Technische Universität Wien

Wintersemester 2005/2006
VO 181.138 Einführung in die Artificial Intelligence

Einführung in Neuronale Netze

Oliver Frölich
Abteilung für Datenbanken und Artificial Intelligence
Institut für Informationssysteme
TU Wien

Überblick

Einführung und Motivation

NN in der Biologie

Künstliche Neuronale Netze

Motivation: Mensch und Maschine

Fähigkeiten des Menschen	Fähigkeiten des Rechners
schnelle Mustererkennung	schnelle arithmetische Operationen
analoges Schließen, Fehlertoleranz	exakte Berechnungen, fehlerfrei
Assoziation und Interpolation	Genauigkeit, Gleichheit
Ähnlichkeiten kontextabhängig erkennen	Gleichheit kontextunabhängig schnell prüfen
Flexibilität, Leistungsreserven	konstante Leistung auch bei Dauerbelastung
parallele Verarbeitung von Informationen	sequentielle Verarbeitung (von Neumann) von Informationen
einfachen Elementaroperationen	komplexen Elementaroperationen (CISC)
große Kapazität für ungenaue Muster	große Speicherkapazitäten möglich
Assoziativer Zugriff auf verteilte Daten	Datenadressierung, lokale Speicherung
Kreativität, Erweitern der Begriffswelt	Schnelles Manipulieren einer festen Begriffswelt
Phantasie, Einführung hypothetischer Welten	
lernen, konditionieren	programmieren

Was sind Neuronale Netze?

- Simulation der Arbeitsweise des Gehirns
- erlauben „biologische“ Informationsverarbeitung
- sind gleichzeitig ein technisches Konzept zur Lösung von Problemen:
  - Mustererkennung
  - Vorhersage
  - Planen
  - …

Vorteile

- Weniger notwendig relevante Faktoren der Eingabe zu bestimmen (im Gegensatz zu wissensbasierten Systemen)
- Im Gegensatz zu statistischen Modellen einfacher zu entwickeln
- Fehlertoleranz
- Inherente Parallelität
- Lernen (Anpassung von Parametern)
- Gute Ergebnisse bei Klassifikation und Vorhersage

