Rekurrente Neuronale Netze
Das Elman Netz

Stefan Vikoler

Universität Salzburg
Fachbereich Computerwissenschaften

Betreuung durch
Dr. Claudia-Lidia Badea

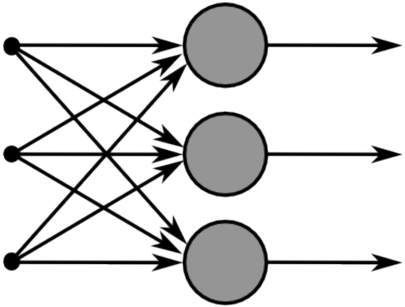
22.06.2012

Architekturen von neuronalen Netzen

- Einschichtige Feedforward Netze
- Mehrschichtige Feedforward Netze
- Rekurrente Netze

Einschichtige Feedforward Netze

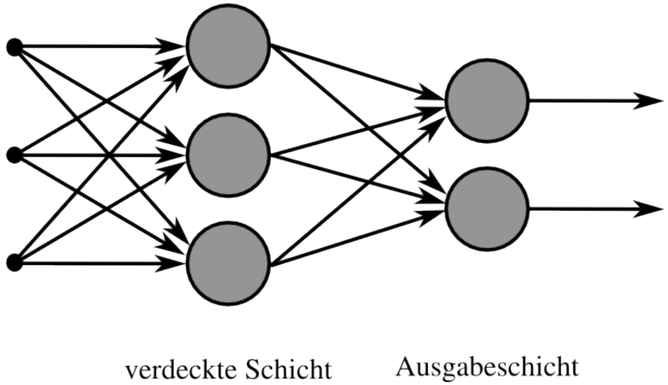
- Schichten
- Input $\rightarrow$ Output



Ausgabeschicht

Mehrschichtige Feedforward Netze

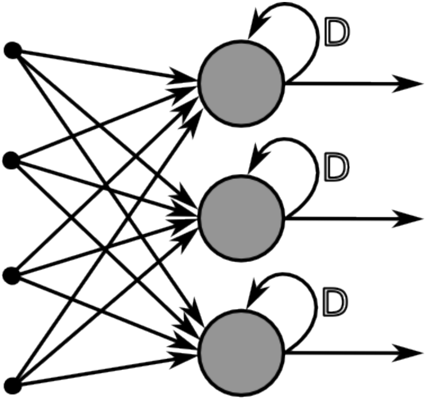
- eine oder mehrere versteckte Schichten
- Berechnung zwischen Input und Output



Rekurrente Netze

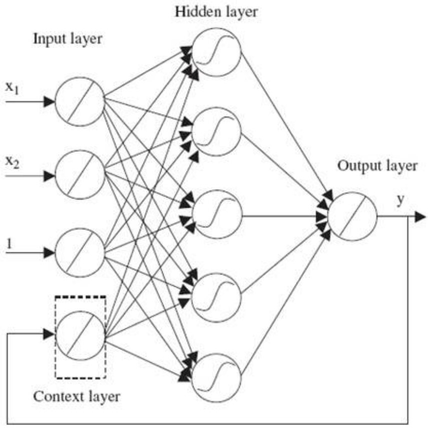
- 'feedback loop'
- Output von Neuronen wird wieder als Input verwendet
- Lernkapazität und Performanz
- Zeitverzögerung

Rekurrente Netze



Ausgabeschicht

Rekurrente Netze



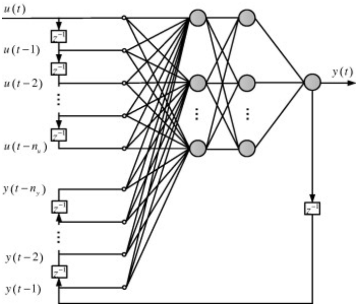
Rekurrente Netze

- 'feedback loop'
- Output von Neuronen wird wieder als Input verwendet
- Lernkapazität und Performanz
- Zeitverzögerung

