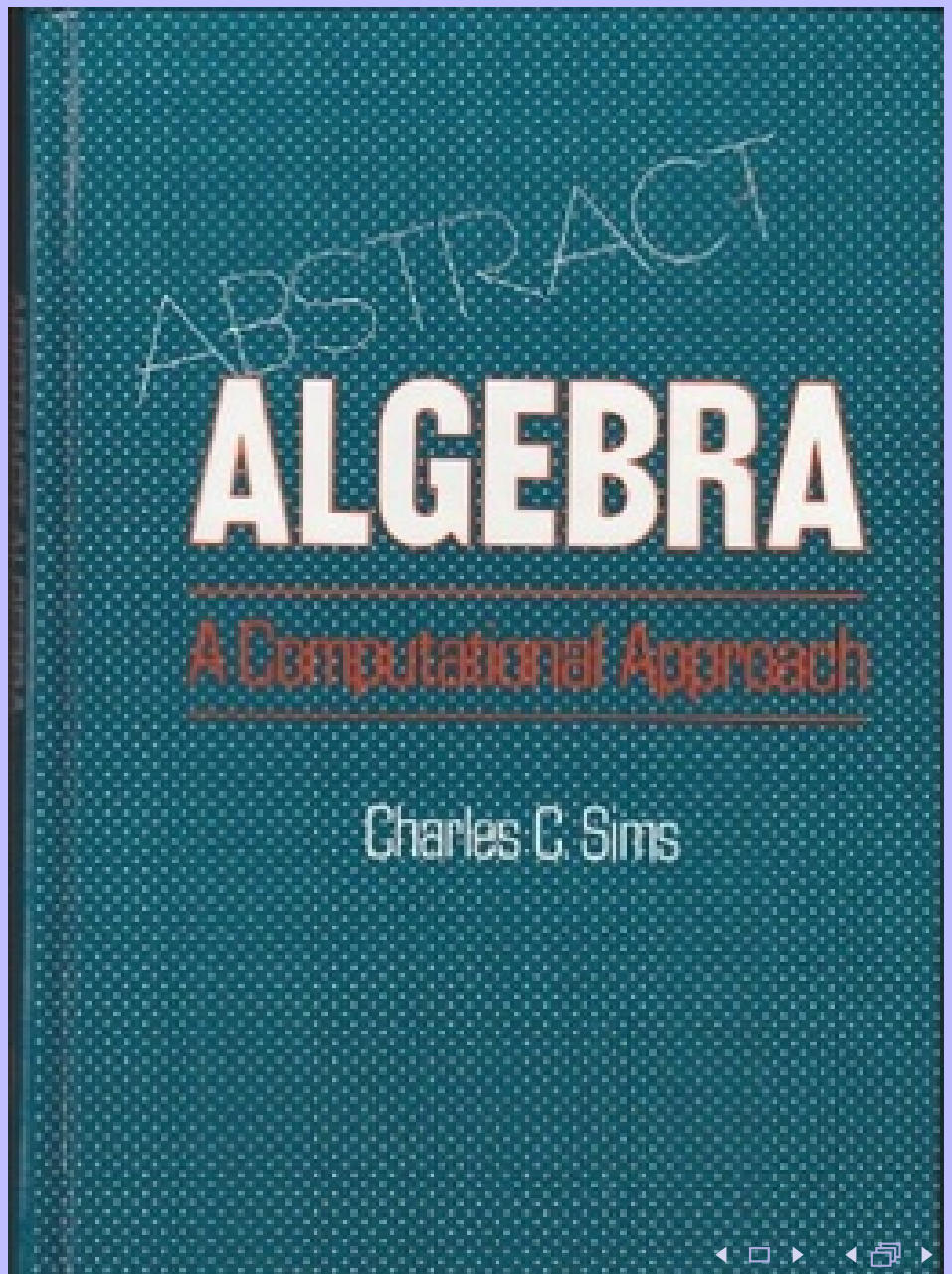
Dieter Kilsch

eh. Technische Hochschule Bingen, FB 2 • Technik, Informatik und Wirtschaft
GSE und API-Germany
Böblingen

November 28th, 2016



My first „date“ with APL: 1984



My first APL-conference: Ст. Петерсбург 1992

Convenience by Autonomous Vehicles

2007: Vision

The car manages itself,

the driver's mind

is free to enjoy live.

Convenience by Autonomous Vehicles

2007: Vision

The car manages itself,

the driver's mind

is free to enjoy live.

Convenience by Autonomous Vehicles

heute: ? Vision ?

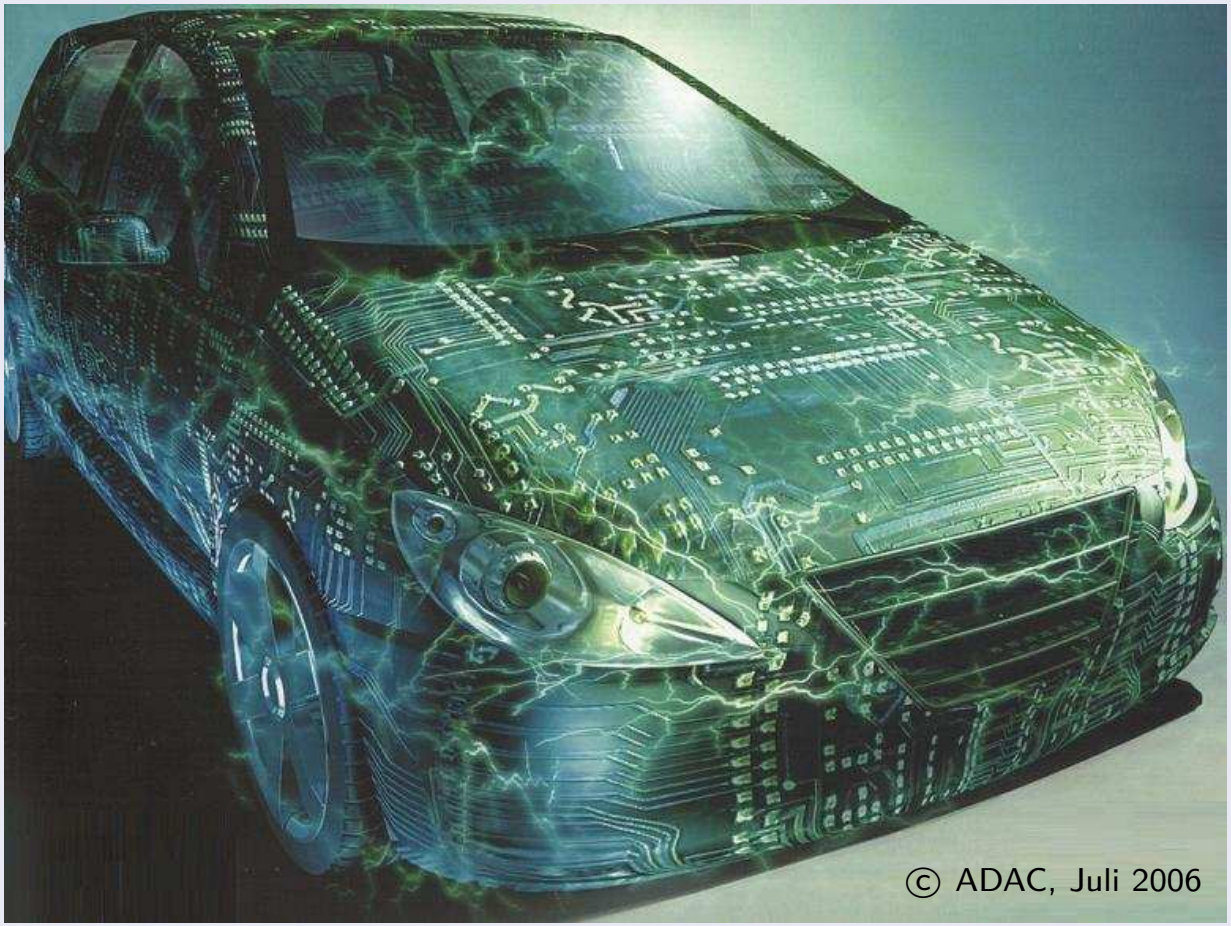
The car manages itself,

the driver's mind

is free to enjoy live.

Vision: Requirements

Vision



The car senses

and controls.

Autonomous Cars Come True

“First Autonomes Car on Public Roads”

TU Braunschweig, https://www.tu-braunschweig.de/presse/medien/presseinformationen?year=2010&pinr=133

The car drives, the driver enjoys . . .

Leonie 8.10.2010

Weltweit erstes automatisches Fahren im realen Stadtverkehr

Forschungsfahrzeug „Leonie“ fährt automatisch auf dem Braunschweiger Stadtring

Weltpremiere in Braunschweig: Erstmals fährt heute ein Fahrzeug automatisch im alltäglichen Stadtverkehr. Im Rahmen des Forschungsprojekts „Stadtpilot“ hat die Technische Universität Braunschweig in ihrem Kompetenzzentrum, dem Niedersächsischen Forschungszentrum Fahrzeugtechnik, ein Forschungsfahrzeug entwickelt, dass automatisch eine vorgegebene Strecke im regulären Verkehr fährt.

Autonomous Cars Come True

Self-Driving Car Test: Steve Mahan (youtube)

Autopiloten 4.10.2012

Das Wort Geisterfahrer bekommt in Kalifornien gerade eine ganz neue Bedeutung: Seit vergangener Woche dürfen auf den Highways selbststeuernde Autos cruisen. Wundern darf sich der Kalifornier also nicht, wenn er demnächst ein Auto ohne Fahrer neben sich hat. Denn der sitzt vermutlich auf der Rückbank und guckt Fernsehen, während der Autopilot lenkt.

Das neue System stammt von Google, dem es nun nicht mehr reicht, Straßen nur abzufilmen. 300 000 Meilen hätten die Gefährte schon unfallfrei zurückgelegt, teilte der Internetkonzern mit.

Autonomous Cars Come True

Freightliner Inspiration Truck Unveiled at Hoover Dam

LAS VEGAS, 5-5-2015, DTNA

First Licensed Autonomous Commercial Truck to Drive on U.S. Public Highway

In a spectacular evening ceremony at Hoover Dam, Daimler Trucks North America (DTNA) unveiled the Freightliner Inspiration Truck to several hundred international news media, trucking industry analysts and officials.

Autonomous Cars Come True

Freightliner Inspiration Truck Unveiled at Hoover Dam

LAS VEGAS, 5-5-2015, DTNA

First Licensed Autonomous Commercial Truck to Drive on U.S. Public Highway

In a spectacular evening ceremony at Hoover Dam, Daimler Trucks North America (DTNA) unveiled the Freightliner Inspiration Truck to several hundred international news media, trucking industry analysts and officials.

Daimler Inspiration Truck - So fährt der

Robo-Truck von Mercedes (youtube)

Autonomous Cars Come True

Freightliner Inspiration Truck Unveiled at Hoover Dam

LAS VEGAS, 5-5-2015, DTNA

First Licensed Autonomous Commercial Truck to Drive on U.S. Public Highway

In a spectacular evening ceremony at Hoover Dam, Daimler Trucks North America (DTNA) unveiled the Freightliner Inspiration Truck to several hundred international news media, trucking industry analysts and officials.

Daimler Inspiration Truck - So fährt der Robo-Truck von Mercedes (youtube)

The Freightliner Inspiration Truck is the first licensed autonomous commercial truck to operate on an open public highway in the United States. Developed by engineers at DTNA, it promises to unlock autonomous vehicle advancements that reduce accidents, improve fuel consumption, cut highway congestion, and safeguard the environment.

Autonomous Cars Come True

Freightliner Inspiration Truck Unveiled at Hoover Dam

LAS VEGAS, 5-5-2015, DTNA

First Licensed Autonomous Commercial Truck to Drive on U.S. Public Highway

In a spectacular evening ceremony at Hoover Dam, Daimler Trucks North America (DTNA) unveiled the Freightliner Inspiration Truck to several hundred international news media, trucking industry analysts and officials.

Daimler Inspiration Truck - So fährt der Robo-Truck von Mercedes (youtube)

The Freightliner Inspiration Truck is the first licensed autonomous commercial truck to operate on an open public highway in the United States. Developed by engineers at DTNA, it promises to unlock autonomous vehicle advancements that reduce accidents, improve fuel consumption, cut highway congestion, and safeguard the environment.

http://www.freightlinerinspiration.com/
http://www.freightlinerinspiration.com/newsroom/press/inspiration-truck-unveiled/
https://www.youtube.com/watch?v=mRkOGU3Gz9Y
https://www.youtube.com/watch?v=LL4dbq-n8Pg
https://www.youtube.com/watch?v=LJz4Ms_5AXE

How to Solve a Problem?

Algorithm

- Intuitively build a model
- Deduce a numerical algorithm
- Put it into a program
- Use it respecting the preconditions

Expert System

Neural Network

How to Solve a Problem?

Algorithm

- Intuitively build a model
- Deduce a numerical algorithm
- Put it into a program
- Use it respecting the preconditions

Expert System

- Intuitively build an model
- Formulate rules
- Apply Rules
- may solve related problems

Neural Network

How to Solve a Problem?

Algorithm

- Intuitively build a model
- Deduce a numerical algorithm
- Put it into a program
- Use it respecting the preconditions

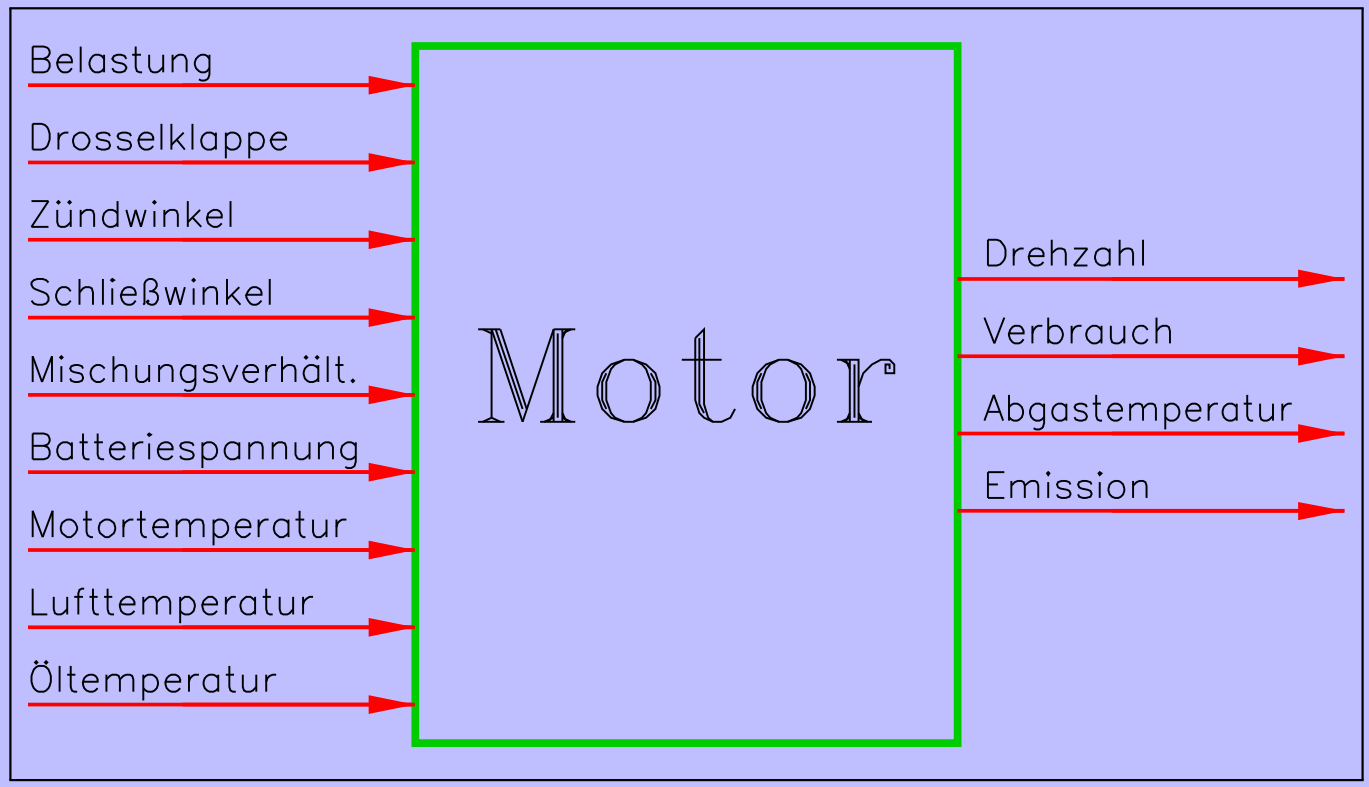
Expert System

- Intuitively build an model
- Formulate rules
- Apply Rules
- may solve related problems

Neural Network

- Intuitively build an model
- needs sampling points
- generalizes based on sampling data
- applies to related problems