Überblick

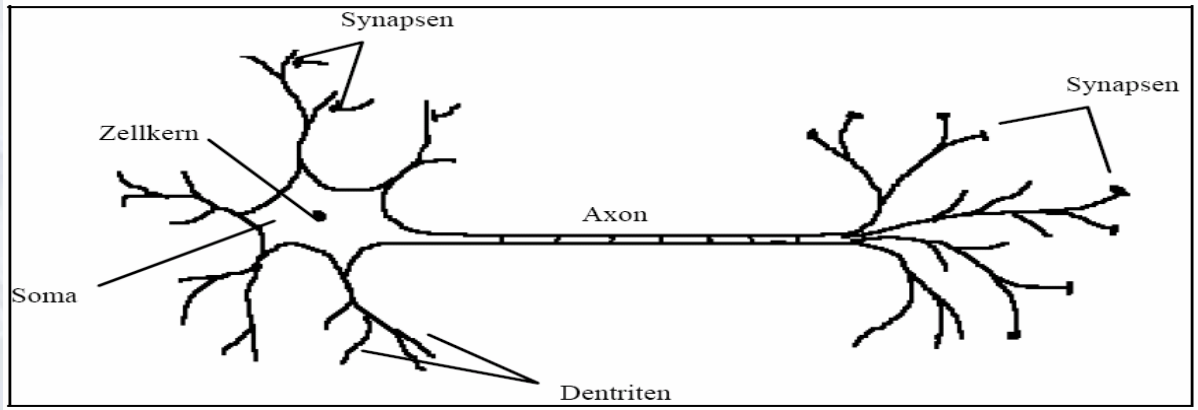
Einführung und Motivation

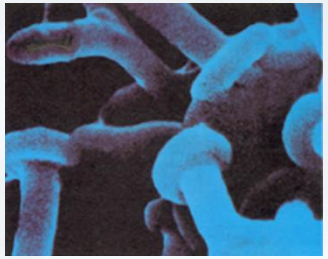
NN in der Biologie

Künstliche Neuronale Netze

Neuronale Netze in der Biologie

- beziehen sich auf Strukturen des Gehirns von Tieren und Menschen.
- Neuronen sind eine best. Art von Nervenzellen, aus denen Strukturen des Gehirns aufgebaut sind.
- Jedes Neuron besteht aus:
  - einem Zellkörper (**Soma**)
  - den **Dendriten** für den Empfang von Signalen anderer Neuronen
  - dem **Axon** zur Übertragung von Signalen an andere Neuronen mittels baumartig aufgefächerter
  - **Synapsen**, die den Kontakt zu den Dendriten anderer Neuronen herstellen.





Synapsen eines Neurons im menschlichen Gehirn

Neuronale Netze in der Biologie II

- Beispiel menschliche Großhirnrinde:
  - 10 Milliarden Neuronen
  - können parallel arbeiten
  - starke Vernetzung: jedes Neuron ist mit ca. 2.000 anderen Neuronen direkt verbunden
- *Lernen* wird aufgefaßt als Veränderung der Verbindungsstärken der Neuronen
- Ein Neuron wird aktiv und sendet über Axon und Synapsen ein *Signal* (=elektrischer Spannungsstoß)
  - Dies geschieht, wenn die *Summe der gewichteten Eingangssignale* einen Schwellwert überschreitet
  - *Summe der gewichteten Eingangssignale* = räumliches und zeitliches Integral über die Eingangsspannungen
- *Informationen* sind nicht in den einzelnen Neuronen gespeichert, sondern werden durch den gesamten Zustand des Netzes mit allen Verbindungen und Bindungsstärken repräsentiert.

Überblick

Einführung und Motivation

NN in der Biologie

Künstliche Neuronale Netze

-Geschichte

-mathematische Beschreibung (mit Beispielen)

-Klassifikation und Konnektivität

-Lernen im kNN

-Backpropagation Netze (mit Rechenbeispiel)

-Anwendungen

Geschichte der künstlichen Neuronalen Netze

1943 *McCulloch* und *Pitts*: erstes Neuronenmodell

kann bel. arithmetische und logische Funktionen berechnen

noch ohne Lernen

1949 *Hebb*: Lernen von Synapsengewichten (Hebb´sche Regel)

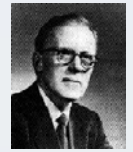
1957/1958 *Rosenblatt:* 1. Neurocomputer Perceptron (mechanisch)

1969: *Minsky* und *Pappert*: vernichtende Kritik der Perceptronen

1986: Rumelhart et al: Backporpagation als mathematisch elegante Lernmethode in n-Schichten Modellen

Verbesserung der Lernalgorithmen, Anwendungen, …

Aufbruchsphase



Hebb

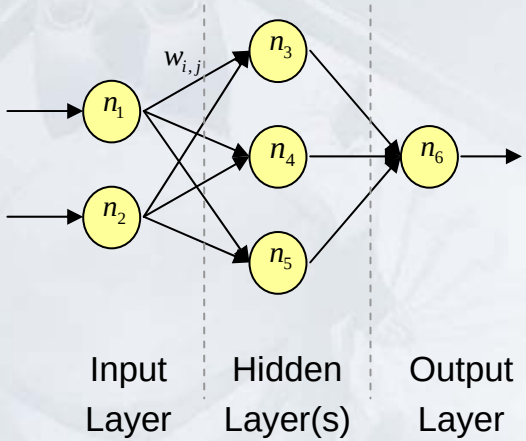
Stagnation

Minsky

Boomphase

Rumelhart

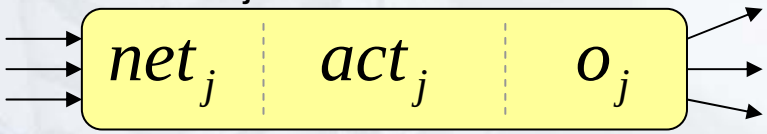
Künstliche Neuronale Netze



Einfaches NN:

- Gerichteter, bewerteter Graph
- schichtartig aufgebaut (Layers)
- Eine bewertete Kante verbindet zwei Neuronen aus unterschiedlichen Schichten