Modelle von Rekurrenten Netzen

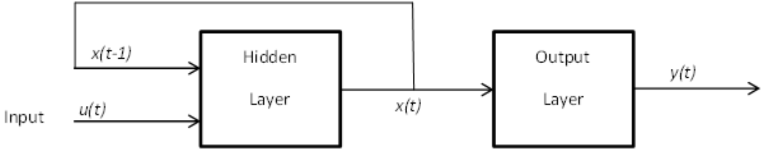
- Input-Output Modell
- State-Space Modell

Input-Output Model



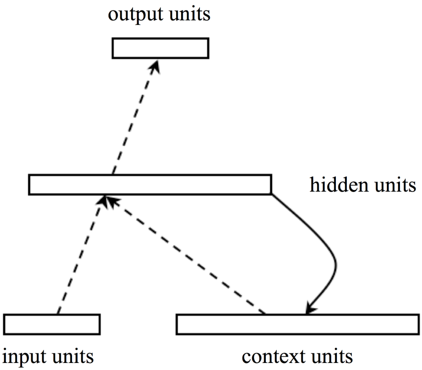
- Inputdaten: $u(t), u(t-1), \ldots, u(t-n_u)$ und $y(t-1), \ldots, y(t-n_y)$
- dynamisches Verhalten:
  $y(t) = F(y(t-1), \ldots, y(t-n_y), u(t), \ldots, u(t-n_u))$

State-Space Model



- Inputdaten: $u(t)$ und $x(t-1)$
- dynamisches Verhalten:
  $x(t) = f(x(t-1), u(t))$ und
  $y(t) = Cx(t)$

Elman Netz

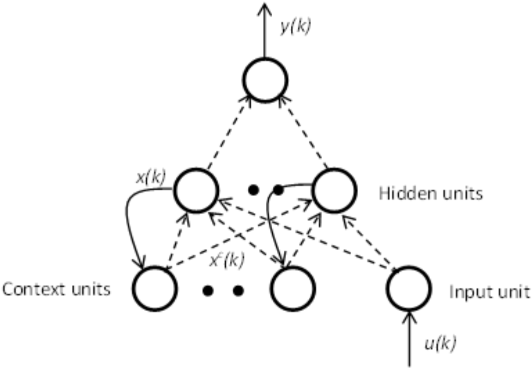


- speichert inneren Zustand in der Kontextschicht

Der Algorithmus

- **dynamic Backpropagation**
- RTRL (Real-Time Recurrent Learning)
- Kalman Filter
- ....

Der Algorithmus



Der Algorithmus

Sei

- externer Input $u(k)$,
- Output des Netzes $y(k)$,
- gesamter Input für ein i-tes verdecktes Neuron $v_i(k)$,
- Output eines i-ten verdeckten Neurons $x_i(k)$,
- Output eines j-ten Kontextneurons $x_j^c(k)$.

Der Algorithmus

Dann gilt:

$$
\begin{aligned}
v_i(k) &= \sum_{j=1}^{n} w_{i,j}^{x}(k-1)x_j^c(k) + w_i^u(k-1)u(k), && \text{(1a)}\\
x_i(k) &= f(v_i), && \text{(1b)}\\
x_j^c(k) &= x_j(k-1), && \text{(1c)}\\
y(k) &= \sum_{i=1}^{n} w_i^y(k-1)x_i(k), && \text{(1d)}
\end{aligned}
$$

wobei $w_i^u, w_{i,j}^x$ und w_i^y, $i,j = 1, \ldots, n$, die Gewichte der Verbindungen zwischen Input- und verdeckter Schicht, Kontext- und verdeckter Schicht und zwischen verdeckter und Outputschicht sind und f eine sigmoidale Aktivierungsfunktion ist.

