Recurrent Neural Networks

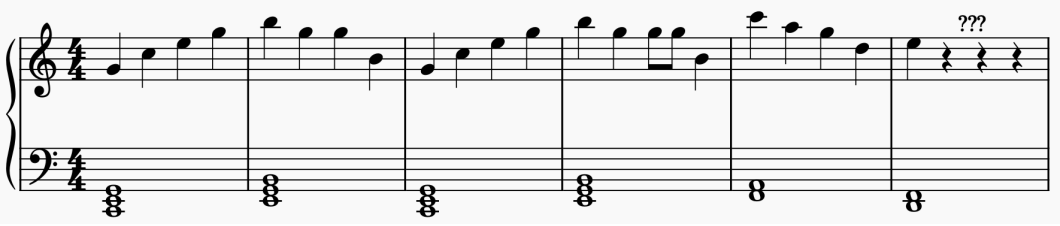
Benedikt Zönnchen

7. März 2023

Motivation

Motivation

Melodie-Generierung



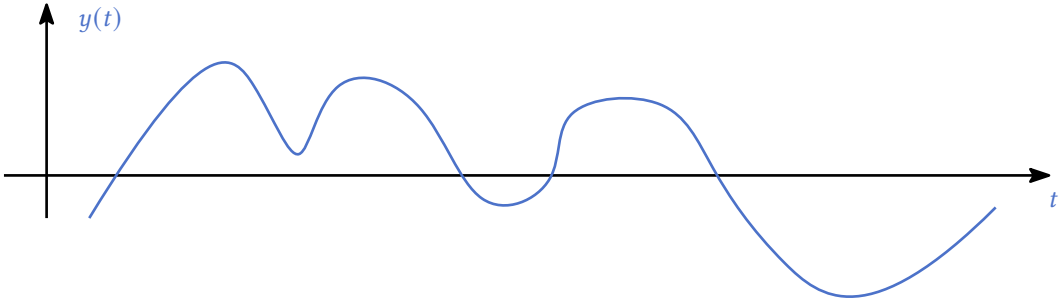
Motivation

Text-Generierung

"Die Wolken befinden sich im **[???]**"

Motivation

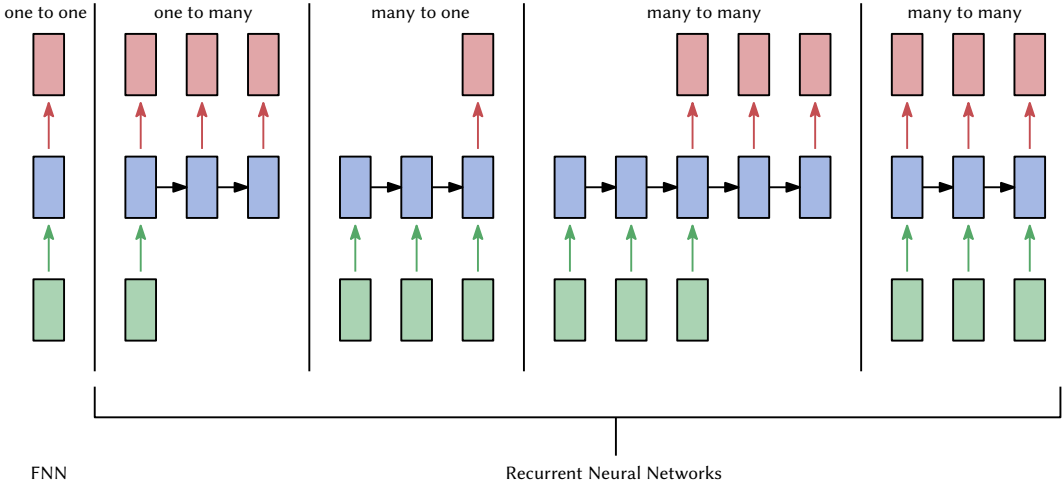
Zeitreihen allgemein



Viele Probleme lassen sich in ein sequenzielles Problem transformieren. Zum Beispiel:

Objekterkennung auf einem Bild durch Betrachtung unterschiedlicher Bereiche über die Zeit.