Physical Performance: An Engine on a Test Bench

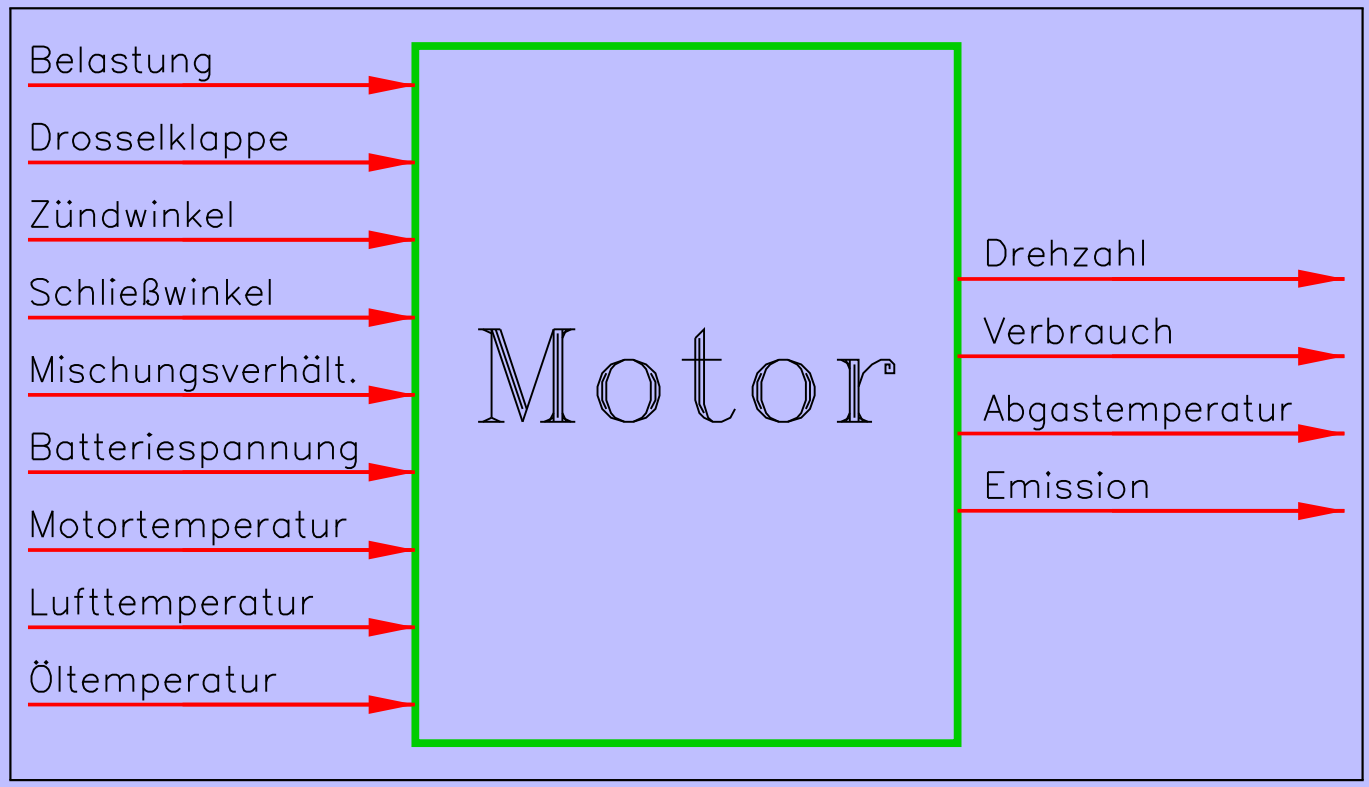


load, throttle walve, ignition angle, dwell angle, mixture, voltage, temperature of engine, air and oil

$\rightarrow$

rotational speed, consumption, temperature and amount of emission

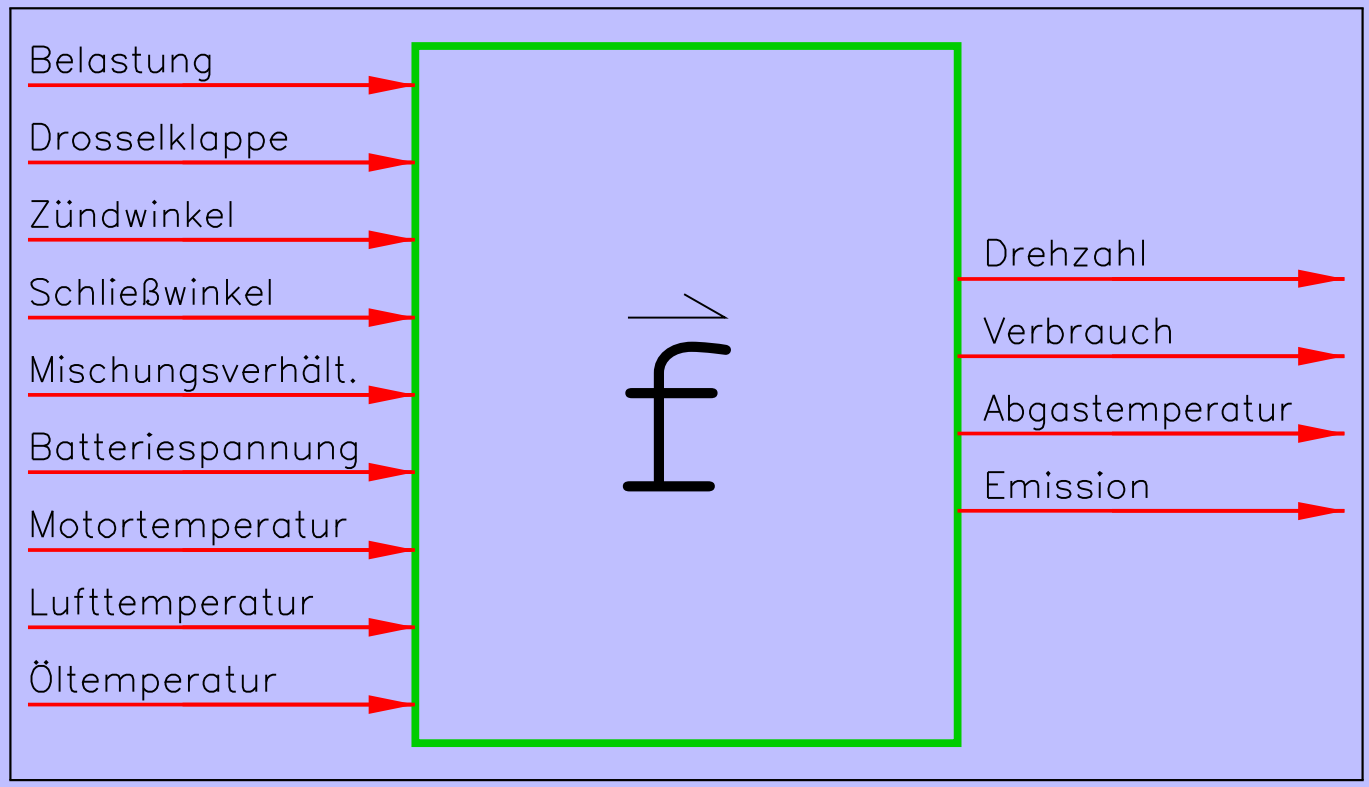
Physical Performance: An Engine on a Test Bench



Targets

- Create the optimal engine characteristic map.
- . . . also regarding start situation.
- Reduce test bench time.

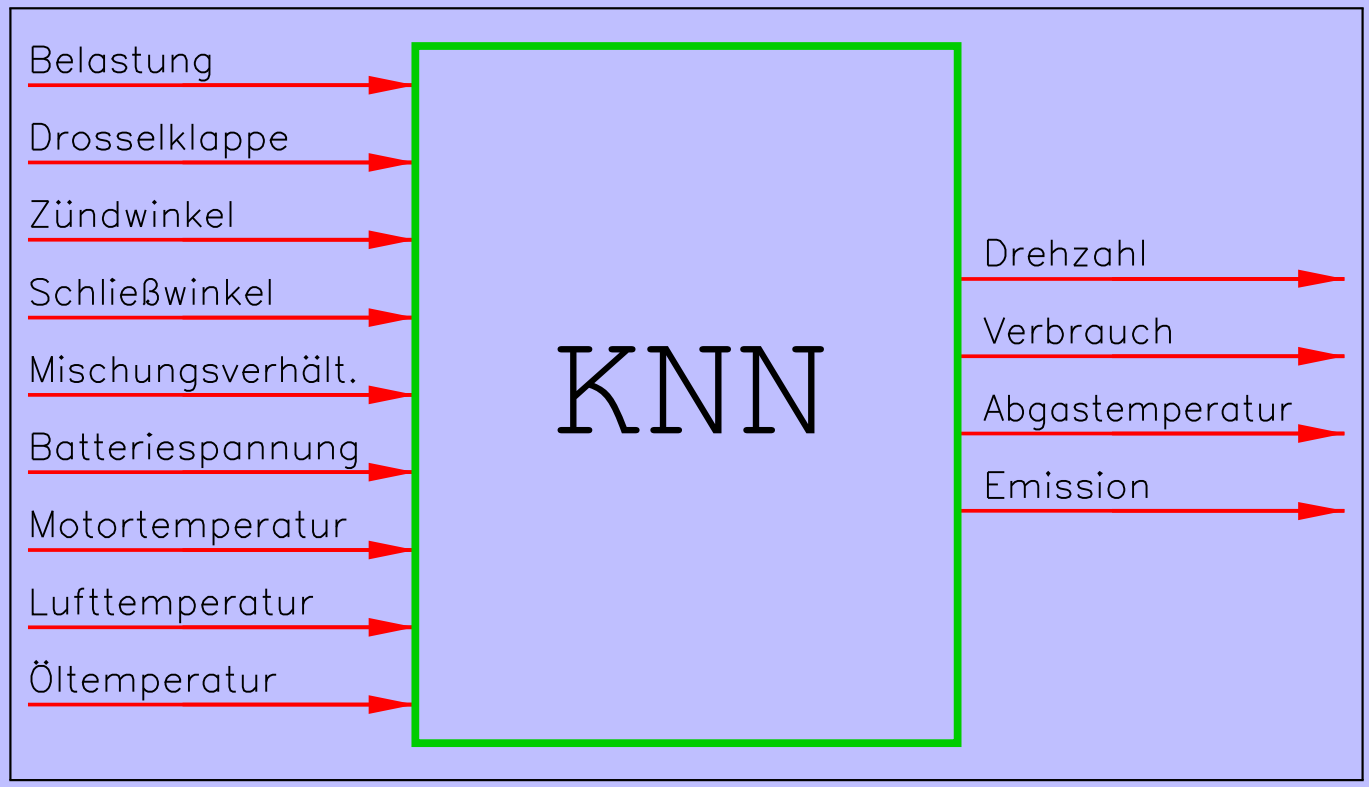
Mathematical Model of an Engine



Mathematical model: abstraction

- Look at the engine as a function
- Assumes functional dependencies (one-one)

Mathematical Model of an Engine

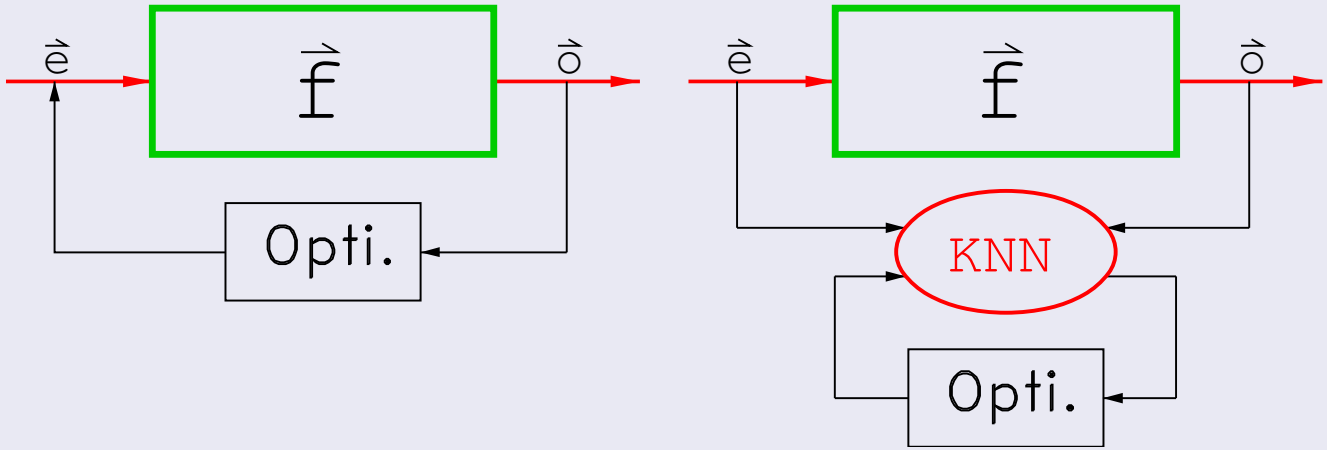


Models with artificial neural network

- Artificial neural networks should learn to “behave” like an engine.
- The knowledge must come from (measured) data.

Application: Reduce Test Bench Time

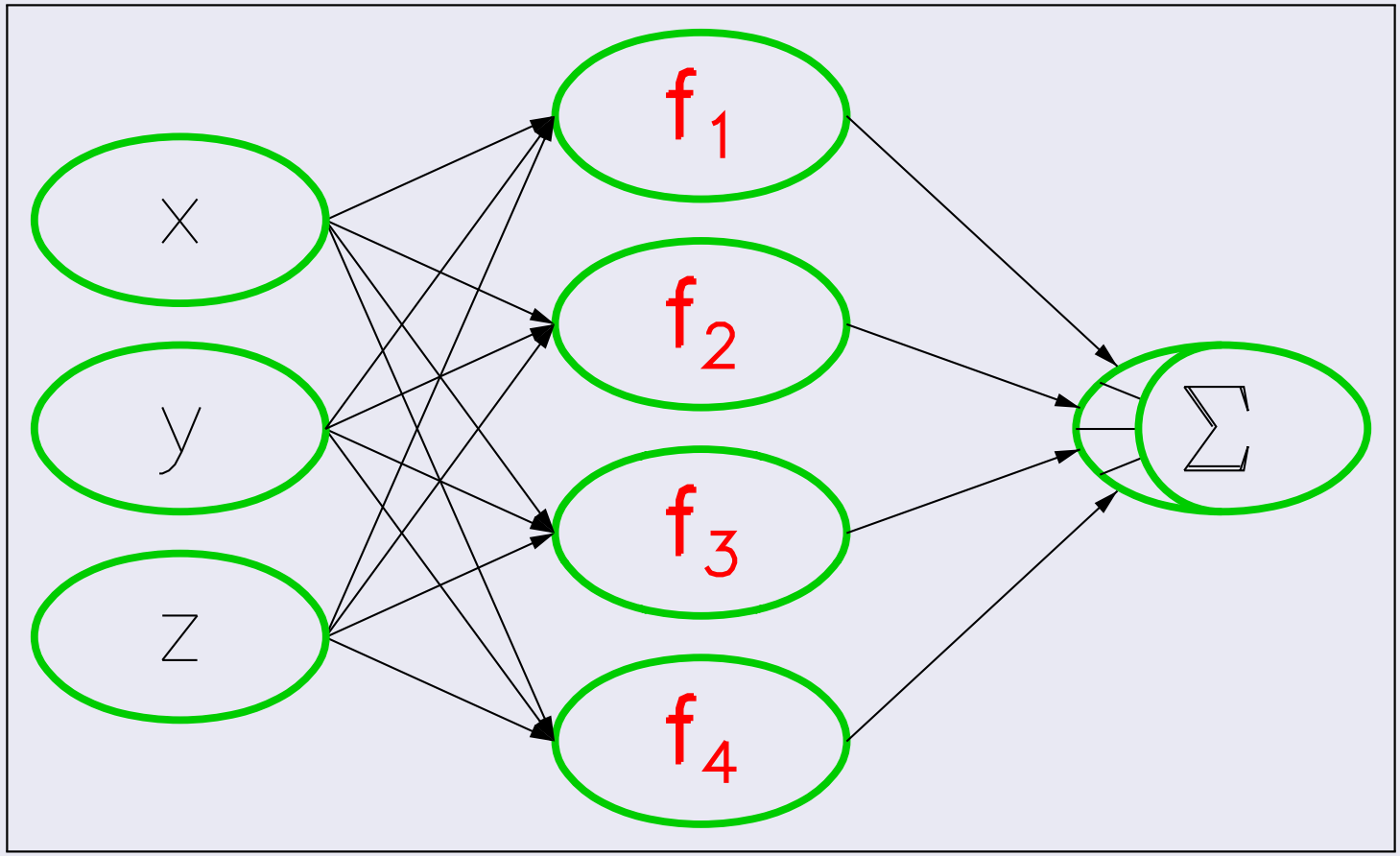
Optimization of characteristic maps using artificial neural networks



- Fill the neural networks with the "knowledge" of several engines: measured data from test bench
- New engine: extending the knowledge base with a few data from test bench
- Optimize the characteristic map using the trained neural network

R. Stricker, BMW AG, 1996

Curve Fitting (Least Square Method, Regression)



"Learning" only in the last layer

Curve Fitting (Least Sq. M., Regression): Examples

Polynomial Fitting

$$f(x) = \sum_{k=0}^{n} a_k x^k$$

Curve Fitting (Least Sq. M., Regression): Examples

Polynomial Fitting

$$f(x) = \sum_{k=0}^{n} a_k x^k$$

Fourier Fitting

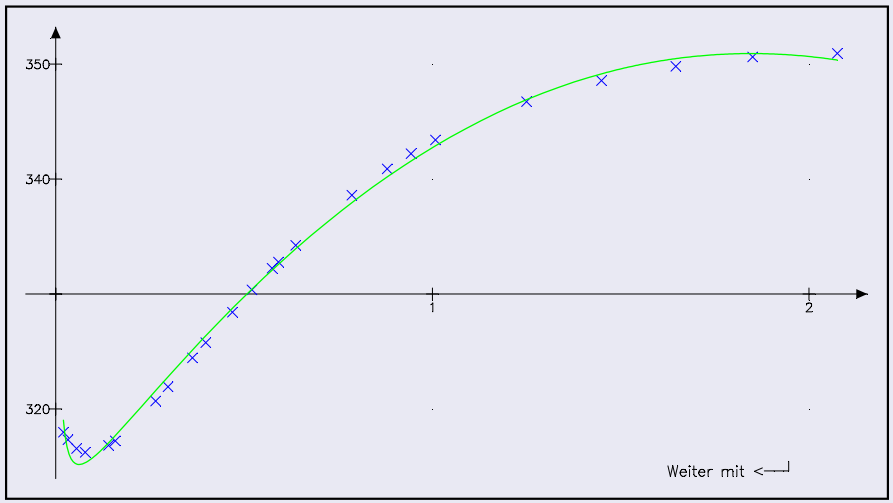
$$f(x) = a_0 + \sum_{k=1}^{n} a_k \cos(\omega k x) + a_k \sin(\omega k x)$$

Curve Fitting (Least Sq. M., Regression): Examples

Polynomial Fitting

Fourier Fitting

Stress-Strain-Diagram — BMW 1996



$$f(x) = \frac{a_{-1}}{x + .1} + a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

- provides:

$a_{-1} = 2.2677;$

$a_0 = 297.9072;$

$a_1 = 71.4932;$

$a_2 = -33.7959;$

$a_3 = 5.5016.$

Brain — Computer: a Comparison

Brain and standard computers

highly perform w.r.t. to different tasks:

Brain

- Highly parallel
- Fault tolerant
- Pattern recognition
- Generalization
- Self-organizing
- ca. 10^{11} neurons, reducing to 10^7
- Every neuron has ca. 10^4 connected neurons.

Computer

Brain — Computer: a Comparison

Brain and standard computers

highly perform w.r.t. to different tasks:

Brain

- Highly parallel
- Fault tolerant
- Pattern recognition
- Generalization
- Self-organizing
- ca. 10^{11} neurons, reducing to 10^7
- Every neuron has ca. 10^4 connected neurons.