Neuron j:



Inputfunktion: $$net_j(o) = \sum_{i=1}^{n} w_{ij} \cdot o_{ij}$$

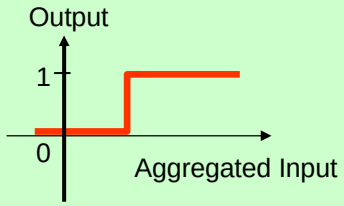
Aktivierungsfunktion: $$a_j = act_j(net_j(o))$$

Outputfunktion: $$o_j = f(a_j) = a_j$$

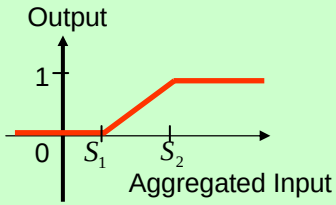
Aktivierungsfunktionen

- Wann „feuert“ das Neuron?
- Berechnung der Aktivierung erfolgt entweder zu diskreten Zeitpunkten oder kontinuierlich (analoge Netze)

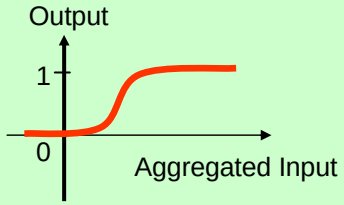
Threshold Logic Function



Threshold Linear Function



Sigmoidal Function



Threshold Logic Function

$$act(x) = \begin{cases} 1: \ falls \ \ net(x) \geq S \\ 0: sonst \end{cases}$$

Threshold Linear Function

$$act(x) = \begin{cases} 1: \ falls \ \ net(x) \geq S_j \\ -ax: \ falls \ \ S_1 \leq net(x) \leq S_2 \\ 0: sonst \end{cases}$$

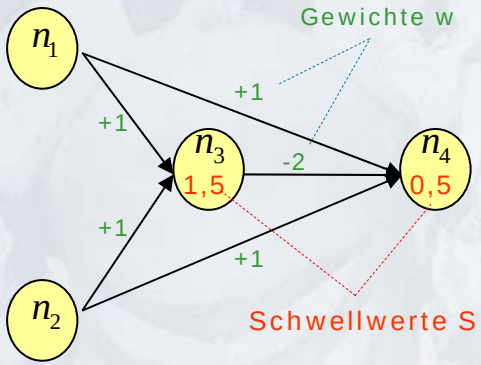
Sigmoidal Function

$$act(x) = \frac{1}{1+e^{-x}}$$

,auch möglich: $act(x) = \tanh(x)$

oder: $act(x) = \sin(x)$

Beispiel für ein einfaches Neuronales Netz: XOR-Netz mit 4 Neuronen



Für die Aktivierungen der Neuronen werden lediglich die binären Werte 0 und 1 verwendet. Die Netzeingabe wird durch folgende **Inputfunktion** berechnet:

$$net_j(x) = \sum_i o_i(x) w_{ij}$$

Als **Aktivierungsfunktion** wird eine binäre Schwellenwertfunktion benutzt:

$$act_j(x) = \begin{cases} 1: falls\, net_j(x) \geq S_j \\ 0: sonst \end{cases}$$

Die **Ausgabefunktion** der Neuronen ist die Identität.

Ausgabe Neuron1:	Ausgabe Neuron2:	Eingabe Neuron3:	Schwellwert Neuron3:	Ausgabe Neuron3:	Eingabe Neuron4:	Schwellenwert Neuron4:	Ausgabe Neuron4:	XOR
o_1	o_2	net_3	S_3	o_3	net_4	S_4	o_4	
0	0	$net_3 = 0*1+0*1 = 0$	1,5	0	$net_4 = 0*1+0*1+0(-2) = 0$	0,5	0	0
0	1	$net_3 = 0*1+1*1 = 1$	1,5	0	$net_4 = 0*1+1*1+0(-2) = 1$	0,5	1	1
1	0	$net_3 = 1*1+0*1 = 1$	1,5	0	$net_4 = 1*1+0*1+0(-2) = 1$	0,5	1	1
1	1	$net_3 = 1*1+1*1 = 2$	1,5	1	$net_4 = 1*1+1*1+1(-2) = 0$	0,5	0	0