Der Algorithmus

Wird nun der Input $u(k)$ um einen Zeitschritt verzögert, bevor er gesendet wird, wird $x_j^c(k)$ mit $x_j(k-1)$ ersetzt und angenommen die verdeckten Neuronen sind linear, dann gelten folgende Gleichungen:

$$v_i(k) = \sum_{j=1}^{n} w_{i,j}^x(k-1)x_j(k-1) + w_i^u(k-1)u(k-1), \tag{2a}$$

$$x_i(k) = v_i(k), \tag{2b}$$

$$y(k) = \sum_{i=1}^{n} w_i^y(k-1)x_i(k). \tag{2c}$$

Diese Gleichungen beschreiben ein State-Space Model n-ter Ordnung eines linearen dynamischen Systems und können als folgende Gleichung dargestellt werden:

$$\begin{aligned} y(k) &= A_1 y(k-1) + A_2 y(k-2) + \cdots + A_n y(k-n) \\ &\quad + B_1 u(k-1) + B_2 u(k-2) + \cdots + B_n u(k-n). \end{aligned} \tag{3}$$

Training durch dynamische Backpropagation

Seien die Trainingswerte das Paar $(u(k), y_d(k))$, $k = 1, 2, \ldots, N$, mit $u(k)$ als Input und $y_d(k)$ als gewünschter Output.
Die Fehlerfunktion des Netzes zum Zeitpunkt k ist

$$E_k = \frac{1}{2}\left(y_d(k) - y(k)\right)^2 . \tag{4}$$

Die Gewichte des Netzes werden zu jedem Zeitpunkt k modifiziert. Der Fehlergradient für $w_i^y(k-1)$ ist

$$\begin{aligned} \frac{\partial E_k}{\partial w_i^y(k-1)} &= -(y_d(k) - y(k))\frac{\partial y(k)}{\partial w_i^y(k-1)} \\ &= -(y_d(k) - y(k))x_i(k). \end{aligned} \tag{5}$$

Training durch dynamische Backpropagation

Die Fehlergradienten für $w_i^u(k-1)$ und $w_{i,j}^x(k-1)$ lauten

$$\begin{aligned}
\frac{\partial E_k}{\partial w_i^u(k-1)} &= -\frac{\partial E_k}{\partial y(k)}\frac{\partial y(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial v_i(k)}\frac{\partial v_i(k)}{\partial w_i^u(k-1)} \\
&= -(y_d(k)-y(k))w_i^y(k-1)f'(u(k)). \qquad (6) \\
\frac{\partial E_k}{\partial w_{i,j}^x(k-1)} &= -\frac{\partial E_k}{\partial y(k)}\frac{\partial y(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)} \\
&= -(y_d(k)-y(k))w_i^y(k-1)\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)}. \qquad (7)
\end{aligned}$$

Training durch dynamische Backpropagation

mit

$$
\begin{aligned}
\frac{\partial x_i(k)}{\partial w^x_{i,j}(k-1)} &= \frac{\partial x_i(k)}{\partial v_i(k)} \frac{\partial v_i(k)}{\partial w^x_{i,j}(k-1)} \\
&= f'\left(\frac{\partial v_i(k)}{\partial w^x_{i,j}(k-1)}\right) \\
&= f'\left(x_j(k-1) + \sum_{l=1}^{n} w^x_{i,l}(k-1) \frac{\partial x_l(k-1)}{\partial w^x_{i,j}(k-2)}\right). \qquad (8)
\end{aligned}
$$

Berechnung der Neuronen

Der dynamische Backpropagation-Algorithmus kann nun wie folgt zusammengefasst werden.

Berechnung der Neuronen:

$$v_i(k) = \sum_{j=1}^{n} w_{i,j}^{x}(k-1)x_j(k-1) + w_i^{u}(k-1)u(k), \tag{9a}$$

$$x_i(k) = f(v_i), \tag{9b}$$

$$y(k) = \sum_{i=1}^{n} w_i^{y}(k-1)x_i(k). \tag{9c}$$

Training durch dynamische Backpropagation

dynamische Backpropagation:

$$w_i^y(k) = w_i^y(k-1) - \eta(y_d(k) - y(k))x_i(k), \tag{10a}$$

$$w_i^u(k) = w_i^u(k-1) - \eta(y_d(k) - y(k))w_i^y(k-1)f'(u(k)), \tag{10b}$$

$$w_{i,j}^x(k) = w_{i,j}^x(k-1) - \eta(y_d(k) - y(k))w_i^y(k-1)\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)}, \tag{10c}$$

$$\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)} = f'\left(x_j(k-1) + \sum_{l=1}^{n} w_{i,l}^x(k-1)\frac{\partial x_l(k-1)}{\partial w_{i,j}^x(k-2)}\right). \tag{10d}$$

