Neuronale Netze

Prof. Dr. Rudolf Kruse

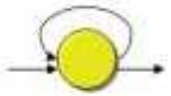
Computational Intelligence
Institut für Intelligente Kooperierende Systeme
Fakultät für Informatik
rudolf.kruse@ovgu.de

Rekurrente Neuronale Netze

Rekurrente Neuronale Netze

Ein **rekurrentes neuronales Netz** ist ein neuronales Netz
mit einem Graph $G = (U, C)$, das Kanten mit einer Rückkopplung enthält:

direkte Rückkopplung: $\exists u \in U \mid u \times u \in C$



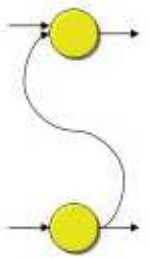
in r-schichteten Netzen:

$U_{\text{hidden}} = U^{(1)}_{\text{hidden}} \cup \cdots \cup U^{(r-2)}_{\text{hidden}}$,

$\forall 1 \leq i < j \leq r-2: \quad U^{(i)}_{\text{hidden}} \cap U^{(j)}_{\text{hidden}} = \emptyset$,

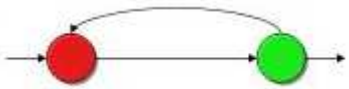
$C \supseteq \left(U_{\text{in}} \times U^{(1)}_{\text{hidden}}\right) \cup \left(\bigcup_{i=1}^{r-3} U^{(i)}_{\text{hidden}} \times U^{(i+1)}_{\text{hidden}}\right) \cup \left(U^{(r-2)}_{\text{hidden}} \times U_{\text{out}}\right)$

seitliche Rückkopplung:



$C \supseteq (U_{\text{in}} \times U_{\text{in}}) \cup \left(\bigcup_{i=1}^{r-3} U^{(i)}_{\text{hidden}} \times U^{(i)}_{\text{hidden}}\right) \cup (U_{\text{out}} \times U_{\text{out}})$

indirekte Rückkopplung von einer hinteren Schicht in eine vordere:



$C \supseteq \left(U^{(1)}_{\text{hidden}} \times U_{\text{in}}\right) \cup \left(\bigcup U^{(j)}_{\text{hidden}} \times U^{(i)}_{\text{hidden}}\right) \ldots \cup \left(U_{\text{out}} \times U^{(r-2)}_{\text{hidden}}\right)$

Wozu Rekurrente Netze?

In viele Anwendungen sind Lernbeispiele zeitabhängig

Standardansatz wäre Verwendung von Differentialgleichungen

Im Folgenden: Differentialgleichungsaufgaben können mit Neuronalen Netze gelöst werden

Idee für Neuronale Netze:
Speichere Aktivierung von Neuronen des aktuellen Lernbeispiels, für das nächste Lernbeispiel

Problem: Fehlerrückpropagation verläuft zyklisch

Inputdaten aus der Praxis: Zeitreihen, Sequenzen, Textdokumente, Aktienkurse, Videos, EKG

Simple Recurrent Networks (SRNs)

Frühes und einfaches Modell für rekurrente Netze

„Speichere“ Aktivierung von Neuronen der **versteckten Schicht** des aktuellen Lernbeispiels in **Kontext-Einheiten** auf Höhe der Eingabeschicht

Verwende Inputs der Kontext-Einheiten für das nächste Lernbeispiel

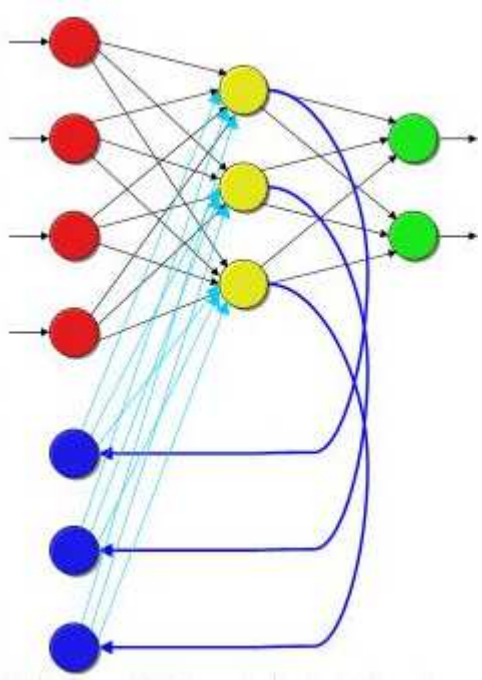
Lösung für Problem der Backpropagation:

Gewichte rekurrenter Kanten sind fixiert auf 1

Backpropagation ignoriert rekurrente Kanten

Kontext-Einheiten enthalten „Kontext“ aus vorherigem Zeitpunkt, deshalb indirekt von jedem vorherigen Zeitpunkt

SRN historisch verwendet für Steuerung von Fahrzeugen



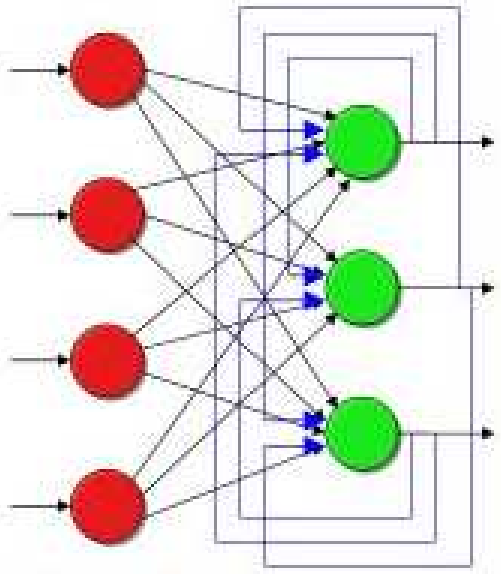
Beispiel für Rekurrente Netze: Autoassociator