Computer

- Precise
- Faultless storing
- Fast algorithmic calculations
- von Neumann architecture
- Nearly stand alone

Brain — Computer: a Comparison

Brain and standard computers

highly perform w.r.t. to different tasks:

Brain

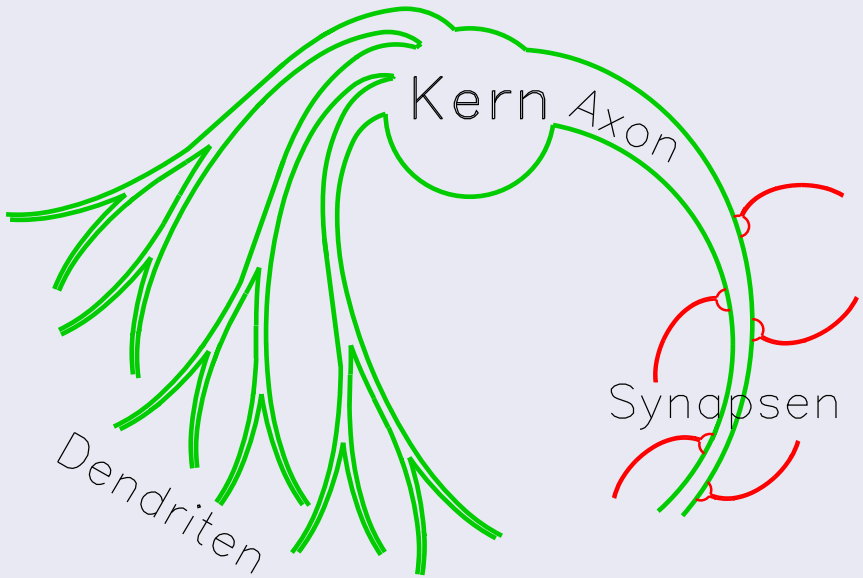
- Highly parallel
- Fault tolerant
- Pattern recognition
- Generalization
- Self-organizing
- ca. 10^{11} neurons, reducing to 10^7
- Every neuron has ca. 10^4 connected neurons.

Computer

- Precise
- Faultless storing
- Fast algorithmic calculations
- von Neumann architecture
- Nearly stand alone

The Biological Neuron

Principles of Operation



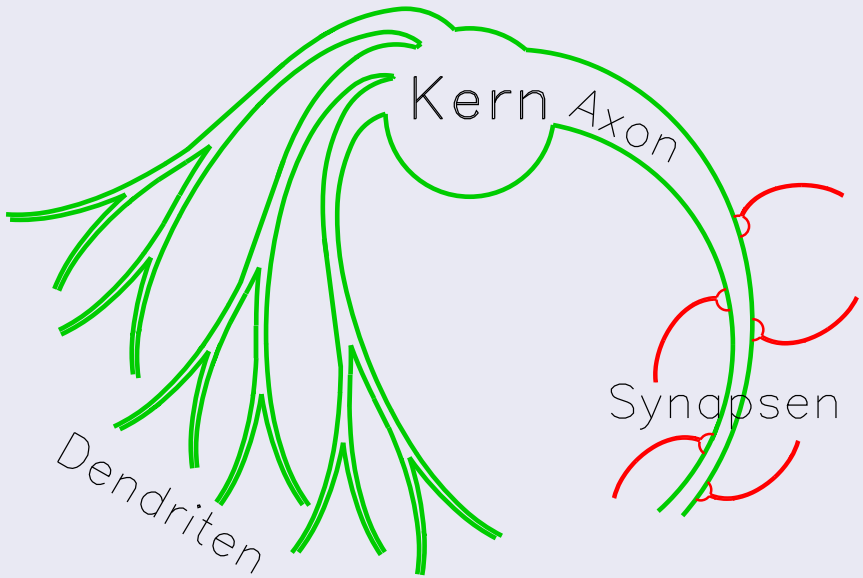
1. Impulse through the axon.
2. Synapses collect impulse. (Chemical reaction)
3. Dendrites transmit it.
4. Nucleus gets impulse.
5. Overall impulse: Excitation of the neuron.
6. Threshold target reached:

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation



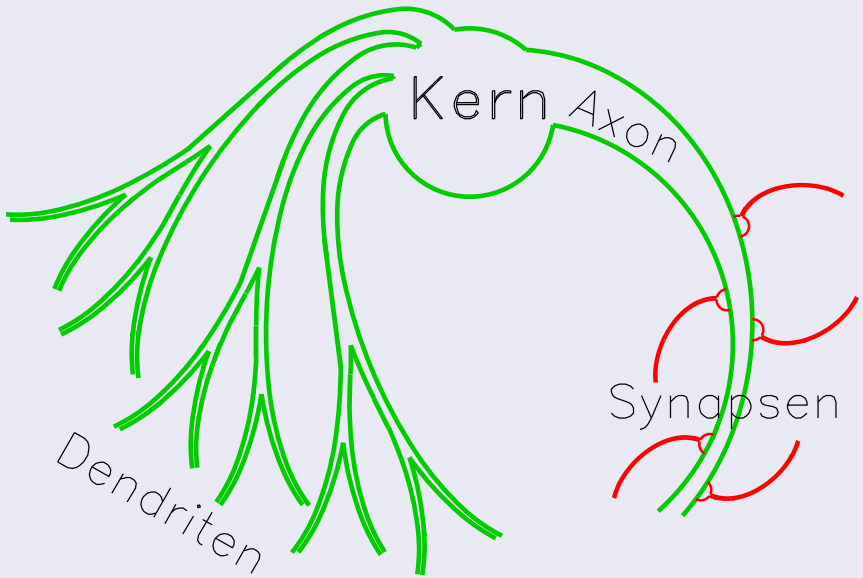
1. Impulse through the axon.
2. Synapses collect impulse. (Chemical reaction)
3. Dendrites transmit it.
4. Nucleus gets impulse.
5. Overall impulse: Excitation of the neuron.
6. Threshold target reached:

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation



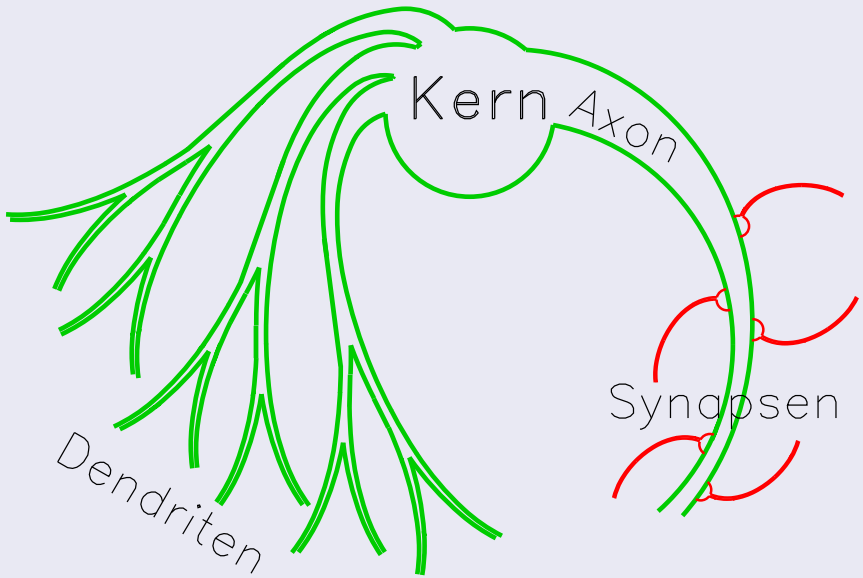
1. Impulse through the axon.
2. Synapses collect impulse. (Chemical reaction)
3. Dendrites transmit it.
4. Nucleus gets impulse.
5. Overall impulse: Excitation of the neuron.
6. Threshold target reached:

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation



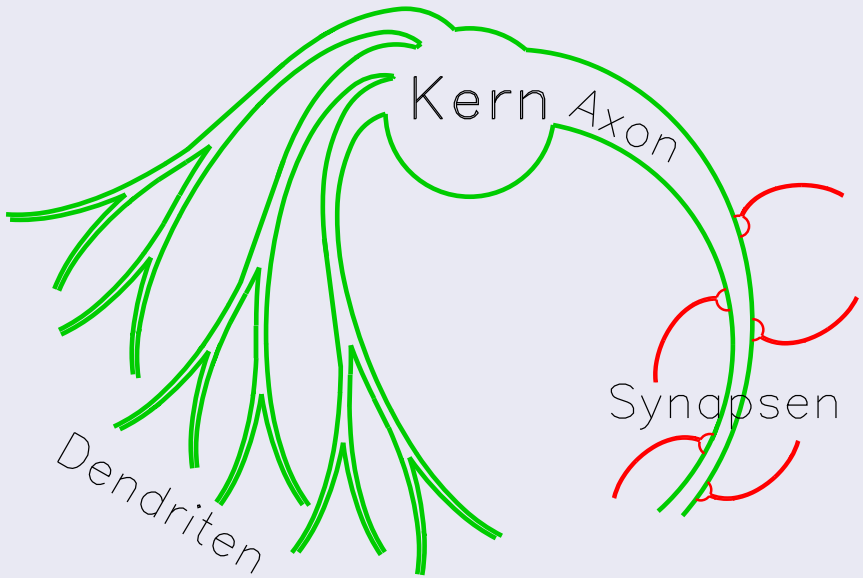
1. Impulse through the axon.
2. Synapses collect impulse.
3. Dendrites transmit it.
4. Nucleus gets impulse. (Electrical impulse)
5. Overall impulse: Excitation of the neuron.
6. Threshold target reached:

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation



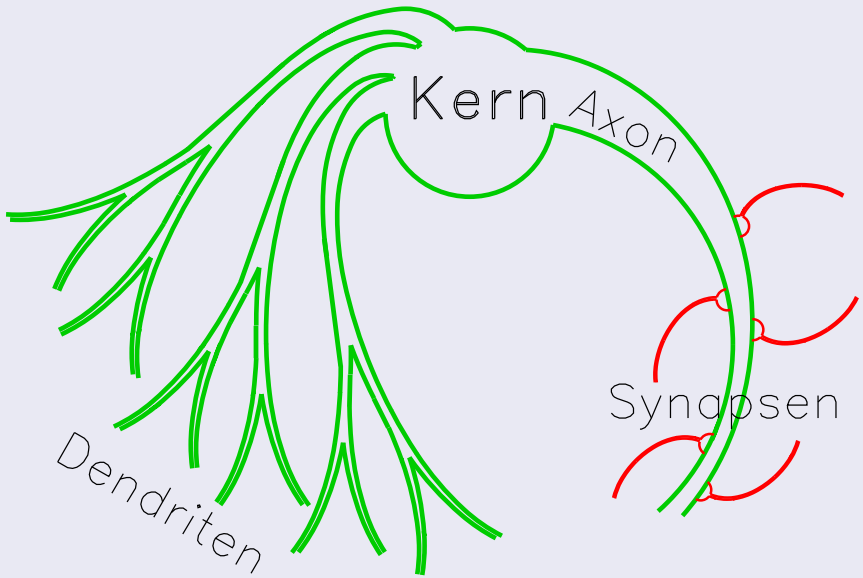
1. Impulse through the axon.
2. Synapses collect impulse.
3. Dendrites transmit it.
4. Nucleus gets impulse. (Electrical impulse)
5. Overall impulse: Excitation of the neuron.
6. Threshold target reached:

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation



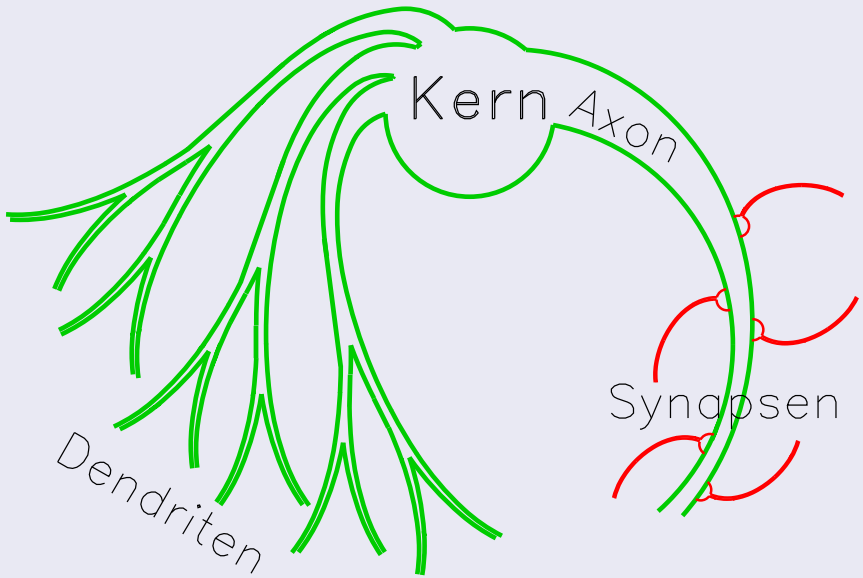
1. Impulse through the axon.
2. Synapses collect impulse.
3. Dendrites transmit it.
4. Nucleus gets impulse.
5. Overall impulse:
   Excitation of the neuron.
6. Threshold target reached:
   neuron sends impulse.

Learning:

Synapses, dendrites enhance their connection.

The Biological Neuron

Principles of Operation

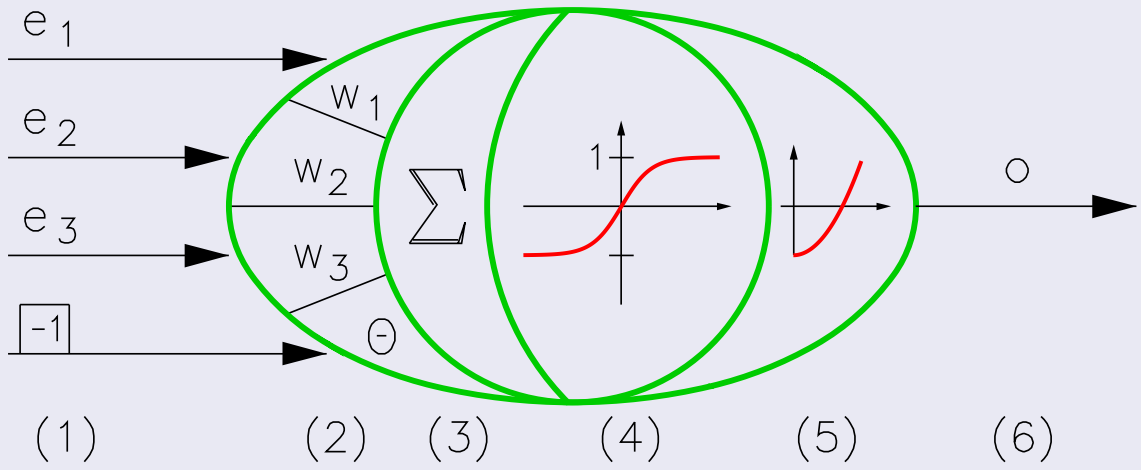


1. Impulse through the axon.
2. Synapses collect impulse.
3. Dendrites transmit it.
4. Nucleus gets impulse.
5. Overall impulse:
   Excitation of the neuron.
6. Threshold target reached:
   neuron sends impulse.

Learning:

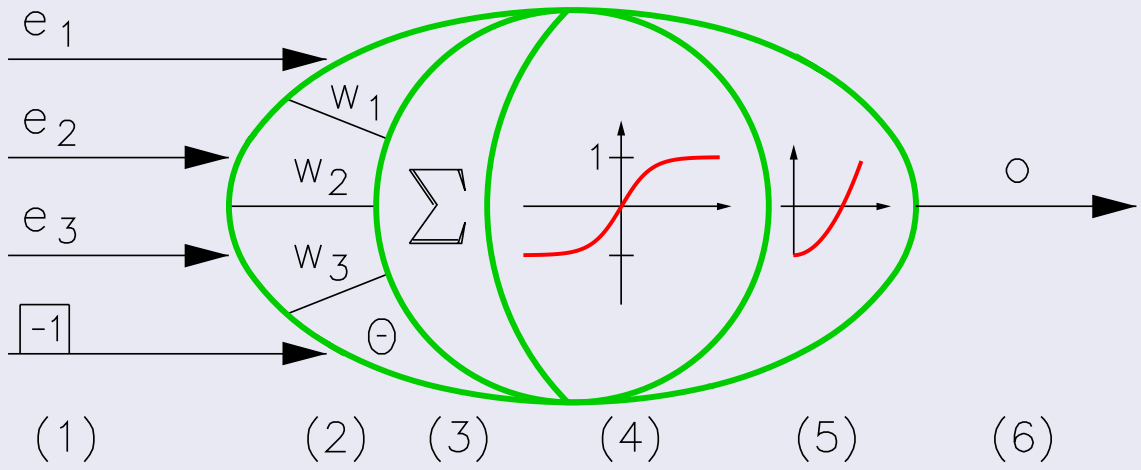
Synapses, dendrites enhance their connection.