Klassifikation

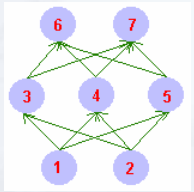
Klassifikation nach mehreren Gesichtspunkten möglich:

- **Applikationsorientiert**
- **Modellorientiert** (Perceptronen, Hopfield-Netze, Kohonen Maps, Backpropagation, …)
- Nach der **Aktivierungsfunktion** (linear, nichtlinear, …)
- Nach dem **Lernverfahren** (supervised, reinforcement, unsupervised)
- Nach der **Konnektivität** (Feedforward-Netze, Recurrent Netze)

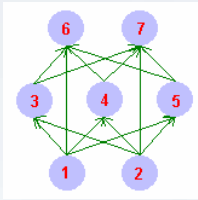
Konnektivität in Neuronalen Netzen

Feedforward-Netze (=Netze ohne Rückkopplung):

- Hier existiert kein Pfad, der von einem Neuron direkt oder über zwischengeschaltete Neuronen wieder zurück zu diesem Neuron führt. Daten werden also nur in eine Richtung weitergegeben.
- Unterscheidung:

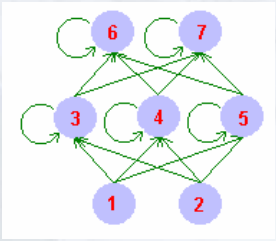


Ebenenweise verbundenes Feedforward-Netz

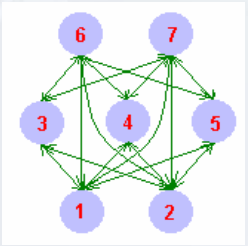


Feedforward-Netz mit Vorwärts-Verbindungen über mehrere Ebenen

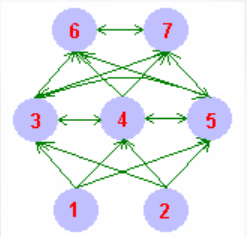
Recurrent-Netze (=Netze mit Rückkopplung):



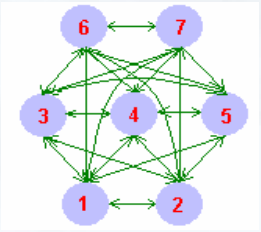
direkte Rückkopplung



indirekte Rückkopplung



laterale Rückkopplung



vollst. verbundenes Netz

Lernen in Neuronalen Netzen

- *Ein neuronales Netz **lernt**, indem es sich gemäß einer fest vorgegebenen Vorschrift, der **Lernregel**, selbst modifiziert.*
- ***Ziel** dabei ist, durch das Netz für eine vorgegebene Eingabe eine gewünschte Ausgabe zu produzieren.*
- *Dies geschieht durch die wiederholte Eingabe von **Trainingsmustern**. Dabei wird versucht, den Fehler zwischen erwarteter und tatsächlicher Ausgabe des Netzes zu minimieren.*
- *Prinzipiell mögliche **Lernprozesse**:*
  - *(1) Entwicklung neuer Verbindungen*
  - *(2) Löschen existierender Verbindungen*
  - *(3) Modifikation der Verbindungsstärke (Veränderung der Gewichte)*
  - *(4) Modifikation des Schwellenwertes*
  - *(5) Modifikation der Aktivierungs- bzw. Ausgabefunktion*
  - *(6) Entwicklung neuer Neuronen*
  - *(7) Löschen bestehender Neuronen*
- *(3) wird am häufigsten verwendet. Verfahren, die auch eine Veränderung der Topologie beinhalten, gewinnen ebenfalls an Bedeutung.*

Lernen in Neuronalen Netzen: Lernparadigmen

Es lassen sich drei Arten unterscheiden:

- ***Supervised Learning (überwachtes Lernen):***

  Hier gibt ein externer Lehrer dem Netz zu jeder Eingabe die Differenz der tatsächlichen zur korrekten Ausgabe an. Anhand dieser Differenz modifiziert dann die Lernregel das Netz. Diese Technik setzt Trainingsdaten voraus, bestehend aus Paaren von Ein- und Ausgabedaten.
- ***Reinforcement Learning:***

  Hier wird dem Netz lediglich mitgeteilt, ob seine Ausgabe korrekt oder inkorrekt war (Reward / Punishment). Das Netz erfährt nicht den exakten Wert des Unterschiedes.
- ***Unsupervised Learning (auch: self-organized Learning):***

  Ohne externen Lehrer versucht das Netz sich selbst zu optimieren durch Kriterien, die das Netz selbst hat.