Zweischichtiges neuronales Netz (Eingabeschicht und Ausgabeschicht)

Schichten vollständig verbunden

Seitliche Rekurrenzen von jedem Ausgabeneuron zu jedem anderen Ausgabeneuron

Backpropagation passt rekurrente Kanten nur einmal an



Rekurrente Netze: Abkühlungsgesetz

Ein Körper der Temperatur ϑ_0 wird in eine Umgebung der Temperatur ϑ_A eingebracht.

Die Abkühlung/Aufheizung des Körpers kann beschrieben werden durch das **Newtonsche Abkühlungsgesetz**:

$$\frac{\mathrm{d}\vartheta}{\mathrm{d}t} = \dot{\vartheta} = -k(\vartheta - \vartheta_A).$$

Exakte analytische Lösung:

$$\vartheta(t) = \vartheta_A + (\vartheta_0 - \vartheta_A)e^{-k(t-t_0)}$$

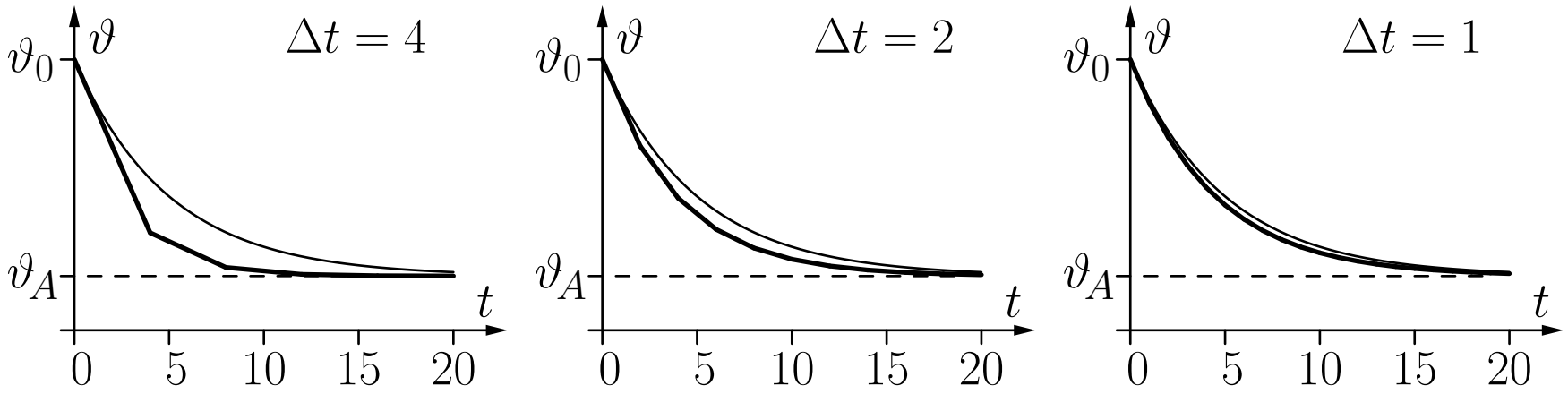
Ungefähre Lösung mit Hilfe des **Euler-Cauchyschen Polygonzuges**:

$$\vartheta_1 = \vartheta(t_1) = \vartheta(t_0) + \dot{\vartheta}(t_0)\Delta t = \vartheta_0 - k(\vartheta_0 - \vartheta_A)\Delta t.$$
$$\vartheta_2 = \vartheta(t_2) = \vartheta(t_1) + \dot{\vartheta}(t_1)\Delta t = \vartheta_1 - k(\vartheta_1 - \vartheta_A)\Delta t.$$

Allgemeine rekursive Gleichung:

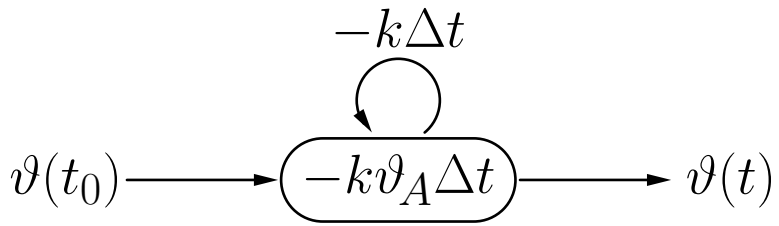
$$\vartheta_i = \vartheta(t_i) = \vartheta(t_{i-1}) + \dot{\vartheta}(t_{i-1})\Delta t = \vartheta_{i-1} - k(\vartheta_{i-1} - \vartheta_A)\Delta t$$

Euler–Cauchy-Polygonzüge für verschiedene Schrittweiten:



Die dünne Kurve ist die genaue analytische Lösung.