The Artificial Neuron



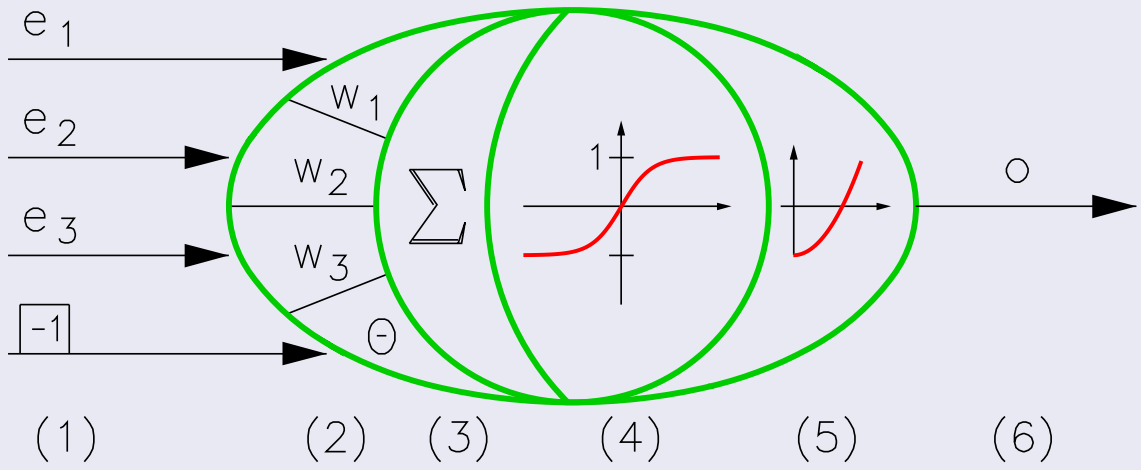
1. input (vector) $\vec{e} = (e_1, \ldots, e_n)$, -1 to be used by threshold
2. weights and threshold $\vec{w} = (w_1, \ldots, w_n)$ and θ
3. net (value), propagation $net = \langle \vec{e}, \vec{w} \rangle - \theta = \sum_{i=1}^{n} e_i w_i - \theta$
4. activation (primitive function), activity $a = a(\langle \vec{e}, \vec{w} \rangle - \theta)$
5. output function
6. otuput $o = o(a(\langle \vec{w}, \vec{e} \rangle - \theta))$

The Artificial Neuron



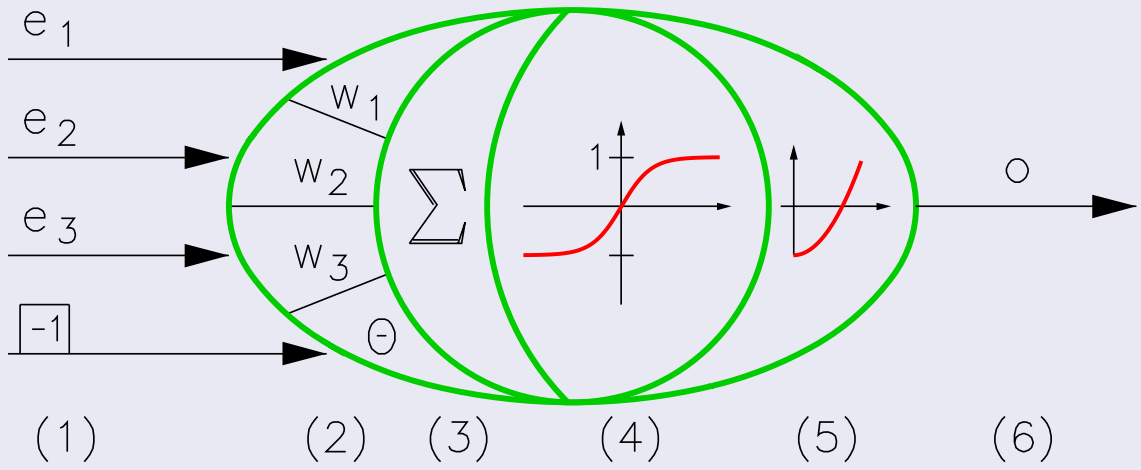
1. input (vector) $\vec{e} = (e_1, \ldots, e_n)$, -1 to be used by threshold
2. weights and threshold $\vec{w} = (w_1, \ldots, w_n)$ and θ
3. net (value), propagation $net = \langle \vec{e}, \vec{w} \rangle - \theta = \sum_{i=1}^{n} e_i w_i - \theta$
4. activation (primitive function), activity $a = a(\langle \vec{e}, \vec{w} \rangle - \theta)$
5. output function
6. otuput $o = o(a(\langle \vec{w}, \vec{e} \rangle - \theta))$

The Artificial Neuron



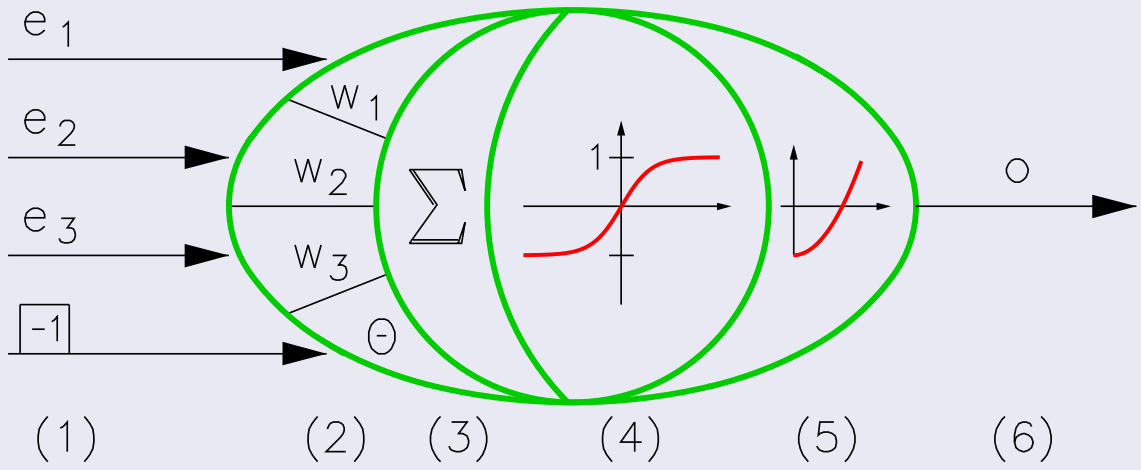
1. input (vector) $\vec{e} = (e_1, \ldots, e_n)$, -1 to be used by threshold
2. weights and threshold $\vec{w} = (w_1, \ldots, w_n)$ and θ
3. net (value), propagation $net = \langle \vec{e}, \vec{w} \rangle - \theta = \sum_{i=1}^{n} e_i w_i - \theta$
4. activation (primitive function), activity $a = a(\langle \vec{e}, \vec{w} \rangle - \theta)$
5. output function
6. otuput $o = o(a(\langle \vec{w}, \vec{e} \rangle - \theta))$

The Artificial Neuron



1. input (vector) $\vec{e} = (e_1, \ldots, e_n)$, -1 to be used by threshold
2. weights and threshold $\vec{w} = (w_1, \ldots, w_n)$ and θ
3. net (value), propagation $net = \langle \vec{e}, \vec{w} \rangle - \theta = \sum_{i=1}^{n} e_i w_i - \theta$
4. activation (primitive function), activity $a = a(\langle \vec{e}, \vec{w} \rangle - \theta)$
5. output function
6. otuput $o = o(a(\langle \vec{w}, \vec{e} \rangle - \theta))$

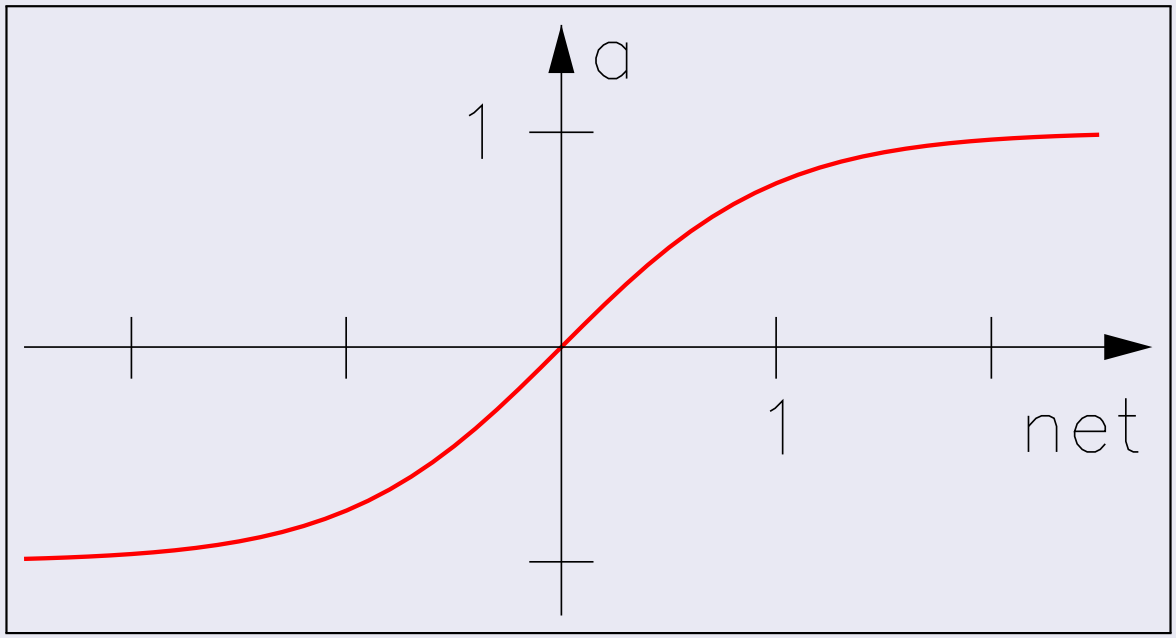
The Artificial Neuron



1. input (vector) $\vec{e} = (e_1, \ldots, e_n)$, -1 to be used by threshold
2. weights and threshold $\vec{w} = (w_1, \ldots, w_n)$ and θ
3. net (value), propagation $net = \langle \vec{e}, \vec{w} \rangle - \theta = \sum_{i=1}^{n} e_i w_i - \theta$
4. activation (primitive function), activity $a = a(\langle \vec{e}, \vec{w} \rangle - \theta)$
5. output function
6. otuput $o = o(a(\langle \vec{w}, \vec{e} \rangle - \theta))$

Sigmoidal Activation

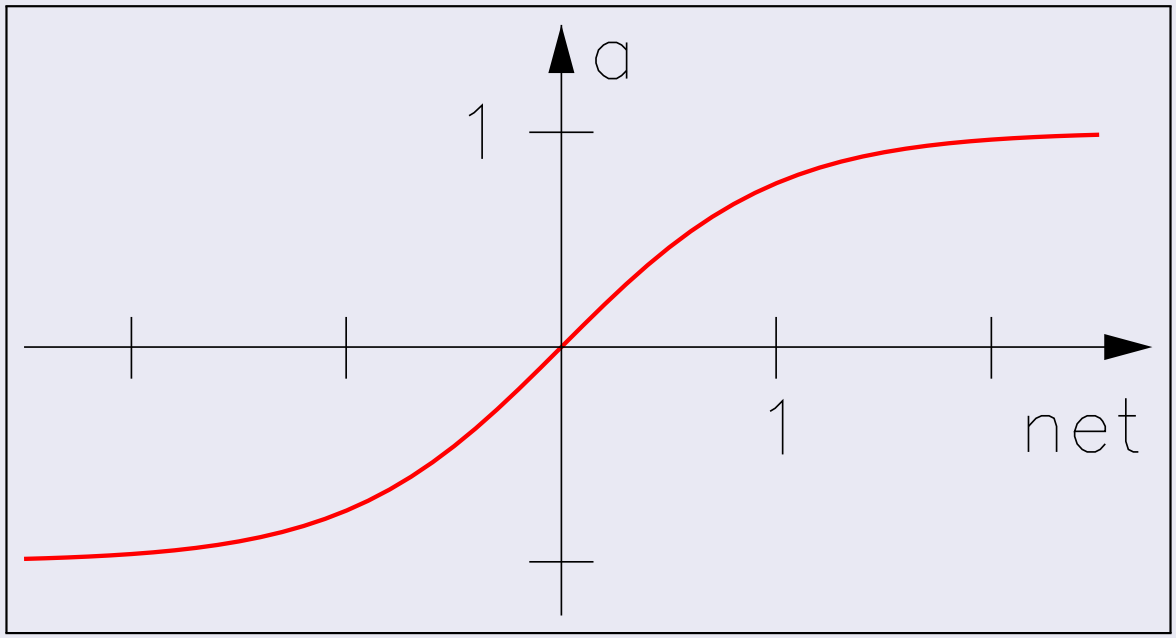
$a(x) = \tanh(gx) : \mathbb{R} \to (-1, 1)$



Derivative: $a'(x) = g(-\tanh(gx)^2) = g(1 - a(x)^2)\,; \quad a'(0) = g$

Sigmoidal Activation

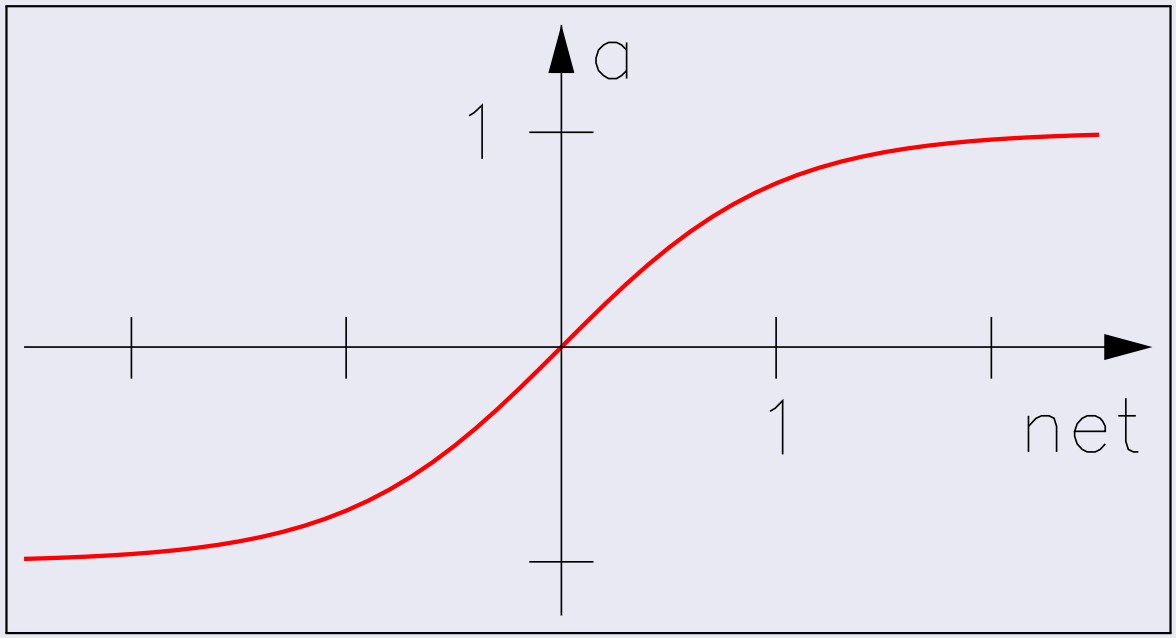
$a(x) = \tanh(gx) : \mathbb{R} \to (-1, 1)$



Derivative: $a'(x) = g(-\tanh(gx)^2) = g(1 - a(x)^2)\,; \quad a'(0) = g$

Sigmoidal Activation

$a(x) = \tanh(gx) : \mathbb{R} \to (-1, 1)$



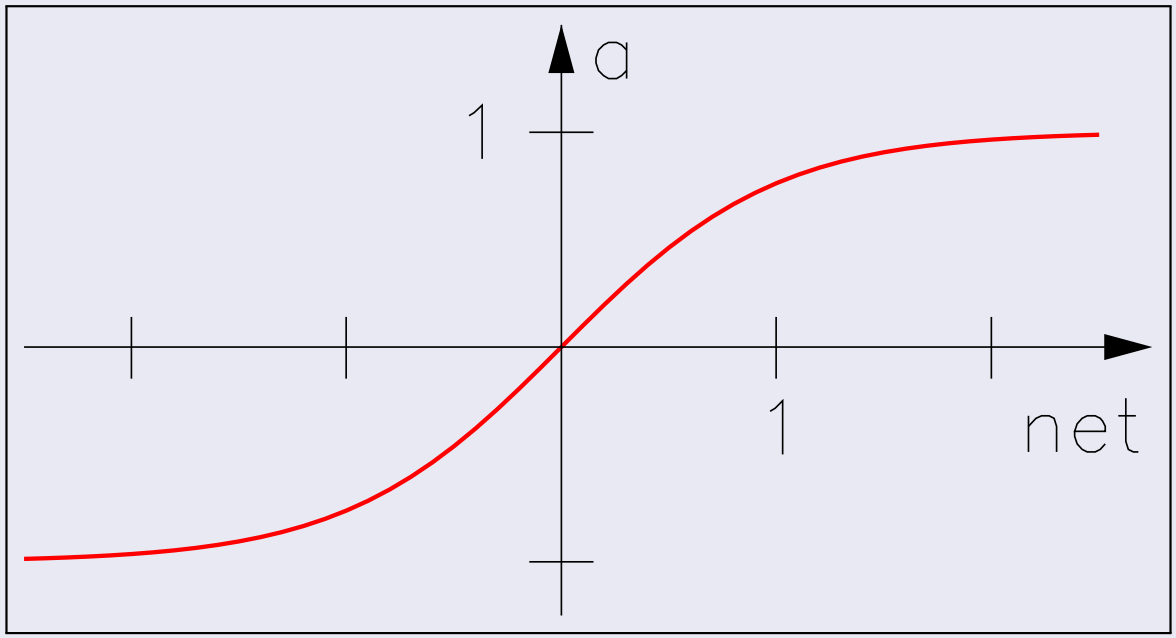
Derivative: $a'(x) = g(-\tanh(gx)^2) = g(1 - a(x)^2)\,; \quad a'(0) = g$

Alternative Activations:

1. $a(x) = \frac{1}{1+e^{-gx}} : \mathbb{R} \to (0, 1),\ a'(x) = g(1 - a(x)) \cdot a(x)\,; \quad a'(0) = \frac{g}{4}$
2. Piecewise parabola, easy implementation in hard ware (Carmen Stumm)

Sigmoidal Activation

$a(x) = \tanh(gx) : \mathbb{R} \to (-1, 1)$



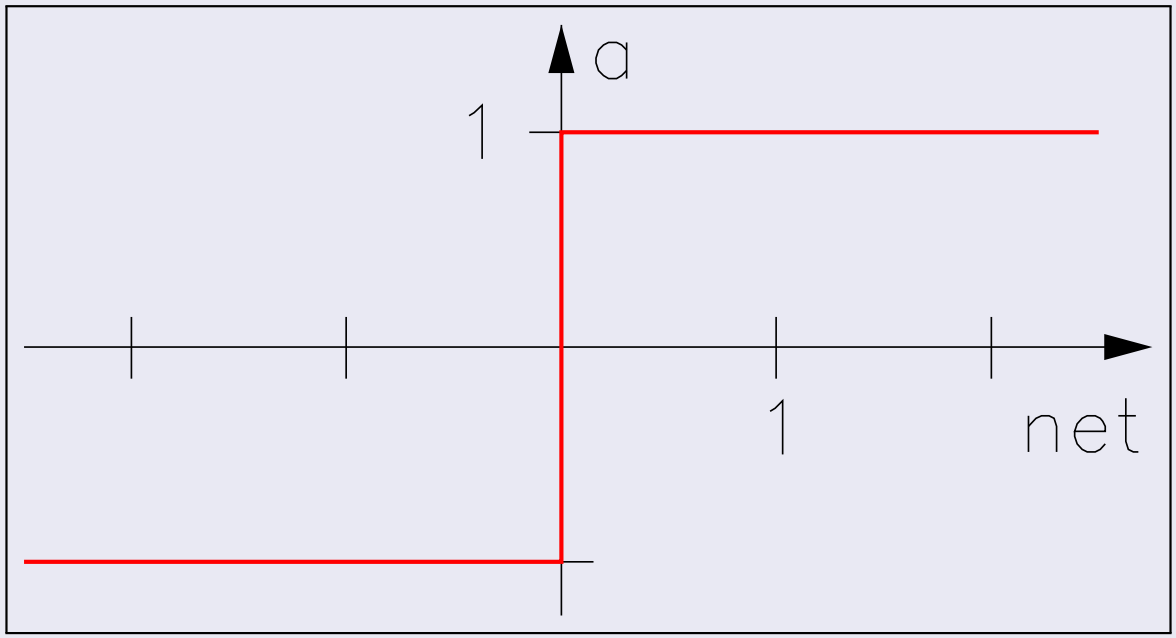
Derivative: $a'(x) = g(-\tanh(gx)^2) = g(1 - a(x)^2)\,; \quad a'(0) = g$