Backpropagation Learning ist eine best. Form des Supervised Learning mittels Gewichtsveränderungen für Multi-Layer Feedforward Netze.

Backpropagation: Algorithmus

Bei Backpropagation über Gewichtsänderungen werden folgende Schritte durchlaufen:

1. **[Forwardpropagation]:** Die angelegten Eingabewerte x_i laufen vorwärts durch das Netz und erzeugen die Ausgabewerte y_i .

2. **[Backpropagation]:**
   a) Mittels der erwarteten Output-Werte e_j wird der Fehler δ_j berechnet:
      - Falls Neuron j in der Output-Schicht liegt: $\delta_j = act'(x_j) \cdot (e_j - y_j)$
      - Falls Neuron j ein Hidden Neuron ist: $\delta_j = act'(x_j) \cdot \sum_m (\delta_m - w_{jm})$
        (dabei läuft m über alle Neuron j nachgeschalteten Neuronen)

   b) Berechnung der Gewichtsänderung: $\Delta w_{ij} = k \cdot \delta_j \cdot y_i$
      (mit k als Lernrate und y_i als berechneter Output von Neuron i)

3. Wiederhole Schritte 1 und 2 bis der Ausgangsfehler kleiner als ein gegebener Wert ist.

Backpropagation: Algorithmus (II)

Wird als Aktivierungsfunktion die sigmoide Funktion

$$act\,(x) = \frac{1}{1+e^{-x}}$$

verwendet, so erhält man:

$$act'(x) = act(x)\cdot(1-act(x))$$

Dadurch kann die Berechnung folgendermaßen vereinfacht werden:

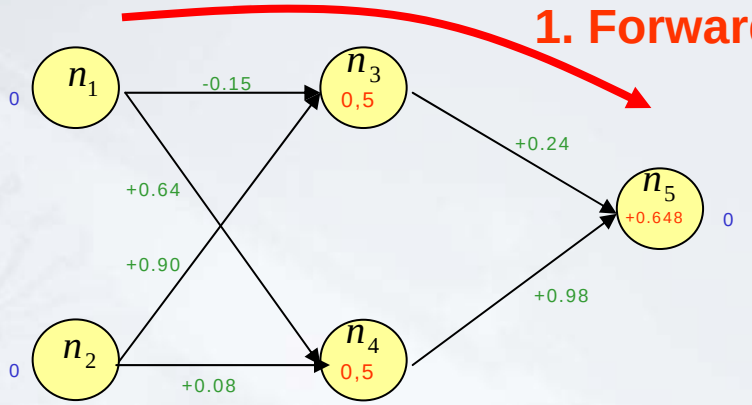
$$\delta_j = (1-y_j)\cdot(e_j - y_j)$$

für ein Output-Neuron j, und

$$\delta_j = y_j \cdot (1-y_j)\cdot\sum_m (\delta_m - w_{jm})$$

für ein Hidden Neuron j.

Backpropagation Learning: Beispiel



$$\delta_{n_5} = y_{n_5} \cdot (1 - y_{n_5}) \cdot (e_{n_5} - y_{n_5}) = 0.648 \cdot (1 - 0.648) \cdot (0 - 0.648) = -0.148$$

$$\delta_{n_4} = y_{n_4} \cdot (1 - y_{n_4}) \cdot (\delta_{n_5} \cdot w_{n_4 n_5}) = 0.5 \cdot (1 - 0.5) \cdot (-0.148 \cdot 0.98) = -0.036$$