Motivation

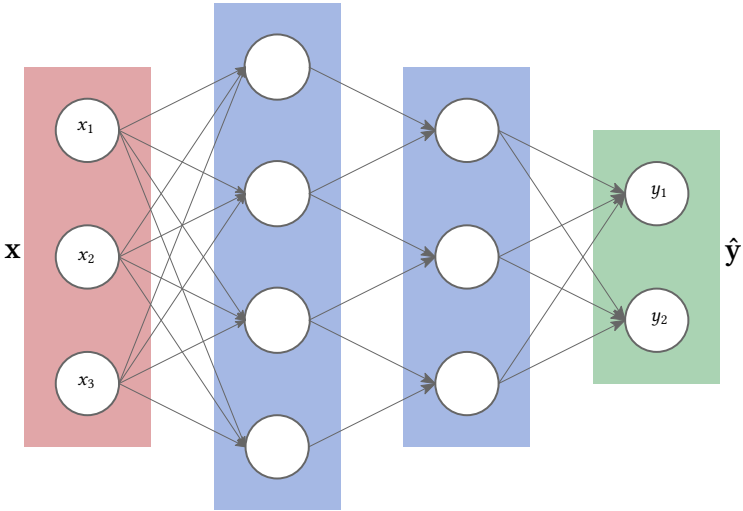

one to one
one to many
many to one
many to many
many to many
FNN
Recurrent Neural Networks

Von Feedforward zu Recurrent

Von Feedforward zu Recurrent

Die Eingabe durchläuft im Falle von Feedforward Neuronal Networks (FNN) das Netz von vorne nach hinten ohne Zyklus. Das Netz bildet einen gerichteten azyklischen Graphen.

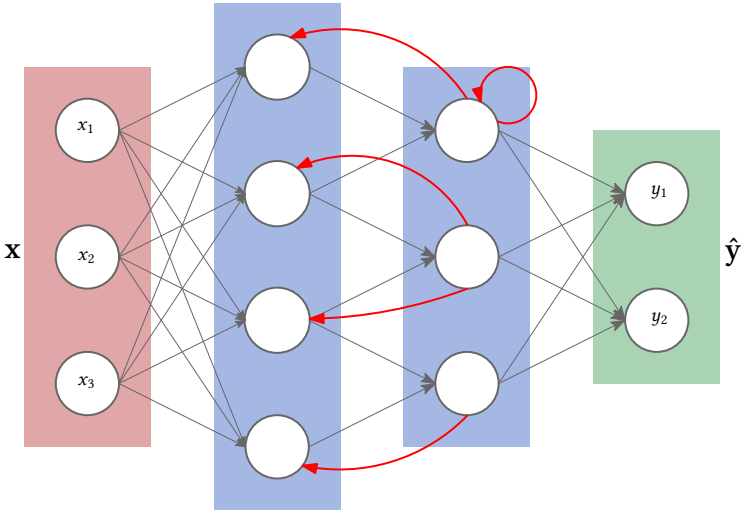
Von Feedforward zu Recurrent



Von Feedforward zu Recurrent

Führen wir Zyklen in das Netz ein, sprechen wir von einem sog. Recurrent Neural Network (RNN).

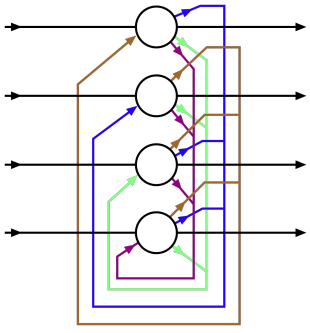
Von Feedforward zu Recurrent



Von Feedforward zu Recurrent

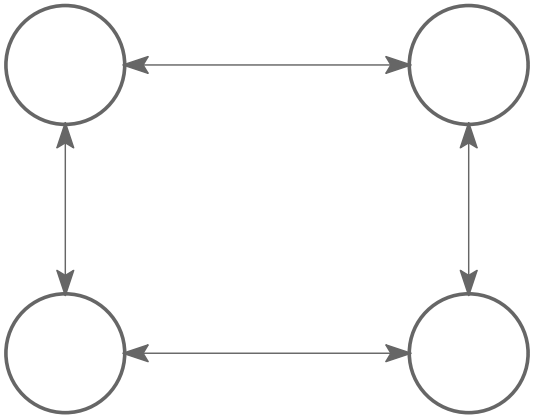
Hopfield Networks

- Erste Einführung durch John J. Hopfield [2] im Jahr 1982
- Motiviert durch:
  - Automatische Fehlerkorrektur
  - Informationsvervollständigung
  - Dynamische Systeme mit stabilen Zuständen (Beispiel: Planetensysteme)
- Fehlerhafte oder unvollständige Information 'konvergiert' zu korrekter Information (Assoziation)