Alternative Activations:

1. $a(x) = \frac{1}{1+e^{-gx}} : \mathbb{R} \to (0, 1),\ a'(x) = g(1 - a(x)) \cdot a(x)\,; \quad a'(0) = \frac{g}{4}$
2. Piecewise parabola, easy implementation in hard ware (Carmen Stumm)

Bipolar and Binary Threshold Function

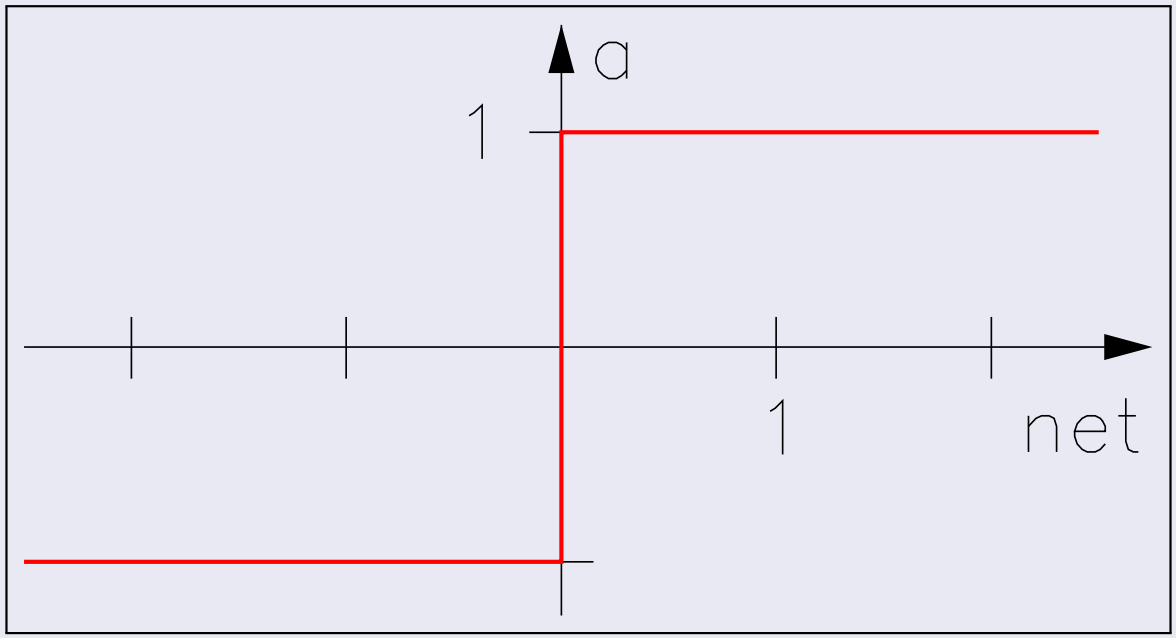
$a(x) = sign(x) : \mathbb{R} \rightarrow [-1, 1]$



Activity: $a(x) = \text{sign}(\langle \vec{e}, \vec{w} \rangle - \theta) = 2(\langle \vec{e}, \vec{w} \rangle - \theta \geq 0) - 1$

Bipolar and Binary Threshold Function

$a(x) = sign(x) : \mathbb{R} \rightarrow [-1, 1]$



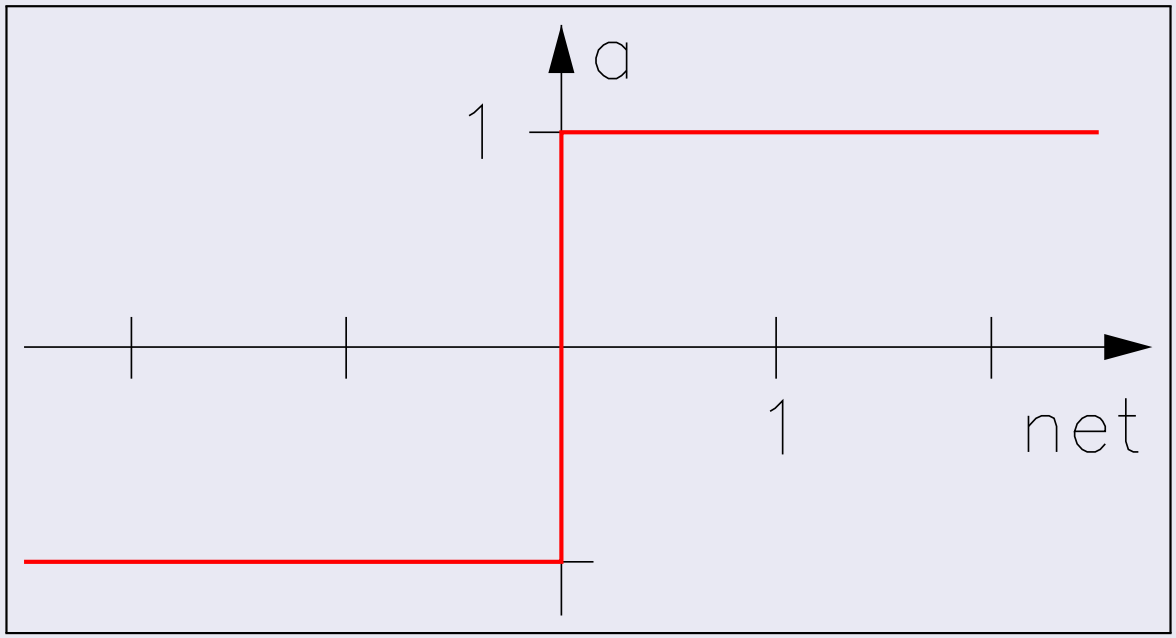
Activity: $a(x) = \text{sign}(\langle \vec{e}, \vec{w} \rangle - \theta) = 2(\langle \vec{e}, \vec{w} \rangle - \theta \geq 0) - 1$

Alternative Activities:

$a(x) = (x \geq 0) = \frac{1+\text{sign}(x)}{2} = \langle x - 0 \rangle^0 : \mathbb{R} \rightarrow [0, 1]$

Bipolar and Binary Threshold Function

$a(x) = sign(x) : \mathbb{R} \rightarrow [-1, 1]$



Activity: $a(x) = \text{sign}(\langle \vec{e}, \vec{w} \rangle - \theta) = 2(\langle \vec{e}, \vec{w} \rangle - \theta \geq 0) - 1$

Linear Separation

Threshold neurons linearly separate input data, logically combined threshold neurons define a simplex.

Linearly separable Sets

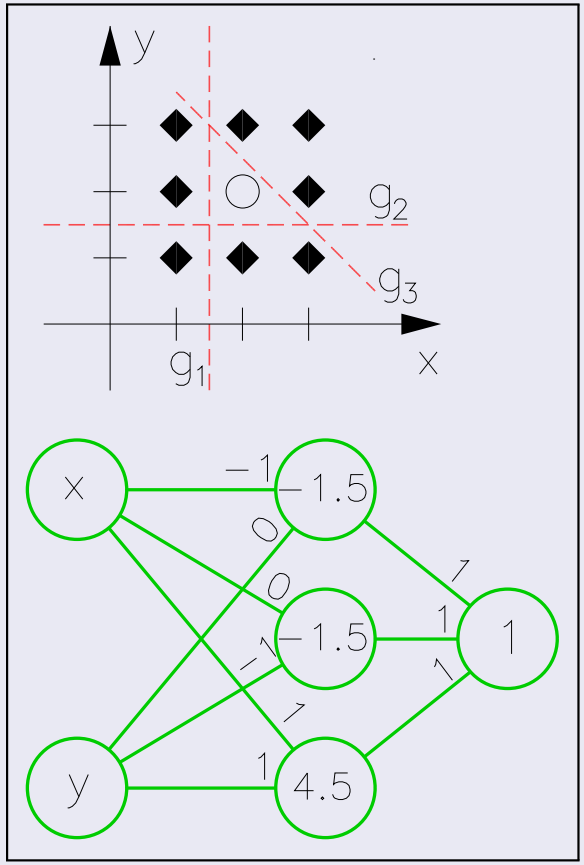
Hidden neurons separate linearly

$$\begin{pmatrix} x \\ y \end{pmatrix}\begin{pmatrix} -1 \\ 0 \end{pmatrix} \geq \begin{pmatrix} 1.5 \\ 0 \end{pmatrix}\begin{pmatrix} -1 \\ 0 \end{pmatrix} = -1.5$$

$$\begin{pmatrix} x \\ y \end{pmatrix}\begin{pmatrix} 0 \\ -1 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 1.5 \end{pmatrix}\begin{pmatrix} 0 \\ -1 \end{pmatrix} = -1.5$$

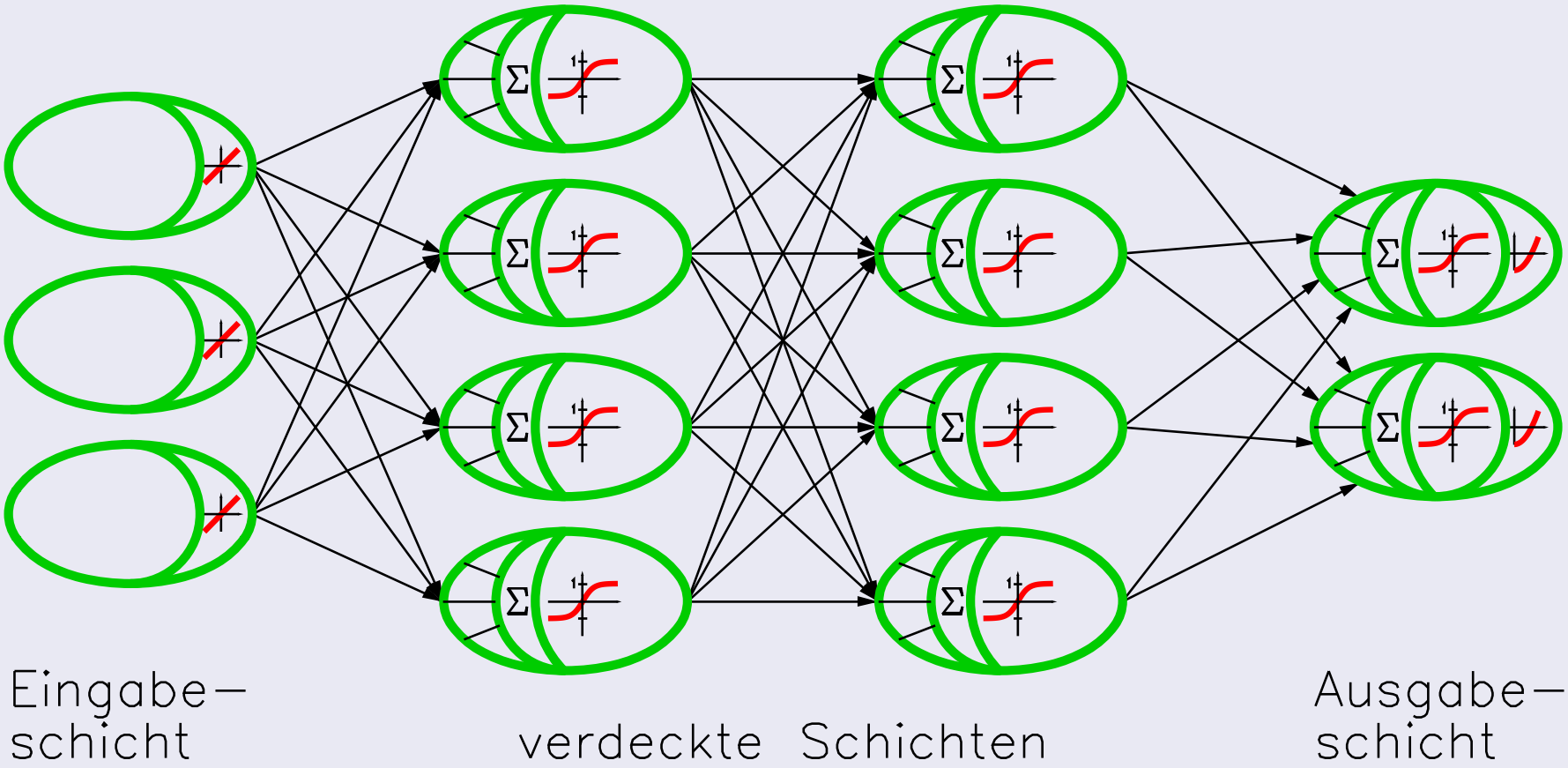
$$\begin{pmatrix} x \\ y \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \geq \begin{pmatrix} 3 \\ 1.5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 4.5$$

The output neuron gathers these results using the logical OR-function.
A positive answer ($o = 1$) signals that the element belongs to the outer region (positive region).



Multi-layered Feed Forward Networks

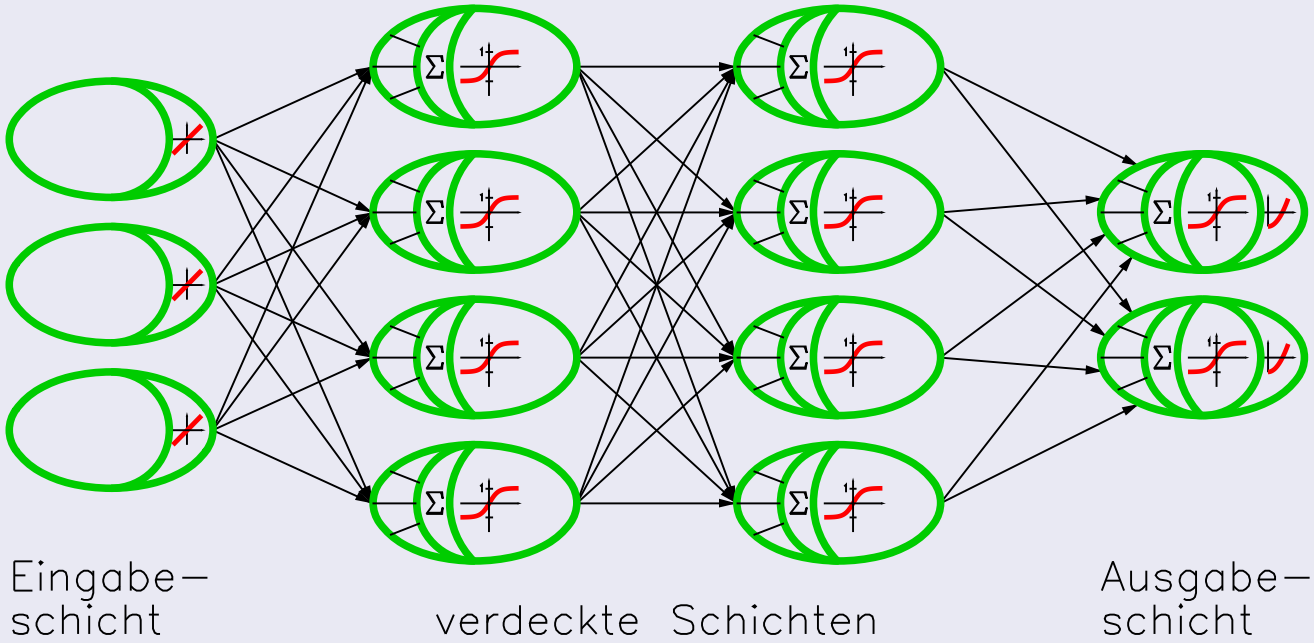
Feed forward network with topology 3-4-4-2



Learning: Change weights and threshold until the result satisfies.

Multi-layered Feed Forward Networks

Feed forward network with topology 3-4-4-2




```
r←s Bpforw ein;anzs;aus;is;net
anzs←↑⍴bpan◇aus←net←bpan⍴¨0◇aus[1]←⊂⍉ein⊥¨⊂⌽bpte          ⍝ Eing. trans.
is←1
DO4:→(anzs<is←is+1)/UNDO4          ⍝ Schleife über Schichten: Ausgaben
aus[is]←⊂1÷1+*-bpap×⊃net[is]←⊂(is⊃bpbi)+[1](r⊃bpgw)+.×(r←is-1)⊃aus◇→DO4
UNDO4:r←⍉(anzs⊃aus)⊥¨⊂⌽bpta
```

Topology of a Feed-forward Network

Theorem (Kolmogorow, 1957)

Every vector-valued function $f : [0, 1]^n \to \mathbb{R}^m$ *can be written as a 3-layer feed-forward network with* n *input neurons,* $2n + 1$ *hidden neurons and* m *output neurons. The activation functions depend on* f *and* n.

Remark

1. *The proof shows the existence in a non-constructive way.*
2. *It does not give the activation functions.*
3. *The theorem has no direct practical impact.*

Topology of a Feed-forward Network

Theorem (Kolmogorow, 1957)

Every vector-valued function $f : [0, 1]^n \to \mathbb{R}^m$ *can be written as a 3-layer feed-forward network with* n *input neurons,* $2n + 1$ *hidden neurons and* m *output neurons. The activation functions depend on* f *and* n.

Zusatz

Für die stetige Funktion $f : [-1, 1]^n \to [-1, 1]$ *gibt es Funktionen* g *und* g_i $(i = 1, \ldots, 2n + 1)$ *in einem Argument und Konstanten* λ_j $(j = 1, \ldots, n)$ *mit*

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{2n+1} g\left(\sum_{j=1}^{n} \lambda_j g_i(x_j)\right) .$$

Topology of a Feed-forward Network

Theorem (Kolmogorow, 1957)

Every vector-valued function $f : [0, 1]^n \to \mathbb{R}^m$ *can be written as a 3-layer feed-forward network with* n *input neurons,* $2n + 1$ *hidden neurons and* m *output neurons. The activation functions depend on* f *and* n.