Die anderen Fehler-Werte werden analog berechnet:

$$\delta_{n_3} = -0.009$$

$$\delta_{n_2} = +0.011$$

$$\delta_{n_1} = -0.022$$

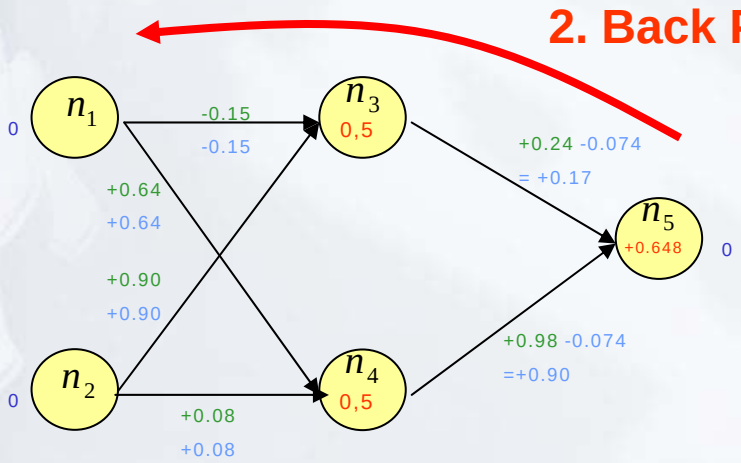
Backpropagation Learning: Beispiel (II)

Berechnung der Gewichtsänderungen (mit $k = 1$):

$$\Delta w_{n_4 n_5} = k \cdot (\delta_{n_5} \cdot y_{n_4}) = 1 \cdot (-0.148) \cdot 0.5 = -0.074$$

Somit ergibt sich das neue Gewicht für $w_{n_4 n_5} = 0.98 - 0.074 = 0.90$

Die anderen Gewichtsveränderungen werden analog berechnet. *Anmerkung: Die angegebenen Zahlen sind gerundet wiedergegeben.*

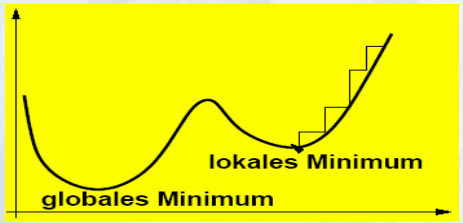


Probleme bei Backpropagation Netzen:

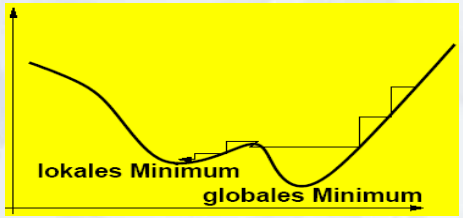
- Anzahl der Hidden Neurons und die Lernrate sind vom Anwendungsgebiet abhängig und sind oft schwierig zu bestimmen.
  - Falls Anzahl der Hidden Neurons *zu klein*: Das Backpropagation-Netz kann das Problem nicht lösen.
  - Falls Anzahl der Hidden Neurons *zu groß*:
    - Es besteht die Gefahr des *Überlernens* (Overtraining).
    - Die überflüssigen Einheiten sind ausschließlich zur Erkennung der restlichen Trainingsmuster bestimmt (quasi lokale Speicherung).
    - Es wird keine grundlegende Abbildungsvorschrift gefunden.

Probleme bei Backpropagation Netzen (II):

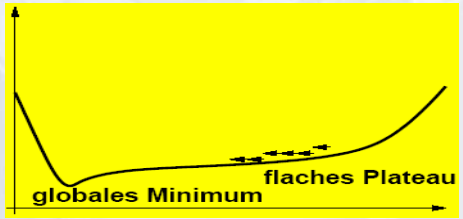
- Schnelle Konvergenz zu einer optimalen Lösung ist nicht immer gegeben:



Optimierung auf nur lokales Minimum möglich



Optimale Minima können während des Lernens verlassen werden



Nur langsames Lernen bei „flachem Plateau“

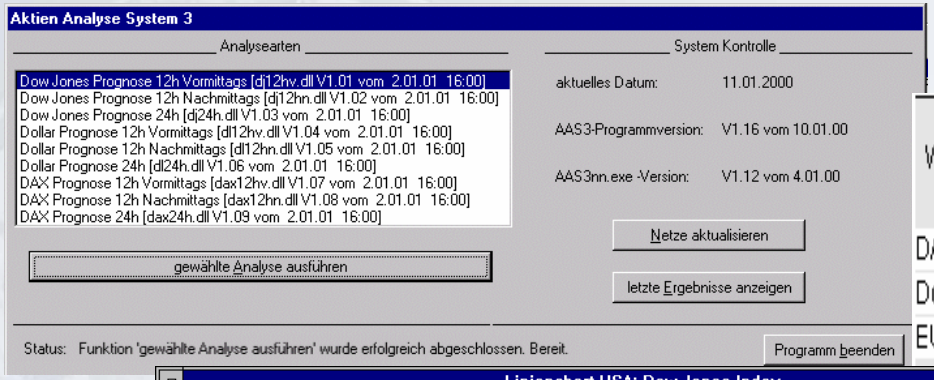
Ausblick: aktuelle Forschung zu Backpropagation

- schnellere Konvergenz, Vermeidung nicht globaler Minima
- Optimierung der Netzwerk-Grösse und Topologie
  - statisch: z.B. mittels genetischer Algorithmen, da die "Berechnung" der optimalen Neuronenzahl schon für einfache Netztypen NP-vollständig ist
  - dynamisch: durch „pruning“ = Ausdünnen der überflüssigen Verbindungen und Neuronen oder Hinzufügen neuer Neuronen und Verbindungen
- Verbindung mit Fuzzy-Logik (Neuro-Fuzzy)

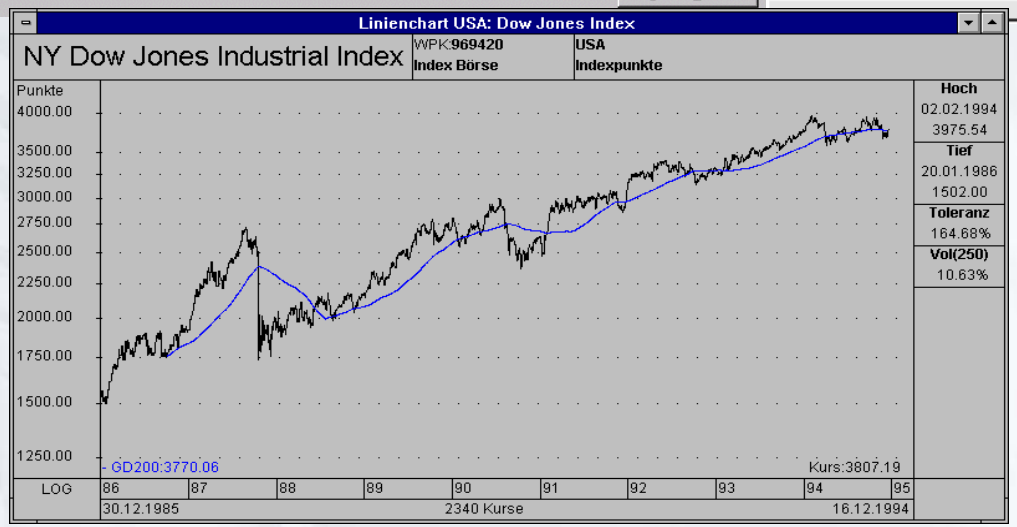
Anwendungen

- Vorhersage von Ozonkonzentrationen (besser als physikalisch/chemisches Modell und statistisches Modell)
- Vorhersage von Aktienkursen
- OCR (Buchstabenerkennung)
- Schriftenerkennung und Unterschriftenerkennung
- Spracherkennung
- SNCF: Fahrplanoptimierungen
- Ampelschaltung (Tübingen)