Von Feedforward zu Recurrent

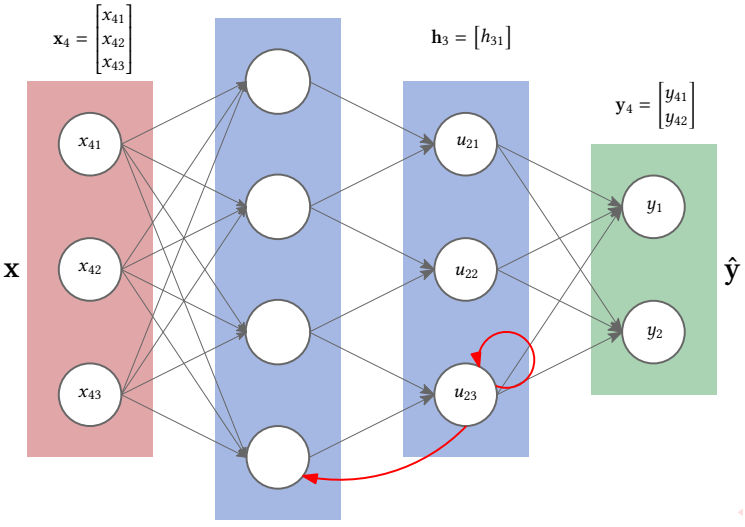
Hopfield Networks am Beispiel der Platzwahl



Von Feedforward zu Recurrent

Recurrent Neural Networks (RNN)

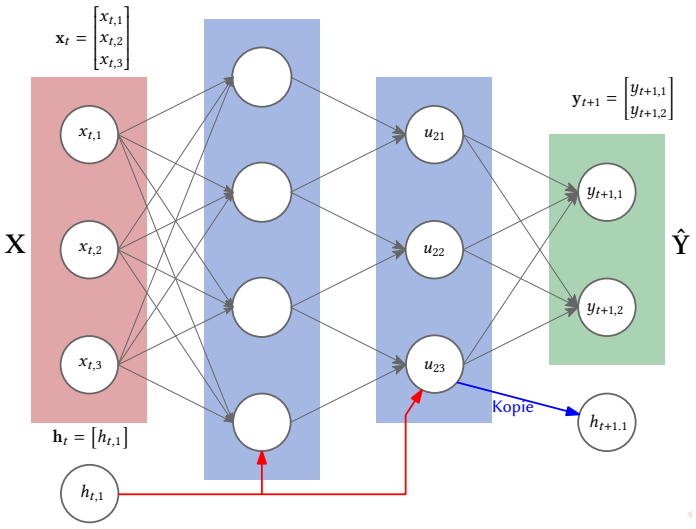
- Eingabe: Sequenz $\mathbf{X} = \mathbf{x}_0, \ldots, \mathbf{x}_n$
- Ausgabe: Sequenz: $\mathbf{Y} = \mathbf{y}_0, \ldots, \mathbf{y}_n$
- Hidden States: $\mathbf{H} = \mathbf{h}_0, \ldots, \mathbf{h}_n$



Von Feedforward zu Recurrent

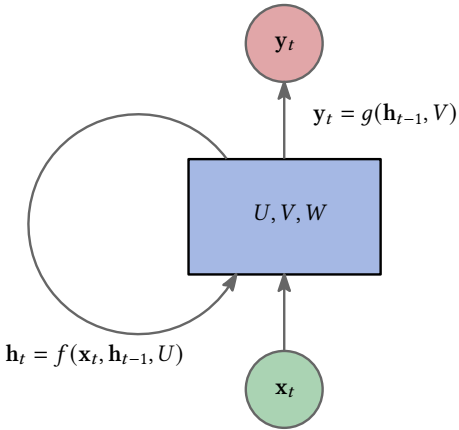
Recurrent Neural Networks (RNN)

- Eingabe: Sequenz $\mathbf{X} = \mathbf{x}_0, \ldots, \mathbf{x}_n$
- Ausgabe: Sequenz: $\mathbf{Y} = \mathbf{y}_0, \ldots, \mathbf{y}_n$
- Hidden States: $\mathbf{H} = \mathbf{h}_0, \ldots, \mathbf{h}_n$