Rekurrentes neuronales Netz:



Rekurrente Netze: Abkühlungsgesetz

Formale Herleitung der rekursiven Gleichung:

Ersetze Differentialquotient durch **Differenzenquotient**

$$\frac{\mathrm{d}\vartheta(t)}{\mathrm{d}t} \approx \frac{\Delta\vartheta(t)}{\Delta t} = \frac{\vartheta(t+\Delta t) - \vartheta(t)}{\Delta t}$$

mit hinreichend kleinem Δt. Dann ist

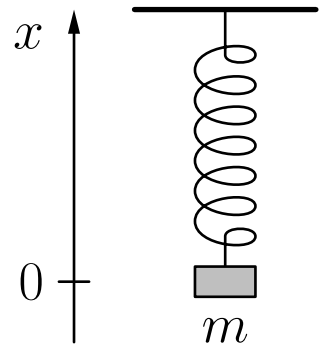
$$\vartheta(t+\Delta t) - \vartheta(t) = \Delta\vartheta(t) \approx -k(\vartheta(t) - \vartheta_A)\Delta t,$$

$$\vartheta(t+\Delta t) - \vartheta(t) = \Delta\vartheta(t) \approx -k\Delta t\vartheta(t) + k\vartheta_A\Delta t$$

und daher

$$\vartheta_i \approx \vartheta_{i-1} - k\Delta t\vartheta_{i-1} + k\vartheta_A\Delta t.$$

Rekurrente Netze: Masse an einer Feder



Zugrundeliegende physikalische Gesetze:

Hooke'sches Gesetz: $F = c\Delta l = -cx$ (c ist eine federabhängige Konstante)

Zweites Newton'sches Gesetz: $F = ma = m\ddot{x}$ (Kraft bewirkt eine Beschleunigung)

Resultierende Differentialgleichung:

$$m\ddot{x} = -cx \qquad \text{oder} \qquad \ddot{x} = -\frac{c}{m}x.$$

Rekurrente Netze: Masse an einer Feder

Allgemeine analytische Lösung der Differentialgleichung:

$$x(t) = a \sin(\omega t) + b \cos(\omega t)$$

mit den Parametern

$$\omega = \sqrt{\frac{c}{m}}, \qquad \begin{aligned} a &= x(t_0) \sin(\omega t_0) \; + \; v(t_0) \cos(\omega t_0), \\ b &= x(t_0) \cos(\omega t_0) \; - \; v(t_0) \sin(\omega t_0). \end{aligned}$$

Mit gegebenen Initialwerten $x(t_0) = x_0$ und $v(t_0) = 0$ und
der zusätzlichen Annahme $t_0 = 0$ bekommen wir den einfachen Ausdruck

$$x(t) = x_0 \cos\left(\sqrt{\frac{c}{m}}\, t\right).$$

Rekurrente Netze: Masse an einer Feder

Wandle Differentialgleichung in zwei gekoppelte Gleichungen um:

$$\dot{x} = v \qquad \text{und} \qquad \dot{v} = -\frac{c}{m}x.$$

Approximiere Differentialquotient durch Differenzenquotient:

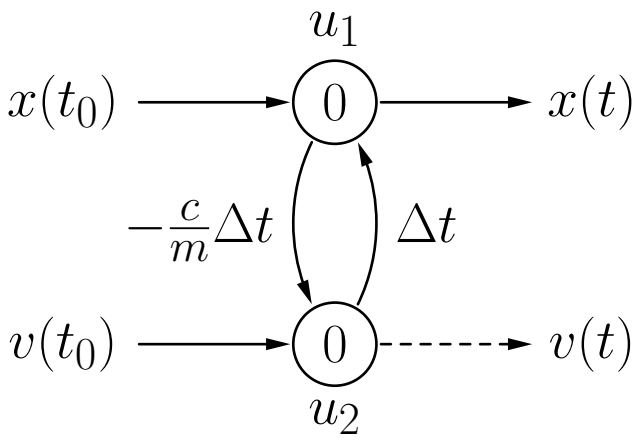
$$\frac{\Delta x}{\Delta t} = \frac{x(t+\Delta t) - x(t)}{\Delta t} = v \qquad \text{und} \qquad \frac{\Delta v}{\Delta t} = \frac{v(t+\Delta t) - v(t)}{\Delta t} = -\frac{c}{m}x$$

Resultierende rekursive Gleichungen:

$$x(t_i) = x(t_{i-1}) + \Delta x(t_{i-1}) = x(t_{i-1}) + \Delta t \cdot v(t_{i-1}) \qquad \text{und}$$

$$v(t_i) = v(t_{i-1}) + \Delta v(t_{i-1}) = v(t_{i-1}) - \frac{c}{m}\Delta t \cdot x(t_{i-1}).$$