Anwendungen

- System Identification
- Approximation von Funktionen

System Identification

- experimentelle Annäherung, um einen Prozess oder eine Anlage mit unbekannten Parametern zu modellieren
- Steps:
  - Experimentelles Planen
  - Auswahl einer Modellstruktur
  - Schätzung der Parameter
  - Modellauswertung
- iterativer Prozess
- Modelle:
  - State-Space Model
  - Input-Output Model

System Identification mit Input-Output Model

Seien $u(t)$ der externe Input und $y(t)$ der Netzoutput zum Zeitschritt t und das Input-Output Model wie folgt:

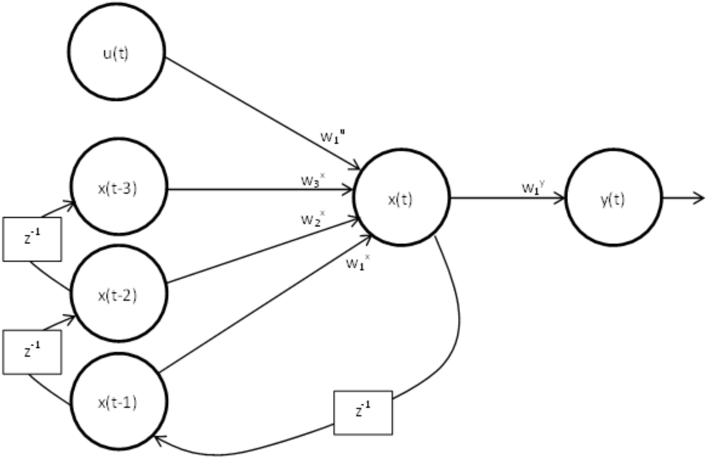
$$y(t) = a_1 y(t-2) + a_2 y(t-2) + a_3 y(t-3) + b_1 u(t-1),$$

wobei $a_1 = 2.627771$, $a_2 = -2.333261$, $a_3 = 0.697676$, $b_1 = 0.017203$.

- Aktivierungsfunktion: $\varphi(x) = \frac{1}{1+e^{-x}}$
- Lernparameter: $\eta = 0,7$
- 100 Input-/Outputwerte
- 100 Epochen

System Identification mit Input-Output Model

Architektur:



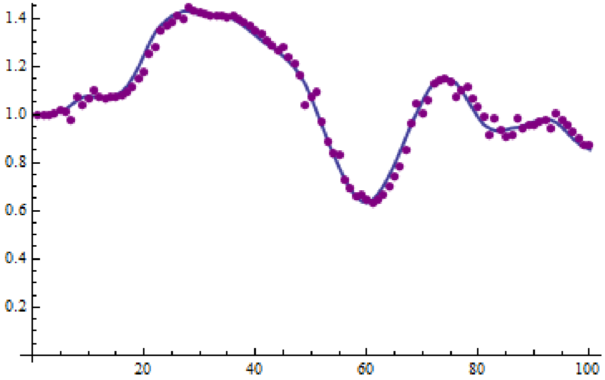
System Identification mit Input-Output Model