Satz (Annäherung durch Netze)

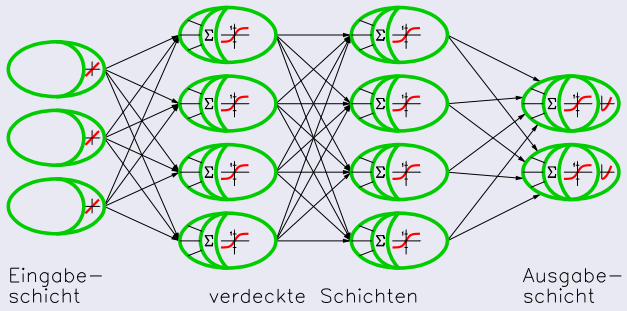
Jede Funktion kann durch Netze mit einer verdeckten Schicht angenähert werden.

Multi-layered Feed Forward Networks

Input layer

- Continuous input:
  Linear transformation into [-1; 1]
- Discrete input:
  One neuron per value, transformed onto -1, 1

Multi-layer network

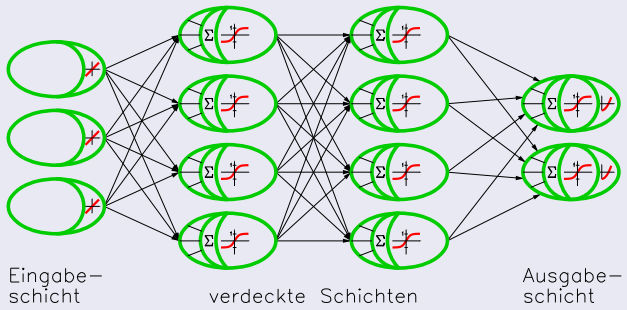


Multi-layered Feed Forward Networks

Input layer

- Continuous input:
  Linear transformation into [-1; 1]
- Discrete input:
  One neuron per value, transformed onto -1, 1

Multi-layer network



Output layer using a tangential activity function

- Target activities should be equally distributed in the interval $[-0.6, 0.6]$!
- The inverse of the output function could be:

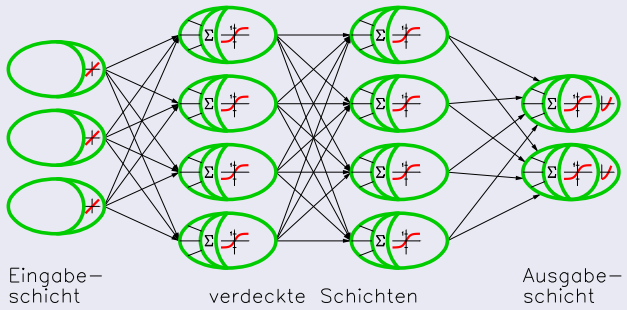
$$f(x) = \left\{ \begin{array}{ccc} [m,\ M] & \rightarrow & [-0.6,\ 0.6] \\ x & \mapsto & -0.6 + 1.2 \left(\frac{x-m}{M-m} \right)^s \ ; \quad s > 0 \end{array} \right\}$$

Multi-layered Feed Forward Networks

Input layer

- Continuous input:
  Linear transformation into [-1; 1]
- Discrete input:
  One neuron per value, transformed onto -1, 1

Multi-layer network



Output layer using a logarithmic activity function

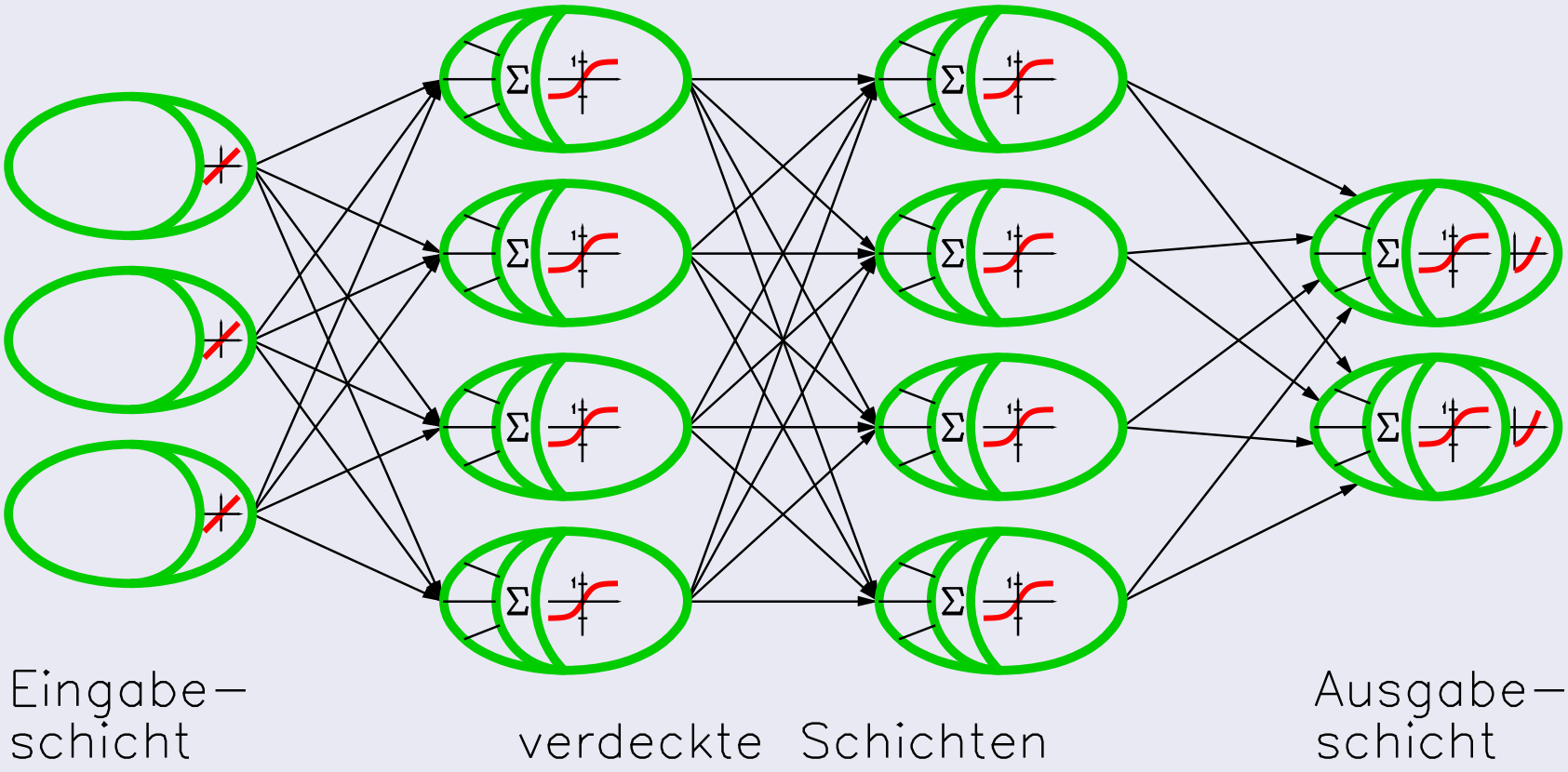
- Target activities should be equally distributed in the interval [0.2, 0.8]!
- The inverse of the output function could be:

$$f(x) = \left\{ \begin{array}{ccc} [m,\ M] & \rightarrow & [0.2,\ 0.8] \\ x & \mapsto & 0.2 + 0.6 \left(\frac{x-m}{M-m} \right)^s \ ; \quad s > 0 \end{array} \right\}$$

Learning in multi-layered networks

Target

Change the weights and thresholds in such a way that the errors in the training data get small.



Learning in multi-layered networks

Target

Change the weights and thresholds in such a way that the errors in the training data get small.

Calculations

error: $$E(\vec{w}) = \frac{1}{2} \sum_{i=1}^{n} \|\vec{z}_i - \vec{o}_i(w)\|^2$$

gradient: $$\overrightarrow{\text{grad}}_w E(\vec{w}) = \left(\frac{\partial E(\vec{w})}{\partial w_1}, \frac{\partial E(\vec{w})}{\partial w_2}, \ldots, \frac{\partial E(\vec{w})}{\partial w_3} \right)$$

Learning in multi-layered networks

Target

Change the weights and thresholds in such a way that the errors in the training data get small.

Calculations

error: $E(\vec{w}) = \frac{1}{2} \sum_{i=1}^{n} \|\vec{z}_i - \vec{o}_i(w)\|^2$

gradient: $\overrightarrow{\text{grad}}_w E(\vec{w}) = \left(\frac{\partial E(\vec{w})}{\partial w_1}, \frac{\partial E(\vec{w})}{\partial w_2}, \ldots, \frac{\partial E(\vec{w})}{\partial w_3} \right)$

Delta-Rule (Gradient descent)

$\Delta \vec{w}^{(t)} = -\sigma \overrightarrow{\text{grad}}_w E(\vec{w}); \qquad \vec{w}^{(t)} = \vec{w}^{(t-1)} + \Delta \vec{w}^{(t)} + \mu \Delta \vec{w}^{(t-1)}$

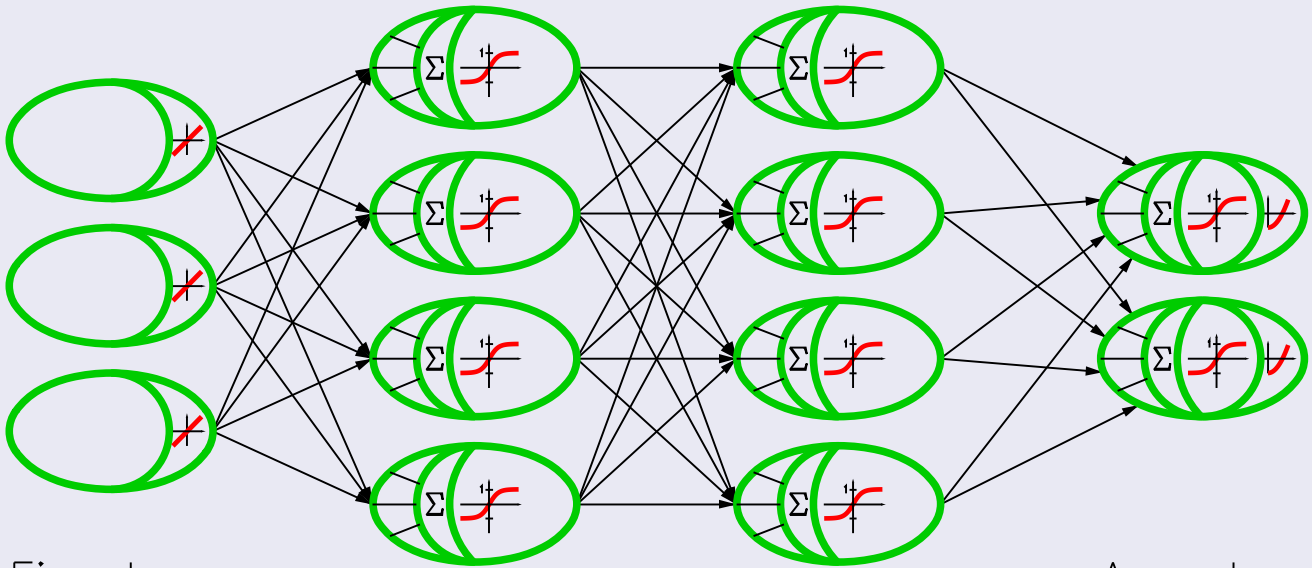
σ decreasing, z.B. von 0.9 auf 0.1, μ increasing, z.B. $\mu = 1 - \sigma$.

Error Back Propagation

Step-by-step error back propagation using the net error δ_i:

$$\vec{\delta}_i := \frac{\partial E}{\partial \vec{n}_i} = \frac{\partial E}{\partial \vec{n}_{i+1}} \cdot \frac{\partial \vec{n}_{i+1}}{\partial \vec{o}_i} \cdot \frac{\partial \vec{o}_i}{\partial \vec{n}_i} = \vec{\delta}_{i+1} \cdot W_{i+1} \cdot A(\vec{n}_i)$$

$$\frac{\partial E}{\partial W_{i,rs}} = \frac{\partial E}{\partial \vec{n}_i} \cdot \frac{\partial \vec{n}_i}{\partial W_{i,rs}} = \vec{\delta}_i \cdot o_{i-1,s} \hat{e}_r = \delta_{i,r} o_{i-1,s}$$



Error Back Propagation

Step-by-step error back propagation using the net error δ_i:

$$\vec{\delta}_i := \frac{\partial E}{\partial \vec{n}_i} = \frac{\partial E}{\partial \vec{n}_{i+1}} \cdot \frac{\partial \vec{n}_{i+1}}{\partial \vec{o}_i} \cdot \frac{\partial \vec{o}_i}{\partial \vec{n}_i} = \vec{\delta}_{i+1} \cdot W_{i+1} \cdot A(\vec{n}_i)$$

$$\frac{\partial E}{\partial W_{i,rs}} = \frac{\partial E}{\partial \vec{n}_i} \cdot \frac{\partial \vec{n}_i}{\partial W_{i,rs}} = \vec{\delta}_i \cdot o_{i-1,s} \hat{e}_r = \delta_{i,r} o_{i-1,s}$$

```
r←ziel Bpback aus;anzs;dgwa;err;is;lr
(aus lr)←aus
err←bpan⍴¨0
dgwa←dgw                                          ⍝ dgw global
is←anzs←↑⍴bpan                                    ⍝ Anzahl Schichten
r←anzs⊃aus                                        ⍝ Fehler letzte Schicht
err[anzs]←⊂-2×bpap×(r×1-r)×ziel-r                 ⍝ Nettofehler
DO:→(1>is←is-1)/UNDO                              ⍝ B.P. über alle Sch.
dgw[is]←⊂(-(1+is)⊃err)∘.×r←is⊃aus                  ⍝ ΔGewicht je Schicht
⍎(is>1)/'err[is]←⊂bpap×(r×1-r)×(⍉is⊃bpgw)+.×⊃err[1+is]'   ⍝ Nettofehler
→DO
UNDO:bpgw←bpgw+lr×dgw+(1-lr)×dgwa                 ⍝ Gewichte ändern
bpbi←bpbi+(dbi←-lr×err)+(1-lr)×dbi                ⍝ Bias ändern
r←anzs⊃err                                        ⍝ Fehler in letzter Sch.
```

Learning in multi-layered networks

Levenberg-Marquardt-Method

$$E(\vec{w}) \quad = \quad \frac{1}{2} \left\langle \vec{f}(\vec{w}), \vec{f}(\vec{w}) \right\rangle \qquad \text{mit} \qquad \vec{f}(\vec{w}) = \vec{z} - \vec{o}(\vec{w})$$

Learning in multi-layered networks

Levenberg-Marquardt-Method

$$E(\vec{w}) = \frac{1}{2}\left\langle \vec{f}(\vec{w}), \vec{f}(\vec{w}) \right\rangle \qquad \text{mit} \qquad \vec{f}(\vec{w}) = \vec{z} - \vec{o}(\vec{w})$$

$$\vec{0} = E'(\vec{w}) = \overrightarrow{\operatorname{grad}}\, E(\vec{w}) = f'^{T}(\vec{w})\vec{f}(\vec{w})$$

Learning in multi-layered networks

Levenberg-Marquardt-Method

$$
\begin{aligned}
E(\vec{w}) &= \frac{1}{2}\left\langle \vec{f}(\vec{w}),\, \vec{f}(\vec{w}) \right\rangle \qquad \text{mit} \qquad \vec{f}(\vec{w}) = \vec{z} - \vec{o}(\vec{w}) \\
\vec{0} &= E'(\vec{w}) = \overrightarrow{\operatorname{grad}}\, E(\vec{w}) = f'^{T}(\vec{w})\vec{f}(\vec{w}) \\
E''(\vec{w}) &= \left(f'^{T}(\vec{w})\vec{f}(\vec{w})\right)' = f''^{T}(\vec{w})\vec{f}(\vec{w}) + f'^{T}(\vec{w})f'(\vec{w}) \\
&= f'^{T}(\vec{w})f'(\vec{w}) \qquad \text{für} \qquad f''^{T}(\vec{w}) \qquad \text{small!}
\end{aligned}
$$

Learning in multi-layered networks

Levenberg-Marquardt-Method

$$
\begin{aligned}
E(\vec{w}) &= \frac{1}{2}\left\langle \vec{f}(\vec{w}),\, \vec{f}(\vec{w}) \right\rangle \qquad \text{mit} \qquad \vec{f}(\vec{w}) = \vec{z} - \vec{o}(\vec{w}) \\
\vec{0} &= E'(\vec{w}) = \overrightarrow{\operatorname{grad}}\, E(\vec{w}) = f'^{T}(\vec{w})\vec{f}(\vec{w}) \\
E''(\vec{w}) &= \left(f'^{T}(\vec{w})\vec{f}(\vec{w})\right)' = f''^{T}(\vec{w})\vec{f}(\vec{w}) + f'^{T}(\vec{w})f'(\vec{w}) \\
&\approx f'^{T}(\vec{w})f'(\vec{w}) \qquad \text{für} \qquad f''^{T}(\vec{w}) \qquad \text{small!} \\
\vec{w}_{k+1} &= \vec{w}_k - E''(\vec{w})^{-1}E'(\vec{w}) \\
\Delta\vec{w} &= -\left(f'^{T}(\vec{w})f'(\vec{w})\right)^{-1} f'^{T}(\vec{w})\vec{f}(\vec{w})
\end{aligned}
$$

Learning in multi-layered networks

Levenberg-Marquardt-Method

$$
\begin{aligned}
E(\vec{w}) &= \frac{1}{2}\left\langle \vec{f}(\vec{w}), \vec{f}(\vec{w}) \right\rangle \qquad \text{mit} \qquad \vec{f}(\vec{w}) = \vec{z} - \vec{o}(\vec{w}) \\
\vec{0} &= E'(\vec{w}) = \overrightarrow{\operatorname{grad}}\, E(\vec{w}) = f'^{T}(\vec{w})\vec{f}(\vec{w}) \\
E''(\vec{w}) &= f'^{T}(\vec{w})f'(\vec{w}) \qquad \text{für} \qquad f''^{T}(\vec{w}) \qquad \text{small!} \\
\vec{w}_{k+1} &= \vec{w}_k - E''(\vec{w})^{-1}E'(\vec{w}) \\
\Delta\vec{w} &= -\left(f'^{T}(\vec{w})f'(\vec{w})\right)^{-1} f'^{T}(\vec{w})\vec{f}(\vec{w})
\end{aligned}
$$