Von Feedforward zu Recurrent

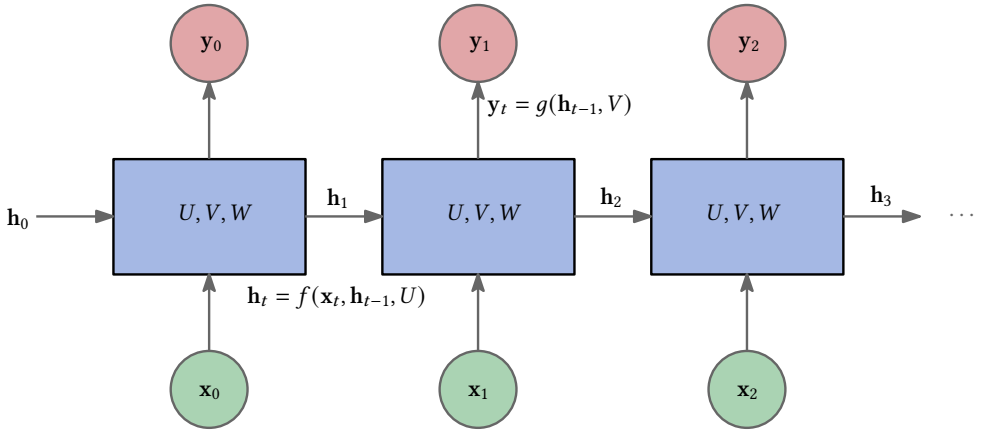
Recurrent Neural Networks (RNN)



Recurrent Neural Networks (RNN)

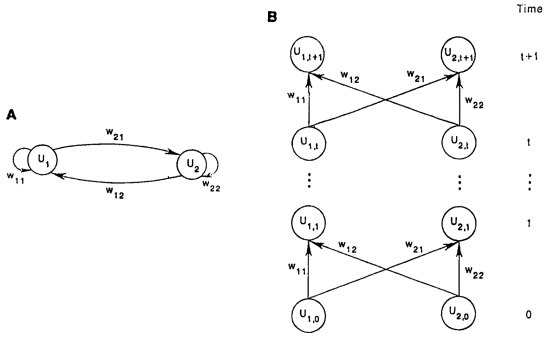
Recurrent Neural Networks (RNN)

Kompakte Darstellung



Recurrent Neural Networks (RNN)

Ausgebreitet

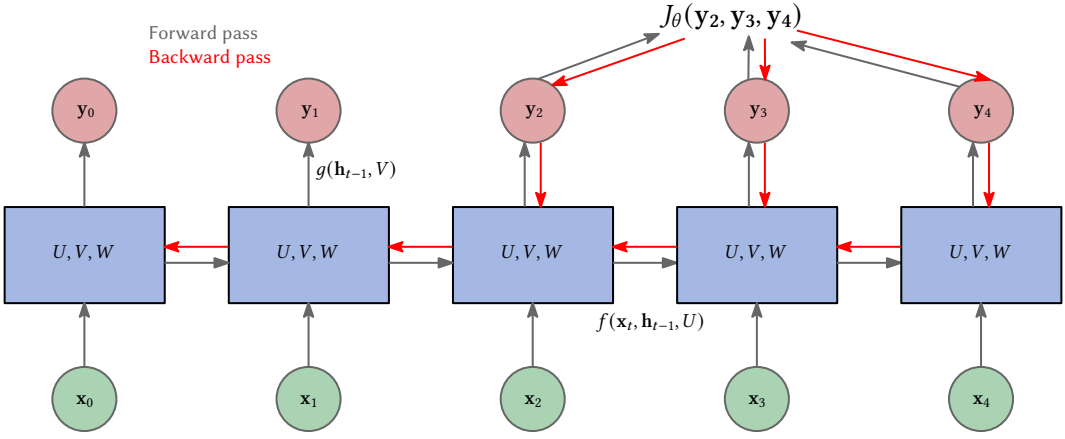


Theorem

Für jedes RNN gibt es ein Feedforward Network mit dem gleichen Verhalten über eine endliche Zeit [3].

Recurrent Neural Networks (RNN)

Back-Propagation Through Time (BPTT)



"If training vanilla neural nets is optimization over functions, training recurrent nets is optimization over programs." – *Andrej Karpathy*

$$\frac{\partial \mathbf{h}_t}{\partial W} = \frac{\partial f(\mathbf{x}_t, \mathbf{h}_{t-1}, W)}{\partial W} + \frac{\partial f(\mathbf{x}_t, \mathbf{h}_{t-1}, W)}{\partial \mathbf{h}_{t-1}} \cdot \frac{\partial \mathbf{h}_{t-1}}{\partial W}$$

$\Rightarrow$ **Exponentielle Abhängigkeit zwischen Fehler und Gewichte W**

$\Rightarrow$ **Gradienten tendieren zu explodieren oder zu verschwinden.**

$$\lim_{N \to \infty} (x \cdot w_{ij}^N) = \begin{cases} \infty & \text{für } w_{ij} > 1.0 \\ 0 & \text{für } w_{ij} < 1.0 \end{cases}$$

Recurrent Neural Networks (RNN)

RNNs in ihrer ursprünglichen Form werden kaum noch benutzt. Stattdessen setzt man auf Long Short-term Memory RNNs kurz LSTMs.