Rekurrente Netze: Masse an einer Feder



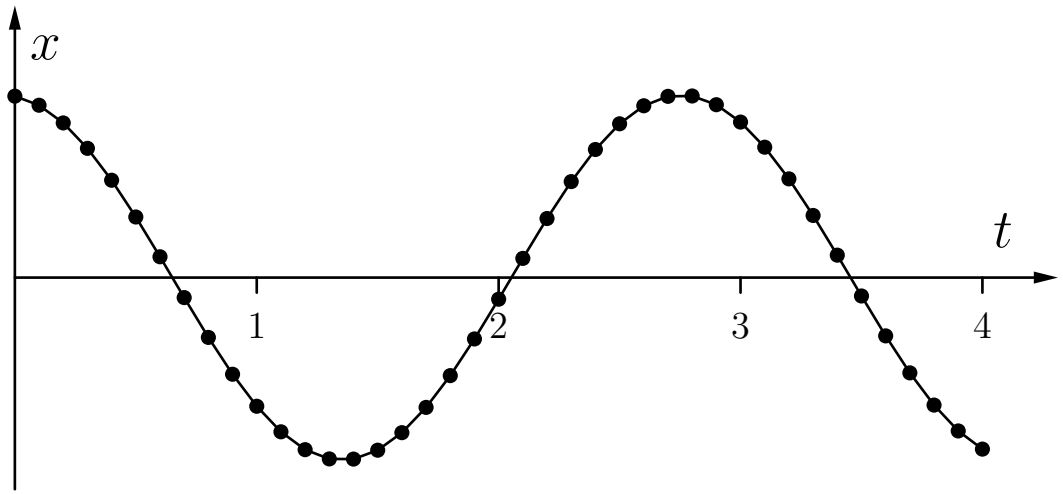
Neuron u_1:
$$f_{\text{net}}^{(u_1)}(v, w_{u_1u_2}) = w_{u_1u_2}v = -\frac{c}{m}\Delta t\ v \qquad \text{und}$$
$$f_{\text{act}}^{(u_1)}(\text{act}_{u_1}, \text{net}_{u_1}, \theta_{u_1}) = \text{act}_{u_1} + \text{net}_{u_1} - \theta_{u_1},$$

Neuron u_2:
$$f_{\text{net}}^{(u_2)}(x, w_{u_2u_1}) = w_{u_2u_1}x = \Delta t\ x \qquad \text{und}$$
$$f_{\text{act}}^{(u_2)}(\text{act}_{u_2}, \text{net}_{u_2}, \theta_{u_2}) = \text{act}_{u_2} + \text{net}_{u_2} - \theta_{u_2}.$$

Rekurrente Netze: Masse an einer Feder

Einige Berechnungsschritte des neuronalen Netzes:

t	v	x
0.0	0.0000	1.0000
0.1	−0.5000	0.9500
0.2	−0.9750	0.8525
0.3	−1.4012	0.7124
0.4	−1.7574	0.5366
0.5	−2.0258	0.3341
0.6	−2.1928	0.1148



Die resultierende Kurve ist nah an der analytischen Lösung.

Die Annäherung wird mit kleinerer Schrittweite besser.

Allgemeine Darstellung expliziter Differentialgleichungen n-ten Grades:

$$x^{(n)} = f(t, x, \dot{x}, \ddot{x}, \ldots, x^{(n-1)})$$

Einführung von $n-1$ Zwischengrößen

$$y_1 = \dot{x}, \qquad y_2 = \ddot{x}, \qquad \ldots \qquad y_{n-1} = x^{(n-1)}$$

Gleichungssystem

$$\begin{aligned}
\dot{x} &= y_1, \\
\dot{y}_1 &= y_2, \\
&\vdots \\
\dot{y}_{n-2} &= y_{n-1}, \\
\dot{y}_{n-1} &= f(t, x, y_1, y_2, \ldots, y_{n-1})
\end{aligned}$$

von n gekoppelten Differentialgleichungen ersten Grades.

Rekurrente Netze: Differentialgleichungen

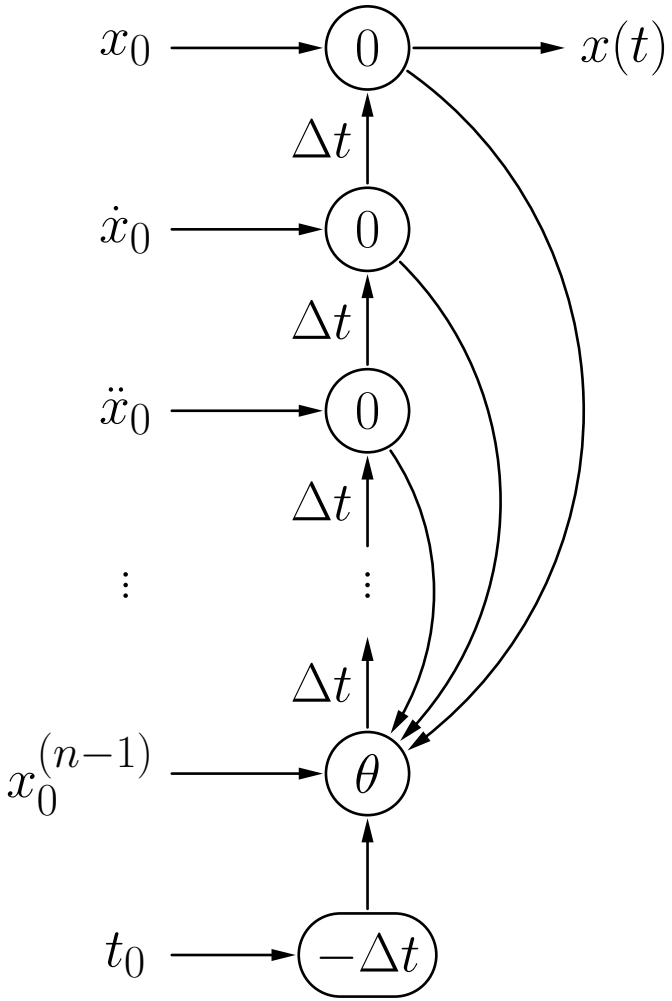
Ersetze Differentialquotient durch Differenzenquotient, um die folgenden rekursiven Gleichungen zu erhalten:

$$
\begin{aligned}
x(t_i) &= x(t_{i-1}) + \Delta t \cdot y_1(t_{i-1}), \\
y_1(t_i) &= y_1(t_{i-1}) + \Delta t \cdot y_2(t_{i-1}), \\
&\vdots \\
y_{n-2}(t_i) &= y_{n-2}(t_{i-1}) + \Delta t \cdot y_{n-3}(t_{i-1}), \\
y_{n-1}(t_i) &= y_{n-1}(t_{i-1}) + f(t_{i-1}, x(t_{i-1}), y_1(t_{i-1}), \ldots, y_{n-1}(t_{i-1}))
\end{aligned}
$$

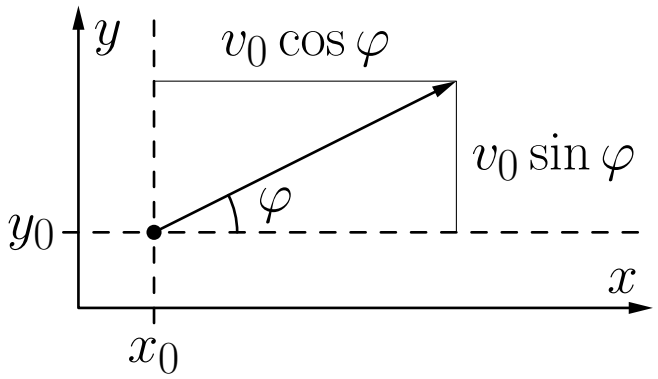
Jede dieser Gleichungen beschreibt die Aktualisierung eines Neurons.

Das letzte Neuron benötigt eine spezielle Aktivierungsfunktion.

Rekurrente Netze: Differentialgleichungen



Rekurrente Netze: Schräger Wurf



Schräger Wurf eines Körpers.

Zwei Differentialgleichungen (eine für jede Koordinatenrichtung):

$$\ddot{x} = 0 \qquad \text{und} \qquad \ddot{y} = -g,$$

wobei $g = 9.81\,\mathrm{ms}^{-2}$.

Anfangsbedingungen $x(t_0) = x_0$, $y(t_0) = y_0$, $\dot{x}(t_0) = v_0 \cos \varphi$ und $\dot{y}(t_0) = v_0 \sin \varphi$.

Rekurrente Netze: Schräger Wurf

Führe Zwischenbedingungen ein:

$$v_x = \dot{x} \qquad \text{und} \qquad v_y = \dot{y}$$

um das System der folgenden Differentialgleichungen zu erhalten:

$$\dot{x} = v_x, \qquad \dot{v}_x = 0,$$

$$\dot{y} = v_y, \qquad \dot{v}_y = -g,$$

aus dem wir das System rekursiver Anpassungsformeln erhalten

$$x(t_i) = x(t_{i-1}) + \Delta t\ v_x(t_{i-1}), \qquad v_x(t_i) = v_x(t_{i-1}),$$

$$y(t_i) = y(t_{i-1}) + \Delta t\ v_y(t_{i-1}), \qquad v_y(t_i) = v_y(t_{i-1}) - \Delta t\ g.$$

Rekurrente Netze: Schräger Wurf

Bessere Beschreibung: Benutze **Vektoren** als Eingaben und Ausgaben

$$\ddot{\vec{r}} = -g\vec{e}_y,$$

wobei $\vec{e}_y = (0, 1)$.

Anfangsbedingungen sind $\vec{r}(t_0) = \vec{r}_0 = (x_0, y_0)$ und $\dot{\vec{r}}(t_0) = \vec{v}_0 = (v_0 \cos\varphi, v_0 \sin\varphi)$.

Führe eine **vektorielle** Zwischengröße $\vec{v} = \dot{\vec{r}}$ ein, um

$$\dot{\vec{r}} = \vec{v}, \qquad\qquad \dot{\vec{v}} = -g\vec{e}_y$$

zu erhalten.

Das führt zu den rekursiven Anpassungsregeln

$$\begin{aligned} \vec{r}(t_i) &= \vec{r}(t_{i-1}) + \Delta t\ \vec{v}(t_{i-1}), \\ \vec{v}(t_i) &= \vec{v}(t_{i-1}) - \Delta t\ g\vec{e}_y \end{aligned}$$

Rekurrente Netze: Schräger Wurf

Die Vorteile vektorieller Netze werden offensichtlich, wenn Reibung mit in Betracht gezogen wird:

$$\vec{a} = -\beta\vec{v} = -\beta\dot{\vec{r}}$$

β ist eine Konstante, die von Größe und Form des Körpers abhängt.
Dies führt zur Differentialgleichung

$$\ddot{\vec{r}} = -\beta\dot{\vec{r}} - g\vec{e}_y.$$

Führe die Zwischengröße $\vec{v} = \dot{\vec{r}}$ ein, um

$$\dot{\vec{r}} = \vec{v}, \qquad\qquad \dot{\vec{v}} = -\beta\vec{v} - g\vec{e}_y,$$