System of linear equations to be solved:

$$f'^{T}(\vec{w})f'(\vec{w})\Delta\vec{w} = -f'^{T}(\vec{w})\vec{f}(\vec{w})$$

Evaluation of a trained network

Error in training data

- maximal error
- mean error
- standard deviation

Evaluation of a trained network

Error in training data

- maximal error
- mean error
- standard deviation

Error in testing data

- 20%-40% of available data
- maximal and mean error
- standard deviation

Evaluation of a trained network

Error in training data

- maximal error
- mean error
- standard deviation

Error in testing data

- 20%-40% of available data
- maximal and mean error
- standard deviation

Insider

- Evaluation of forecasts

Evaluation of a trained network

Error in training data

- maximal error
- mean error
- standard deviation

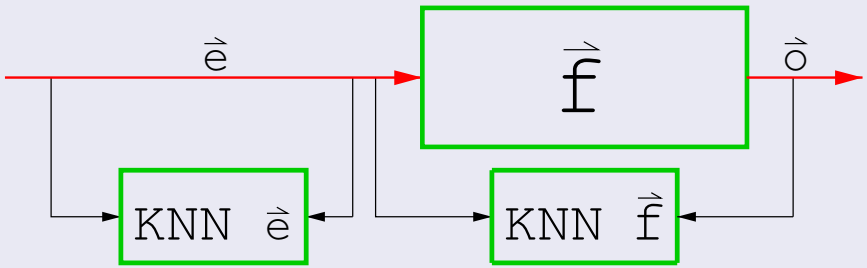
Insider

- Evaluation of forecasts

Error in testing data

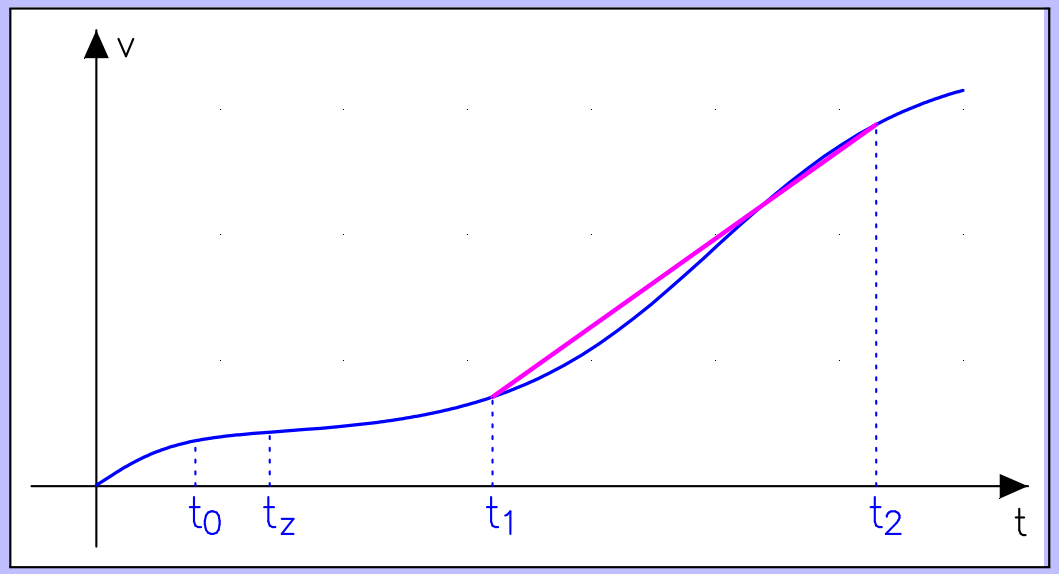
- 20%-40% of available data
- maximal and mean error
- standard deviation

Auto correlation



Accident Severity

with A. Kuhn, J. Urbahn, BMW AG, 2000



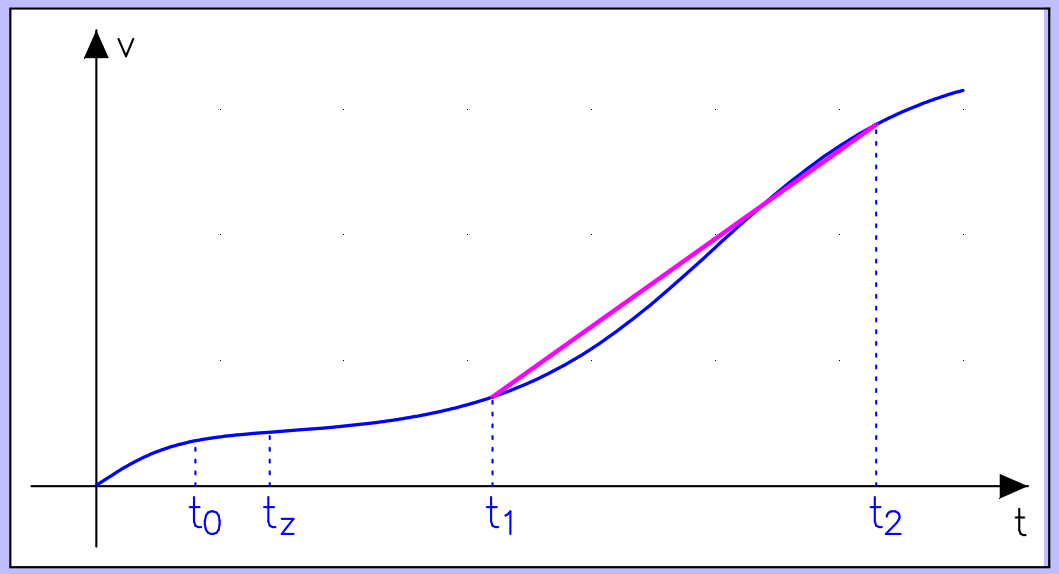
- t_0: decision to fire airbag ...
- t_Z: ignition of airbag ($t_1 - t_Z \approx 30\,\mathrm{ms}$)
- t_1: driver starts forward displacement
- t_2: acceleration decreases

Targets

1. predict the severity of the accident
2. help deciding which action to be taken
3. protect the passengers as good as possible

Accident Severity

with A. Kuhn, J. Urbahn, BMW AG, 2000



- t_0: decision to fire airbag ...
- t_Z: ignition of airbag ($t_1 - t_Z \approx 30\,\mathrm{ms}$)
- t_1: driver starts forward displacement
- t_2: acceleration decreases

Targets

1. predict the severity of the accident
2. help deciding which action to be taken
3. protect the passengers as good as possible

Targets of the Project

Accident severity: possible parameters

1. (mean) velocity of passengers (time, forward displacement)
2. mean acceleration of passengers

Data base

Data from parameter variations with Monte-Carlo method:

1. variation of relevant parameters and testing mode
2. FEM simulations using PamCrash
3. 150 - 300 data sets for every 14 models

Targets of the Project

Accident severity: possible parameters

1. (mean) velocity of passengers (time, forward displacement)
2. mean acceleration of passengers

Data base

Data from parameter variations with Monte-Carlo method:

1. variation of relevant parameters and testing mode
2. FEM simulations using PamCrash
3. 150 - 300 data sets for every 14 models

Targets of the Project

Accident severity: possible parameters

1. (mean) velocity of passengers (time, forward displacement)
2. mean acceleration of passengers

Data base

Data from parameter variations with Monte-Carlo method:

1. variation of relevant parameters and testing mode
2. FEM simulations using PamCrash
3. 150 - 300 data sets for every 14 models

Data from some real crash tests

Targets of the Project

Accident severity: possible parameters

1. (mean) velocity of passengers (time, forward displacement)
2. mean acceleration of passengers

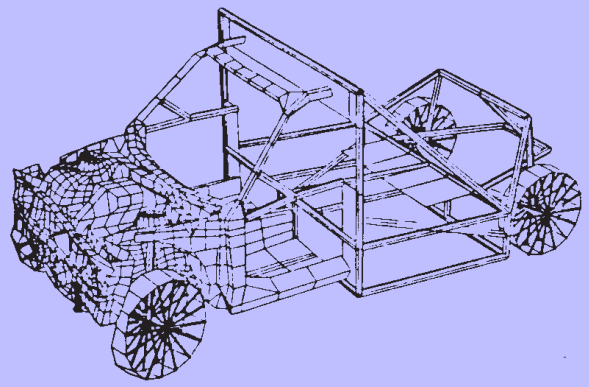
Data base

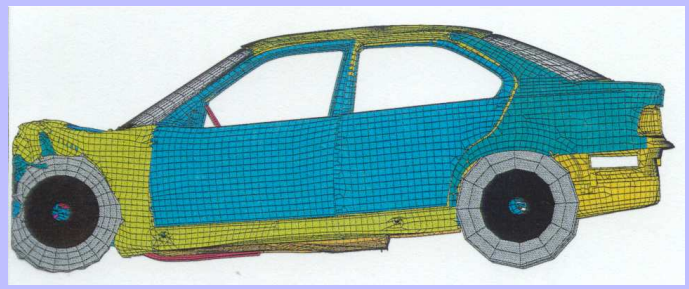
Data from parameter variations with Monte-Carlo method:

1. variation of relevant parameters and testing mode
2. FEM simulations using PamCrash
3. 150 - 300 data sets for every 14 models

Data from some real crash tests

Made possible by

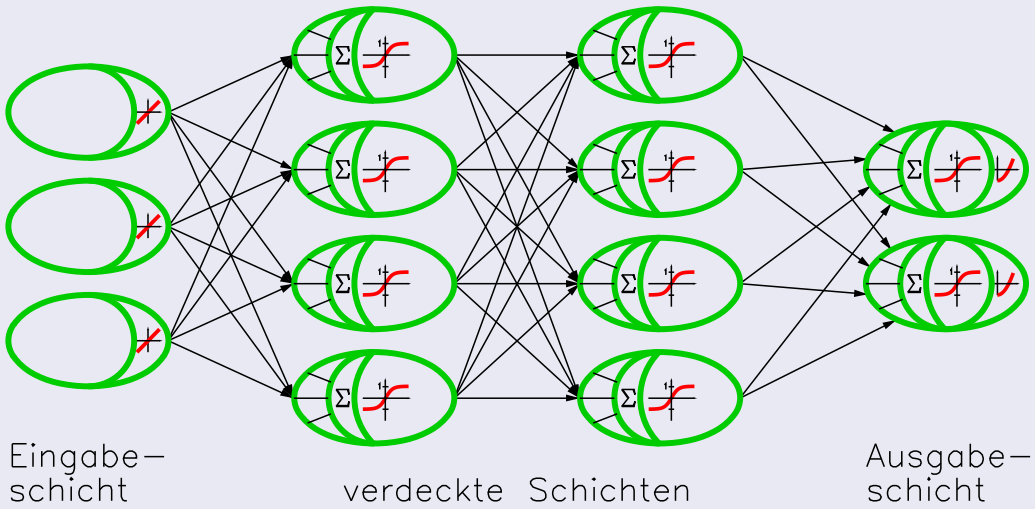




more computer power!

Using the Power of Neural Networks

3- or 4-layer networks



Input

- accelerations, velocities, displacements
- maximal and mean values

Output

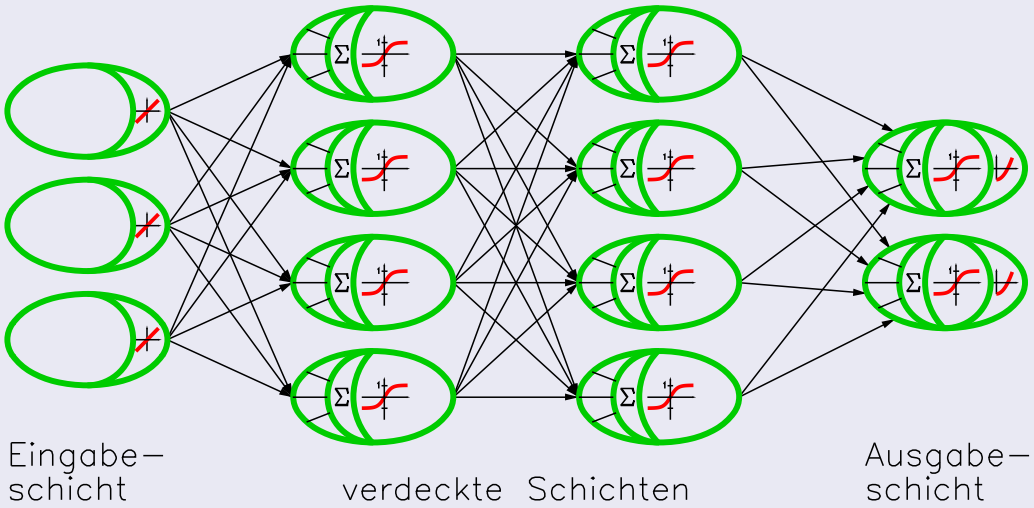
1. velocity
2. mean acceleration (impact to passengers)

Learning:

- activation function: tangential, piecewise parabola
- learning method: gradient descent, Levenberg-Marquardt

Using the Power of Neural Networks

3- or 4-layer networks



Input

- accelerations, velocities, displacements
- maximal and mean values

Output

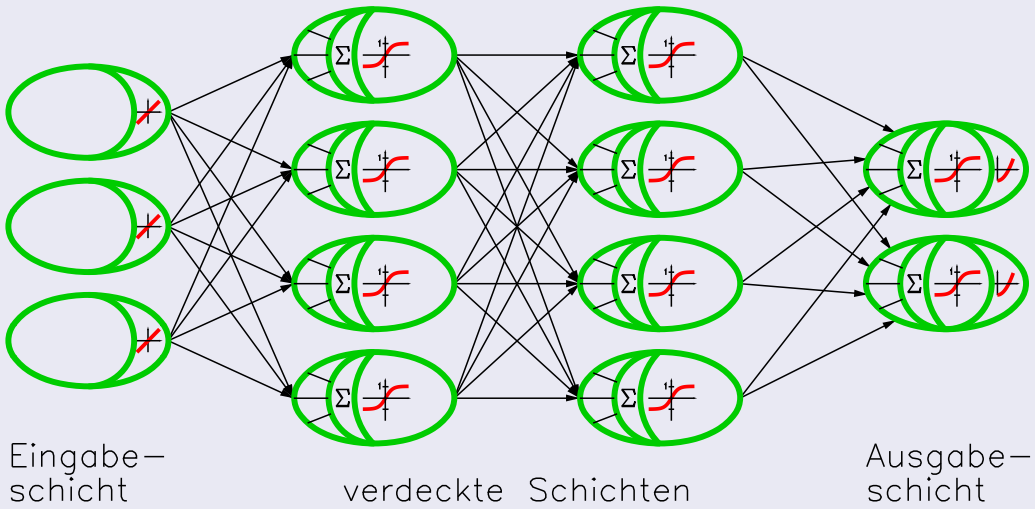
1. velocity
2. mean acceleration (impact to passengers)

Learning:

- activation function: tangential, piecewise parabola
- learning method: gradient descent, Levenberg-Marquardt

Using the Power of Neural Networks

3- or 4-layer networks



Input

- accelerations, velocities, displacements
- maximal and mean values

Output

1. velocity
2. mean acceleration (impact to passengers)

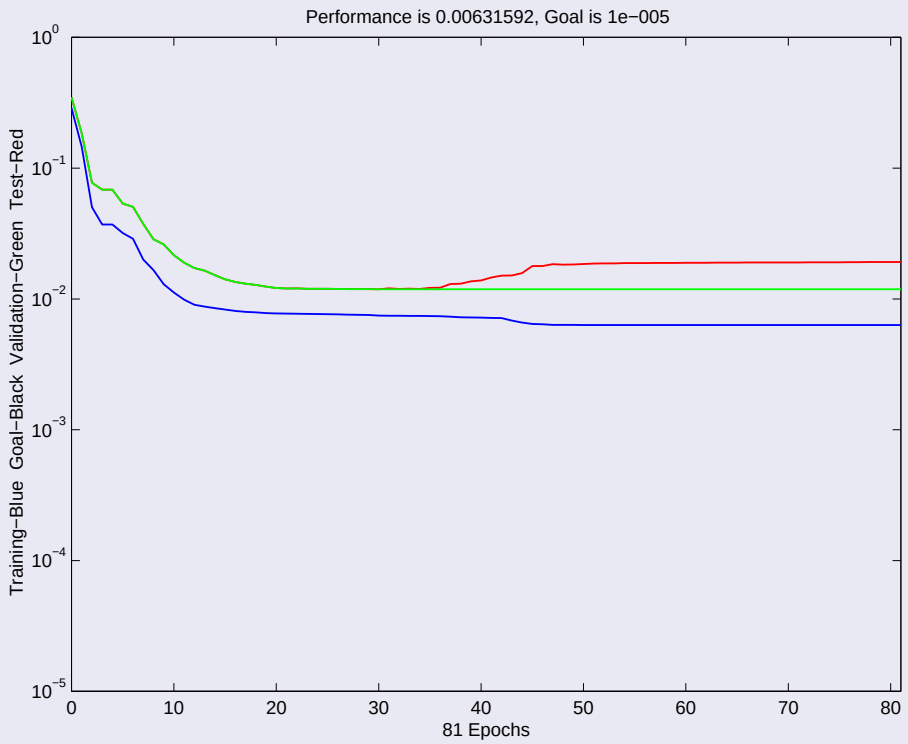
Learning:

- activation function: tangential, piecewise parabola
- learning method: gradient descent, Levenberg-Marquardt

Training the Networks: Learning Strategy

- random choice of 60% learning, 40% testing data
- stop training when the error in testing data increases