Recurrent Neural Networks (RNN)

Vorteile:

+ Variable Eingabegröße
+ Modellgröße explodiert nicht bei längerer Eingabe
+ Berechnung nimmt historische Informationen auf
+ Gewichte werden über die Zeit geteilt

Nachteile:

- Langsame (sequenzielle) Berechnung
- Informationen die lange zurück liegen gehen verloren
- Keine Möglichkeit zukünftige Eingabe in die derzeitige Berechnung einzubinden

Long Short-term Memory

Long Short-term Memory

"Die Wolken befinden sich im **[???]**"

"Ich bin in Frankreich aufgewachsen [...] Ich spreche **[???]**"

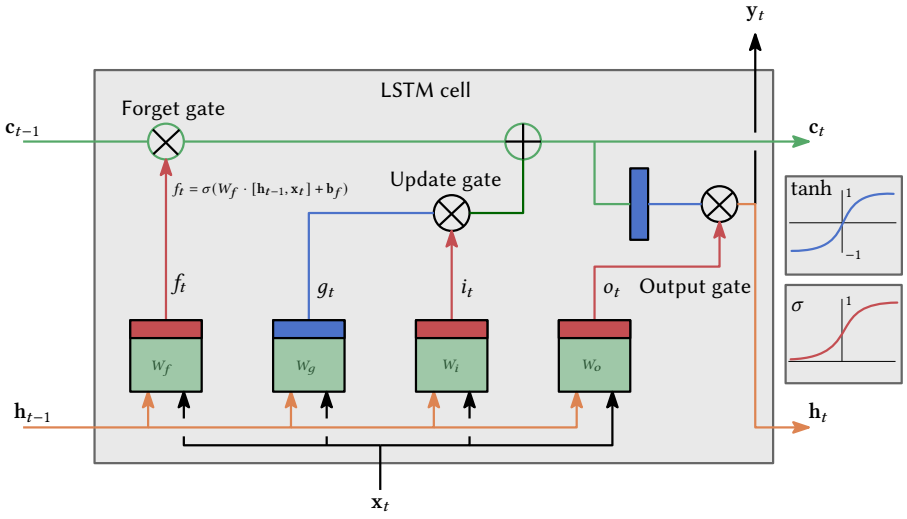
"The nurse notified the patient that his shift would be ending in an hour."

Long Short-term Memory

Probleme:
Um zu lernen Informationen über einen längeren Zeitraum zu Speichern braucht es wegen der explodierenden oder verschwindenden Fehler in der recurrent backpropagation zu viel Ressourcen/Rechenzeit [1].

Lösungsidee:
Schaffe einen direkten Weg durch das abgerollte Netzwerk.

Long Short-term Memory



Long Short-term Memory

Intuition durch Gleichungen

Filter:

$$y(t) = w_1 \cdot x(t) + w_2 \cdot y(t-1)$$

"RNNs":

$$y(t) = w_1 \cdot x(t) + w_2 \cdot m(t-1)$$
$$m(t) = w_3 \cdot x(t) + w_4 \cdot m(t-1)$$

"LSTMs":

$$y(t) = w_1 \cdot x(t) + w_2 \cdot m(t-1)$$
$$m(t) = u(t) \cdot (w_3 \cdot x(t) + w_4 \cdot m(t-1)) + (1 - u(t)) \cdot m(t-1)$$
$$u(t) = \sigma(w_5 \cdot x(t) + w_6 \cdot m(t-1))$$

Long Short-term Memory

Vorteile (gegenüber Vanilla RNN):

+ Kommt mit längeren Abhängigkeiten klar
+ Stabilerer "Speicher"
+ Additiver statt Multiplikativer Fehler beim BPTT

Nachteile (gegenüber Vanilla RNN):

- Längeres Training mit mehr Ressourcen
- Anfällig für Überanpassung

Die sequenzielle Struktur, d.h. der **Informationsfluss durch die Zeit**, bleibt erhalten!

$\Rightarrow$ Dennoch akkumuliert sich der Fehler beim BPTT und die Information wird (wenn auch viel weniger) mit der Zeit verwischt.

Vorschau

Können wir diese Struktur weiter aufweichen? Können wir lernen auf genau die Information zuzugreifen, wenn wir sie brauchen? Können wir **Aufmerksamkeit** lernen?

Referenzen I

[1] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[2] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

[3] Rumelhart, D. E. and McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*, pages 318–362.