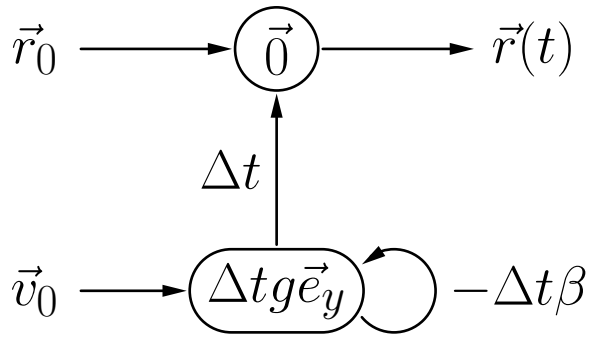
zu erhalten,

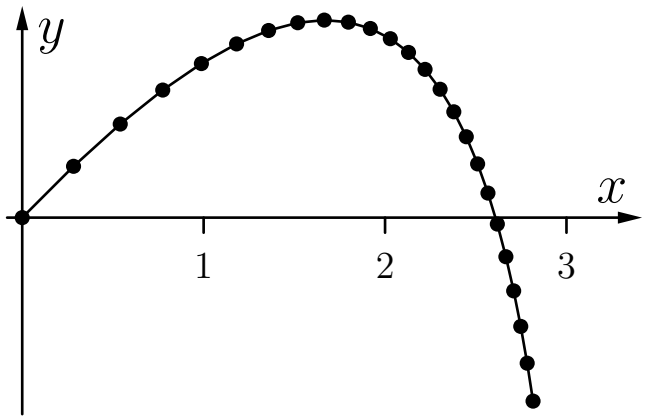
woraus wir die folgenden rekursiven Anpassungsformeln bekommen:

$$\begin{aligned}
\vec{r}(t_i) &= \vec{r}(t_{i-1}) + \Delta t\ \vec{v}(t_{i-1}), \\
\vec{v}(t_i) &= \vec{v}(t_{i-1}) - \Delta t\ \beta\ \vec{v}(t_{i-1}) - \Delta t\ g\vec{e}_y.
\end{aligned}$$

Rekurrente Netze: Schräger Wurf

Sich ergebendes rekurrentes neuronales Netz:





Es gibt keine "seltsamen" Kopplungen wie in einem nicht-vektoriellen Netz.

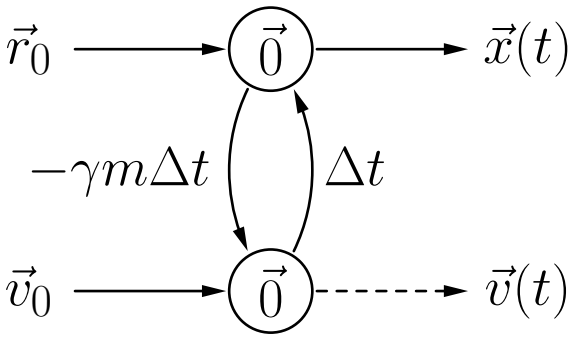
Man beachte die Abweichung von der Parabel, die durch die Reibung bewirkt wird.

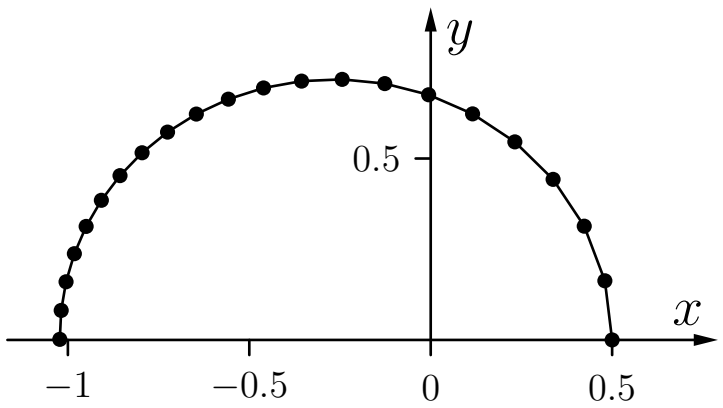
Rekurrente Netze: Umlaufbahnen der Planeten

$$\ddot{\vec{r}} = -\gamma m \frac{\vec{r}}{|\vec{r}|^3} \qquad \Rightarrow \qquad \dot{\vec{r}} = \vec{v} \qquad \dot{\vec{v}} = -\gamma m \frac{\vec{r}}{|\vec{r}|^3}$$

Rekursive Anpassungsregeln:

$$\begin{aligned} \vec{r}(t_i) &= \vec{r}(t_{i-1}) + \Delta t \; \vec{v}(t_{i-1}) \\ \vec{v}(t_i) &= \vec{v}(t_{i-1}) - \Delta t \; \gamma m \frac{\vec{r}(t_{i-1})}{|\vec{r}(t_{i-1})|^3} \end{aligned}$$





Rekurrente Netze: Trainieren der Systemparameter

Die bisherigen Rechnungen sind nur dann möglich, wenn man die Differentialgleichung *und* deren Parameterwerte kennt.

In der Praxis kennt man meist nur die Form der Differentialgleichung. Wenn Messdaten des Systems vorliegen, kann man versuchen, die Systemparameter durch Training eines rückgekoppelten neuronalen Netzes zu bestimmen.

Im Prinzip werden rückgekoppelte neuronale Netze genauso trainiert wie mehrschichtige Perzeptren. Einer direkten Anwendung des Fehler-Rückpropagationsverfahrens stehen jedoch die Rückkopplungen entgegen.