Algorithmus:

$$v(t) = w_1^x x(t-1) + w_2^x x(t-2) + w_3^x x(t-3) + w_1^u u(t)$$
$$x(t) = \varphi(v)$$
$$y(t) = w_1^y x(t)$$

$$w_1^u(t) = w_1^u(t-1) - \eta(y_d(t) - y(t))w_1^y(t-1)\varphi'(u(t))$$
$$w_1^x(t) = w_1^x(t-1) - \eta(y_d(t) - y(t))w_1^y(t-1)\frac{\partial x(t)}{\partial w_1^x(t-1)}$$
$$w_2^x(t) = w_2^x(t-1) - \eta(y_d(t) - y(t))w_1^y(t-1)\frac{\partial x(t)}{\partial w_2^x(t-1)}$$
$$w_3^x(t) = w_3^x(t-1) - \eta(y_d(t) - y(t))w_1^y(t-1)\frac{\partial x(t)}{\partial w_3^x(t-1)}$$
$$w_1^y(t) = w_1^y(t-1) - \eta(y_d(t) - y(t))x(t)$$

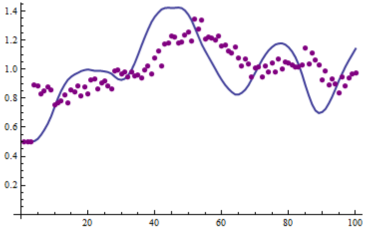
System Identification mit Input-Output Model



System Identification mit Input-Output Model

Probleme:

- Wahl des Lernparameters
- Rundungsfehler mit Double



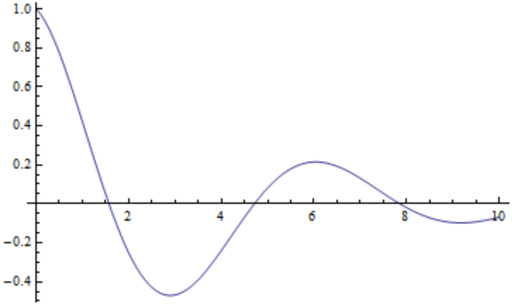
Approximation von Funktionen

- Aktivierungsfunktion: $\varphi(x) = \frac{1}{1+e^{-x}}$
- Lernparameter: $\eta = 0,7$
- 100 Input-/Outputwerte
- 100 Epochen
- selbe Architektur

Approximation von Funktionen

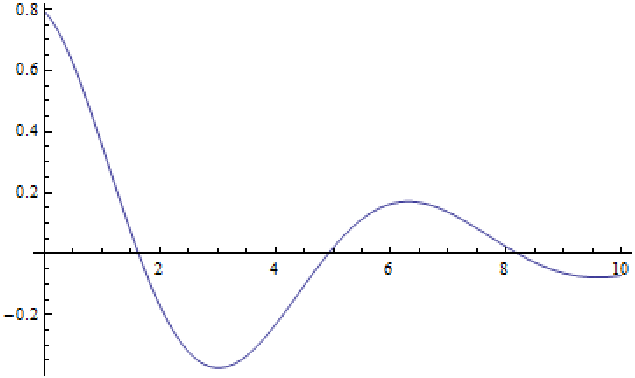
zu approximierende Funktion:

$$e^{-\frac{x}{4}} \cos x$$



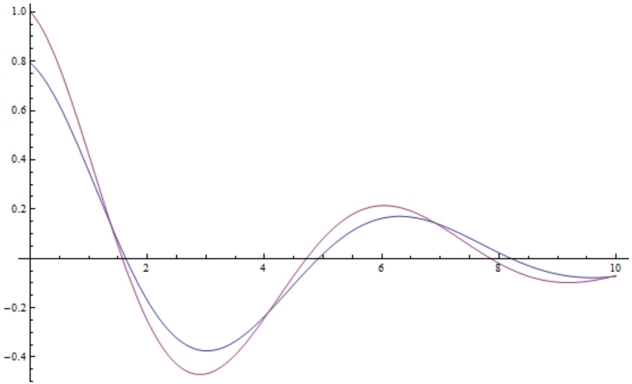
Approximation von Funktionen

Netzoutput:



Approximation von Funktionen

Gegenüberstellung:



Vielen Dank für Ihre Aufmerksamkeit