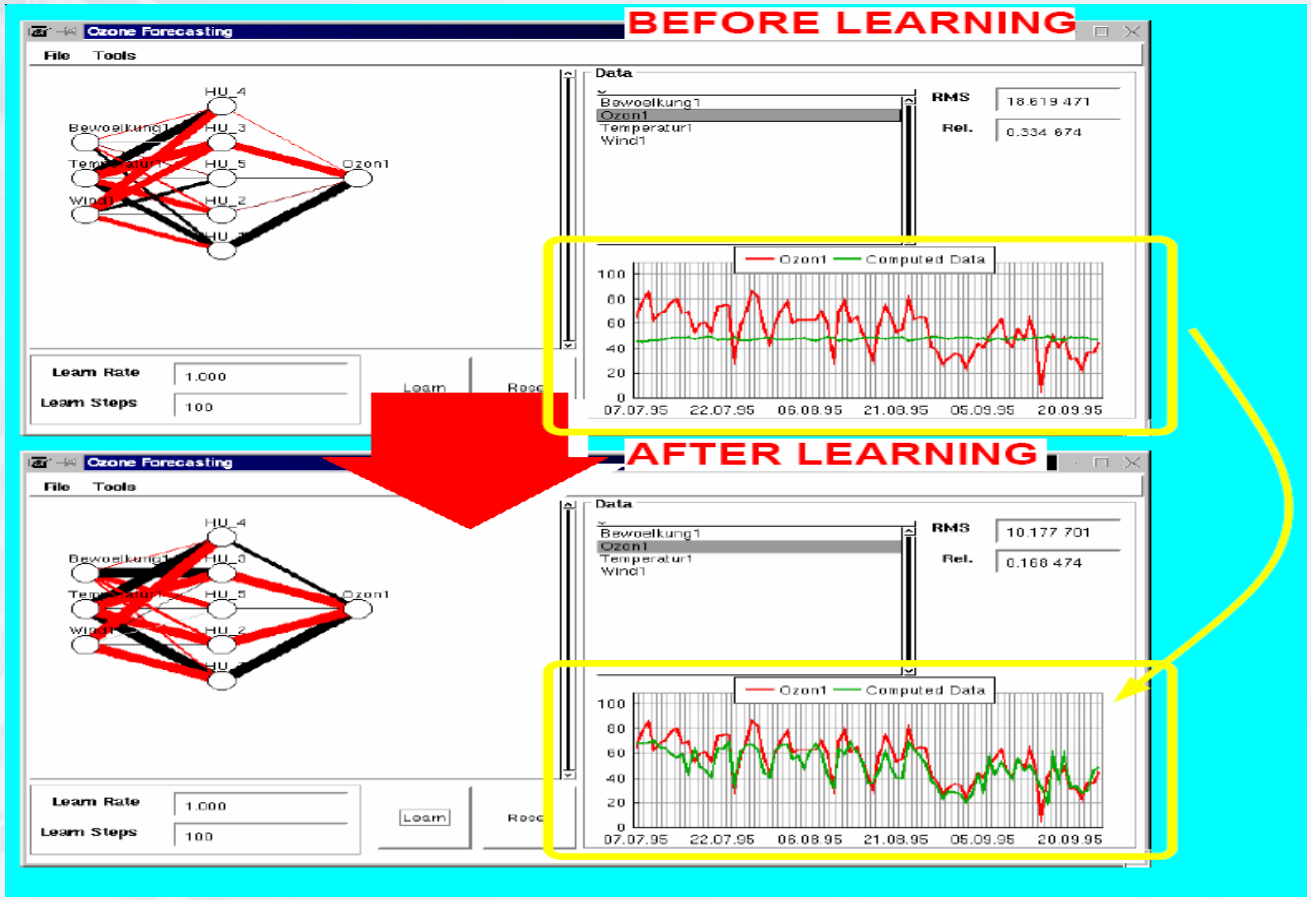
Anwendungen



Wert	WKN	Kurs	Tag Prog.	Tag Qual.	3-Tage Prog.	3-Tage Qual.	Woche Prog.	Woche Qual.	2-Wochen Prog.	2-Wochen Qual.
		8.4		11.4		13.4		15.4		22.4
DAX 30	846900	4.400,68	+0,11	68/72	+0,45	60/60	-0,10	48/60	-0,57	56/72
Dow Jones	969420	10.461,34	+0,21	44/56	+0,44	44/68	+0,69	40/52	-0,13	72/56
EUR/USD	965275	1,2832	+0,18	48/58	+0,27	52/56	+0,79	64/60	+0,47	64/54



Anwendungen (II)



Danke für Ihre Aufmerksamkeit!