Lösung: Die Rückkopplungen werden durch eine *Ausfaltung* des Netzes *in der Zeit* zwischen zwei Trainingsmustern eliminiert (*Fehler-Rückpropagation in der Zeit*)

Rekurrente Netze: Fehler-Backpropagation über die Zeit

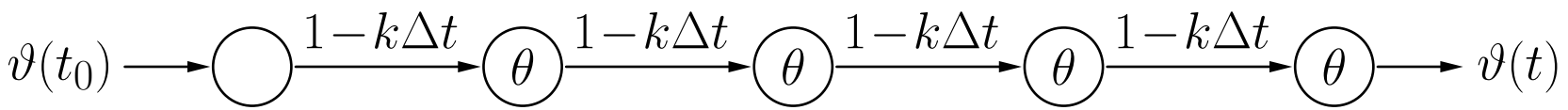
Beispiel: **Newton'sches Abkühlungsgesetz**

Annahme: Wir haben Messwerte der Abkühlung (oder Erwärmung) eines Körpers zu verschiedenen Zeitpunkten. Außerdem sei die Umgebungstemperatur ϑ_A bekannt.

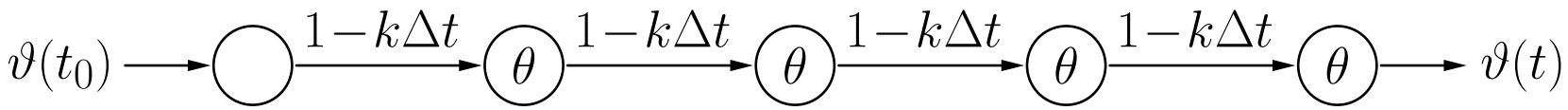
Ziel: Bestimmung des Werts der Abkühlungskonstanten k des Körpers.

Initialisierung wie bei einem MLP: Biaswert und Gewicht der Rückkopplung zufällig wählen.

Die Zeit zwischen zwei aufeinanderfolgenden Messwerten wird in Intervalle unterteilt. Damit wird die Rückkopplung des Netzes *ausgefaltet*. Liegen z.B. zwischen einem Messwert und dem folgenden vier Intervalle ($t_{j+1} = t_j + 4\Delta t$), dann erhalten wir



Rekurrente Netze: Fehler-Backpropagation über die Zeit



Für einen Messwert ϑ_j berechnet dieses Netz nun einen Näherungswert für ϑ_{j+1}.

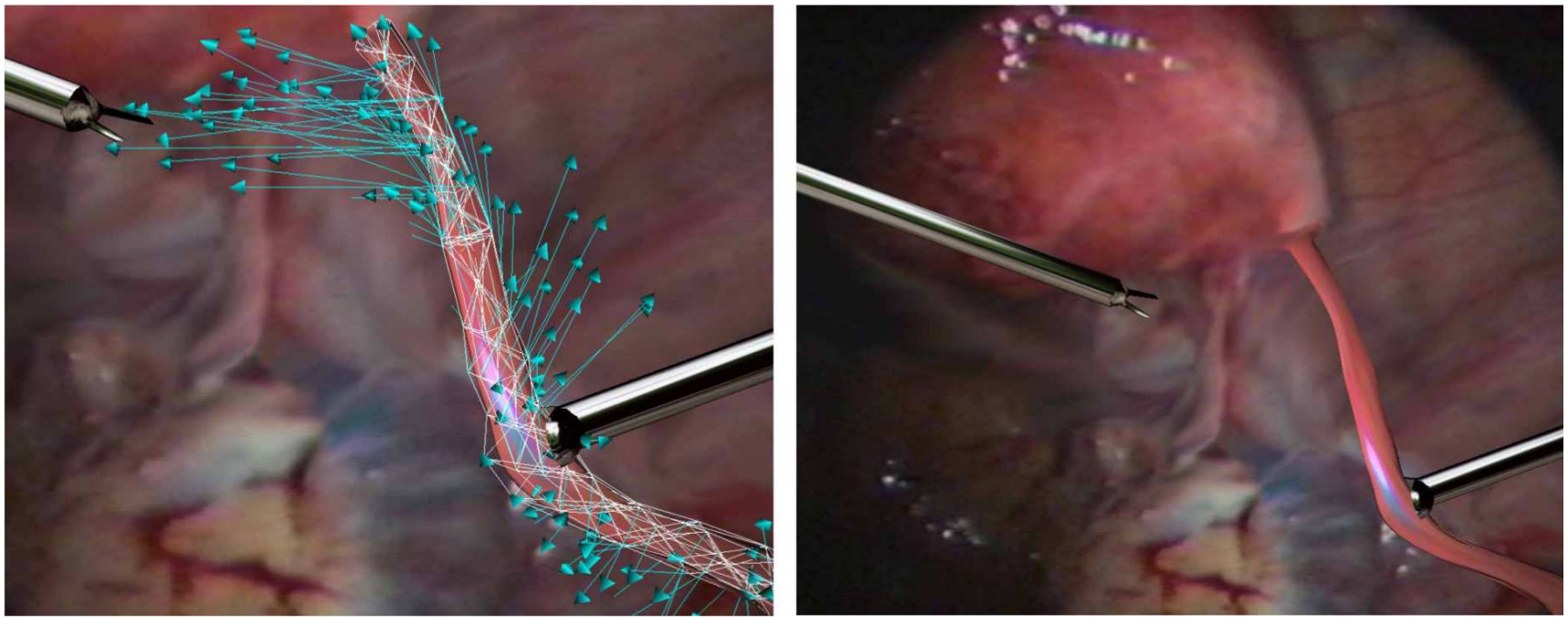
Durch Vergleich mit dem tatsächlichen Wert ϑ_{j+1} erhalten wir ein Fehlersignal, das mittels Fehler-Rückpropagation weitergegeben wird und zu Änderungen der Gewichte und Biaswerte führt.