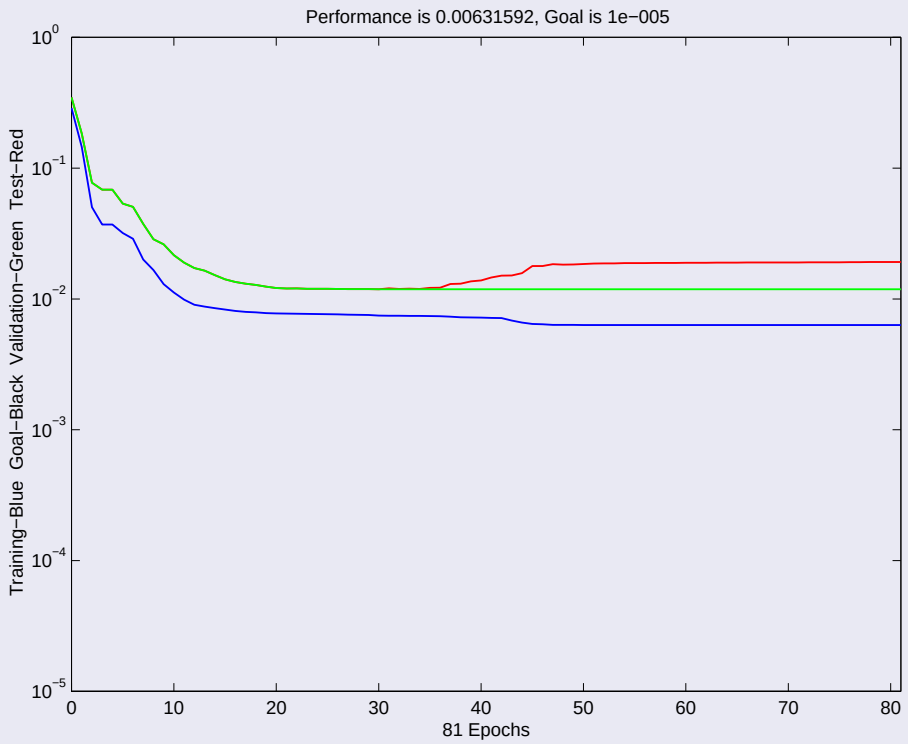
Mean learning error



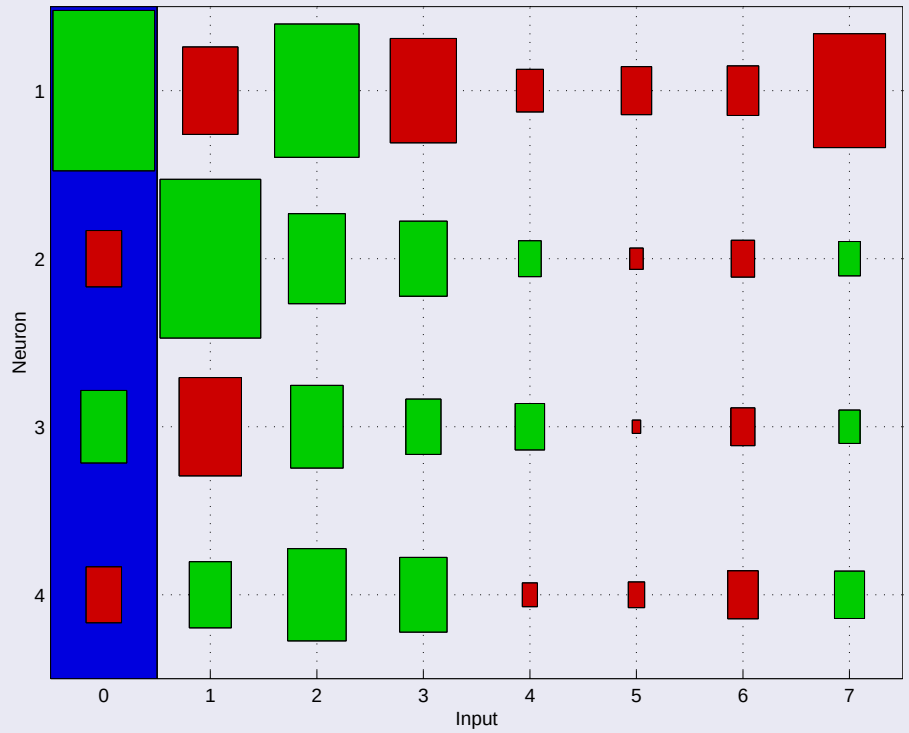
Training the Networks: Influence of the Input

- random choice of 60% learning, 40% testing data
- stop training when the error in testing data increases

Mean learning error

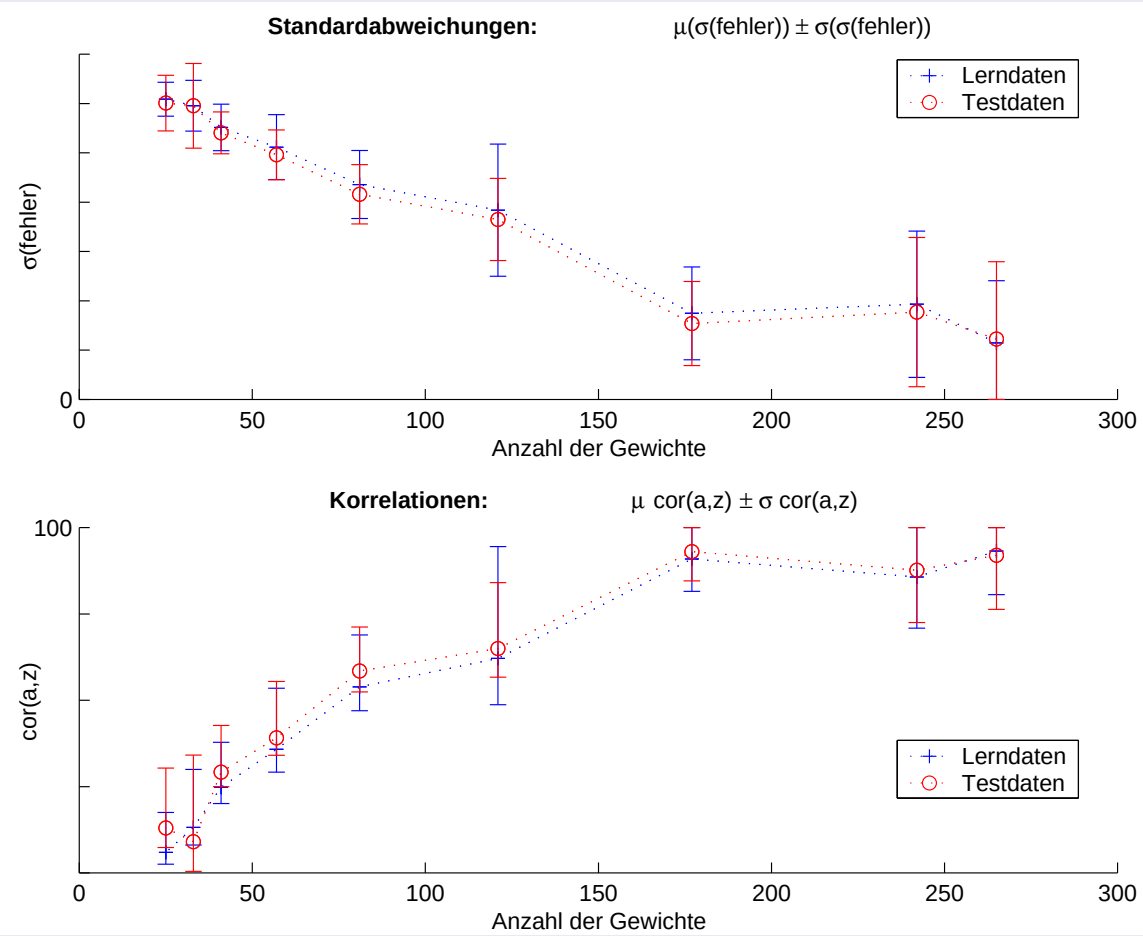


Weights in the first layer



Optimization

Statistics on the number of neurons (1 - 2 hidden layers)



Graphs:

- $\sigma(\sigma(o_i - z_i))$
- correlation

Expectation:

- $\sigma(\sigma)$ gets smaller up to saturation.
- error in learning data only a bit better than testing data

Results

Models

1. FEM simulation data gives a good data base.
2. Usable topologies: e.g.: 4-15-8-1, 4-33-1
3. Usable parameter: mean acceleration
4. Usable input:
   mean accelerations and velocities

σ^2-method allows:

- to choose a network of an appropriate size.
- to judge on the quality of the data.

Results

Models

1. FEM simulation data gives a good data base.
2. Usable topologies: e.g.: 4-15-8-1, 4-33-1
3. Usable parameter: mean acceleration
4. Usable input:
   mean accelerations and velocities

σ^2-method allows:

- to choose a network of an appropriate size.
- to judge on the quality of the data.

Active Damping of Torsion with Ch. Hornung, G. Pflanz, BMW AG, 2005

Problem of a Cabrio: lack of torsion stiffness M_x/d_y

Limousine: 100 %

Cabrio 7.3%

Origin of Unwanted Vibration

Anregung

Car excitation mainly caused by wheel resonance,

$\Rightarrow$ Excitation is transmitted by joints through the axes and spring strut,

$\Rightarrow$ Vibration is observed by passengers.

Origin of Unwanted Vibration

Anregung

Car excitation mainly caused by wheel resonance,

$\Rightarrow$ Excitation is transmitted by joints through the axes and spring strut,

$\Rightarrow$ Vibration is observed by passengers.

Origin of Unwanted Vibration

Anregung

Car excitation mainly caused by wheel resonance,

$\Rightarrow$ Excitation is transmitted by joints through the axes and spring strut,

$\Rightarrow$ Vibration is observed by passengers.

Active Damping: Actuators produce counter-displacement

Sensors and Actuators

- Sensors realize a disturbance
- Actuators produce opposite displacement

$\Rightarrow$ No displacement at the windscreen panel

Active Damping: Actuators produce counter-displacement

Sensors and Actuators

- Sensors realize a disturbance
- Actuators produce opposite displacement

$\Rightarrow$ No displacement at the windscreen panel

Active Damping: Actuators produce counter-displacement

Sensors and Actuators

- Sensors realize a disturbance
- Actuators produce opposite displacement

$\Rightarrow$ No displacement at the windscreen panel

Training

Models

- One or all velocities
- Time series up to 500 ms
- Combination of accelerations

Training of the neural networks

- At least 40% data for testing
- Gradient descent, Levenberg-Marquardt
- Termination: errors in testing data increase

Training

Models

- One or all velocities
- Time series up to 500 ms
- Combination of accelerations

Training of the neural networks

- At least 40% data for testing
- Gradient descent, Levenberg-Marquardt
- Termination: errors in testing data increase

Training and Results

Models

- One or all velocities
- Time series up to 500 ms
- Combination of accelerations

Training of the neural networks

- At least 40% data for testing
- Gradient descent, Levenberg-Marquardt
- Termination: errors in testing data increase

Good results

- time series ca. 200 ms ,
- 2 and 4 input signals,
- both training methods
- small networks $\Rightarrow$ strongly linear behaviour of car body and actuator

Validation

Validation

- Integration of trained network into a simulink-model
- Neural network gives slightly better results than a linear control

Validation

Validation

- Integration of trained network into a simulink-model
- Neural network gives slightly better results than a linear control

Pattern Recognition

Insolvency Detection

Based on annual reports a forecast on the risque of insolvency is given.

Pattern Recognition

Controlling vehicles and robots

Neural Network controls a vehicle or robot. It is trained „on the job“.

Insolvency Detection

Based on annual reports a forecast on the risque of insolvency is given.

Pattern Recognition

Controlling vehicles and robots

Neural Network controls a vehicle or robot. It is trained „on the job“.

Insolvency Detection

Based on annual reports a forecast on the risque of insolvency is given.

Forecast of share value

Forecast based on previous share values and economic data of the company.

Pattern Recognition

Controlling vehicles and robots

Neural Network controls a vehicle or robot. It is trained „on the job“.

Potential termination

Based on power consumption and client data companies that might terminate a contract are identified and get discount.

Insolvency Detection

Based on annual reports a forecast on the risque of insolvency is given.

Forecast of share value

Forecast based on previous share values and economic data of the company.

Pattern Recognition

Chemical reactivity

Predection of its reactivity from qauntivative properties of a bonding.

Potential termination

Based on power consumption and client data companies that might terminate a contract are identified and get discount.

Insolvency Detection

Based on annual reports a forecast on the risque of insolvency is given.

Forecast of share value

Forecast based on previous share values and economic data of the company.

Pattern Recognition

Chemical reactivity

Predection of its reactivity from qauntivative properties of a bonding.

Potential termination

Based on power consumption and client data companies that might terminate a contract are identified and get discount.

Origin of olive oil

The concentration of acids determines the origin (region) of Italian olive oil.

Forecast of share value

Forecast based on previous share values and economic data of the company.

Pattern Recognition

Chemical reactivity

Predection of its reactivity from qauntivative properties of a bonding.

Olfaktometer

Micro crystal system with six different piezo-electric crystal sensors: A neural network learns to recognize flavours.

Origin of olive oil

The concentration of acids determines the origin (region) of Italian olive oil.

Forecast of share value

Forecast based on previous share values and economic data of the company.

Pattern Recognition

Chemical reactivity

Predection of its reactivity from qauntivative properties of a bonding.

Olfaktometer

Micro crystal system with six different piezo-electric crystal sensors: A neural network learns to recognize flavours.

Origin of olive oil

The concentration of acids determines the origin (region) of Italian olive oil.

Structure of a protein

Conclusion from the primary structure of a protein to its secondary spacial structure.

Pattern Recognition

Power consumption

Prediction of the power consumption of companies from one year to the next.

Olfaktometer

Micro crystal system with six different piezo-electric crystal sensors: A neural network learns to recognize flavours.

Origin of olive oil

The concentration of acids determines the origin (region) of Italian olive oil.

Structure of a protein

Conclusion from the primary structure of a protein to its secondary spacial structure.

Pattern Recognition

Power consumption

Prediction of the power consumption of companies from one year to the next.

Olfaktometer

Micro crystal system with six different piezo-electric crystal sensors: A neural network learns to recognize flavours.

Neural stetoscope

A neural networks interprets the noise coming through a stethoscope and provides a diagnoses of a heart problem.

Structure of a protein

Conclusion from the primary structure of a protein to its secondary spacial structure.

Pattern Recognition

Power consumption

Prediction of the power consumption of companies from one year to the next.

Breaking torque

Determining the breaking torque from hydraulic pressure and velocity.

Neural stetoscope

A neural networks interprets the noise coming through a stethoscope and provides a diagnoses of a heart problem.

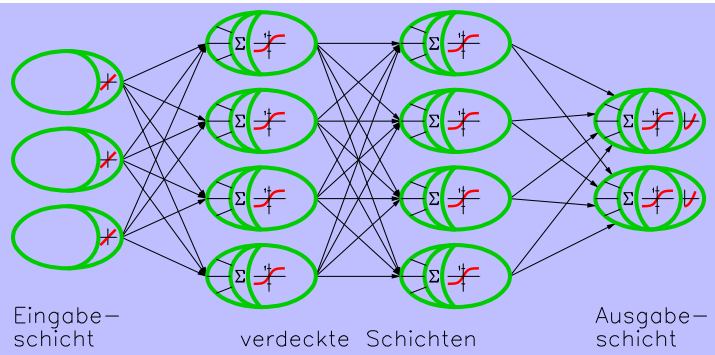
Structure of a protein

Conclusion from the primary structure of a protein to its secondary spacial structure.

Conclusion

Neuronal networks are able to

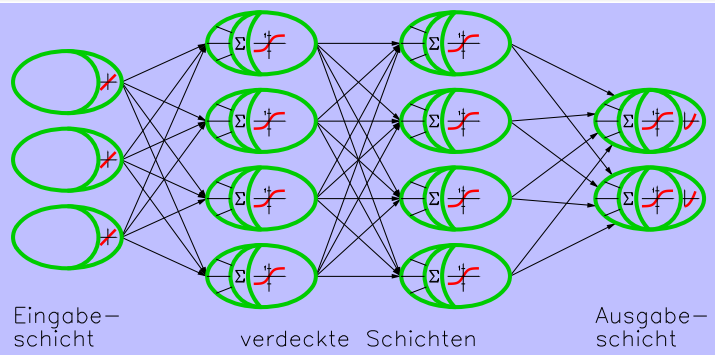
- learn and store know how of a system,
- map functional dependencies



Conclusion

Neuronal networks are able to

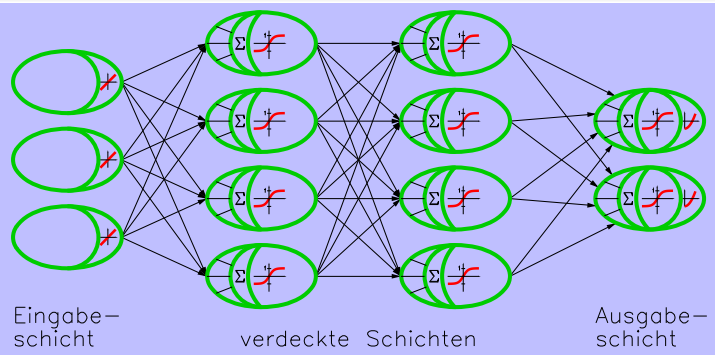
- learn and store know how of a system,
- map functional dependencies



Conclusion

Neuronal networks are able to

- learn and store know how of a system,
- map functional dependencies

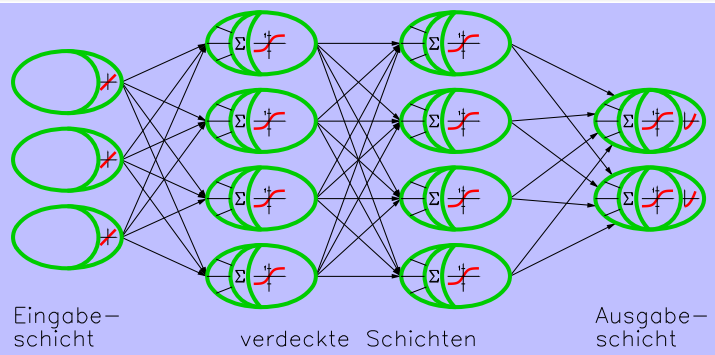


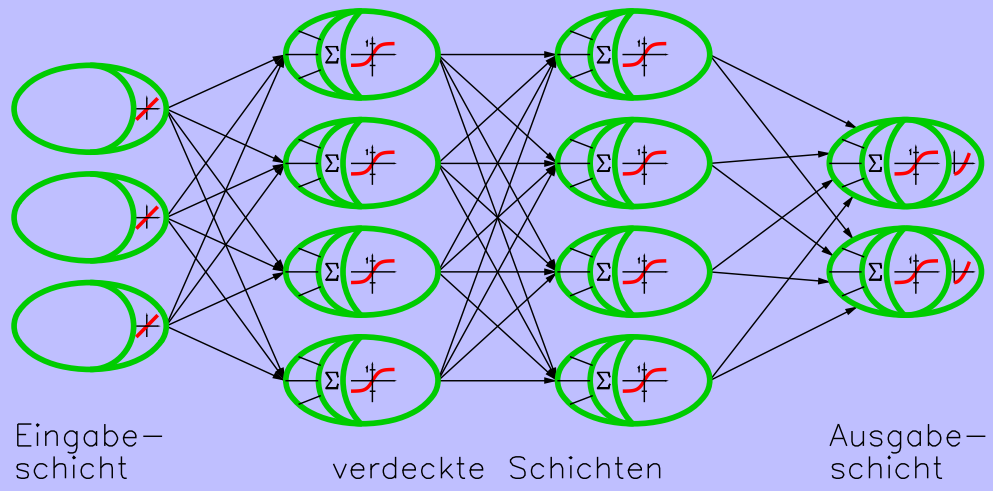
Conclusion

Neuronal networks are able to

- learn and store know how of a system,
- map functional dependencies

using a smooth or balancing interpolation between sampling points.





Thank you for listening to my talk!