Beachte: Das Netz besitzt nur ein Gewicht und einen Biaswert. Die berechneten Änderungen müssen daher aggregiert werden und dürfen erst nach Abschluss des Vorgangs zu einer Änderung des einen Verbindungsgewichtes und des einen Biaswertes führen.

Allgemein ist das Training rückgekoppelter neuronaler Netze dann sinnvoll, wenn die auftretenden Differentialgleichungen nicht mehr auf analytischem Wege gelöst werden können.

Anwendung: Virtuelle Laparoskopie

Beispiel: Bestimmung von Gewebeparametern für die virtuelle Chirurgie (genauer gesagt die virtuelle Laparoskopie) [Radetzky und Nürnberger; 2002]



links: Kraftvektoren der Gitterknoten, rechts: resultierende Deformation

Die hier auftretenden Systeme sind durch die hohe Zahl von Gleichungen viel zu komplex, um analytisch behandelt werden zu können Durch Training rückgekoppelter neuronaler Netze konnten jedoch recht beachtliche Erfolge erzielt werden.

Anwendung: Optimierung von Windparks

Quelle: Landesregierung Schleswig-Holstein

Eine Windkraftanlage:

Input aus Sensoren:

- Windgeschwindigkeit
- Vibration

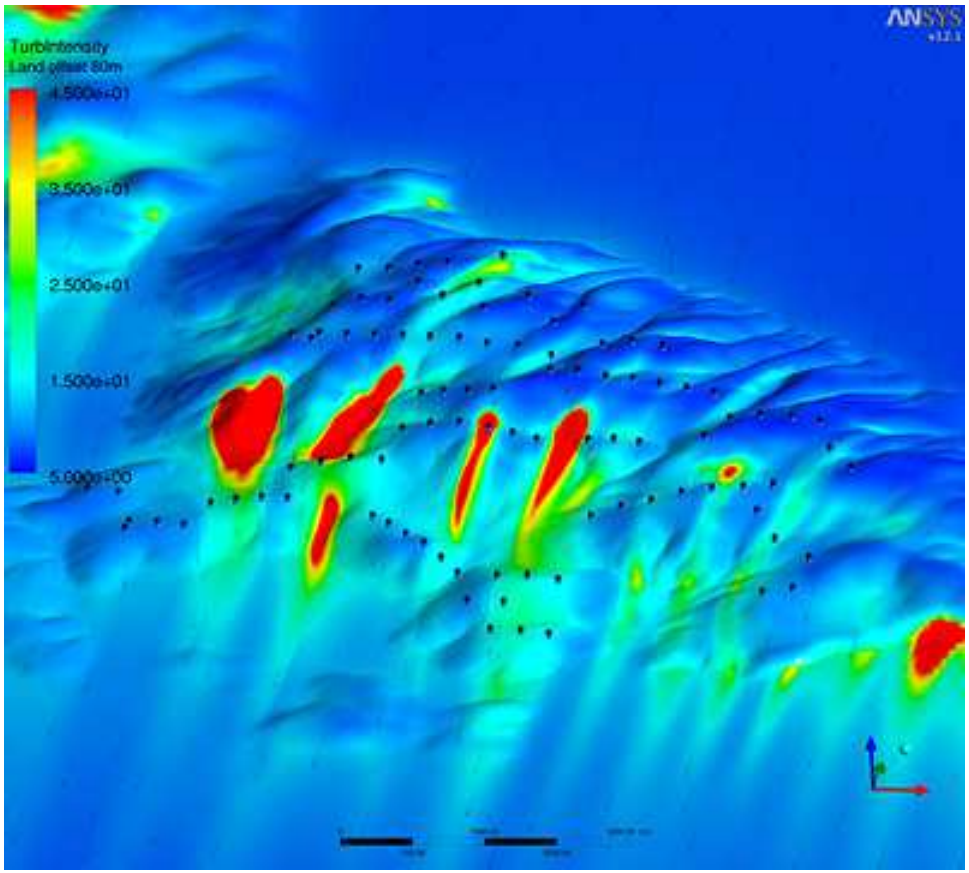
Variablen:

- Stellwinkel der Rotorblätter
- Einstellung des Generators

Ziele:

- Maximierung der Stromgenerierung
- Minimierung von Verschleiß

Anwendung: Optimierung von Windparks



Quelle: Siemens

Viele Windkraftanlagen:

Problem: Windräder verursachen Turbulenzen, die dahinter stehende Windräder beeinträchtigen

Input: Sensordaten aller Windräder

Variablen: Einstellungen aller Windräder

Ziele:

- Maximierung der Summe des erzeugten elektrischen Stroms
- Minimierung von Verschleiß bei allen Windkraftanlagen

Lösung: Verwende ein rekurrentes neuronales Netz