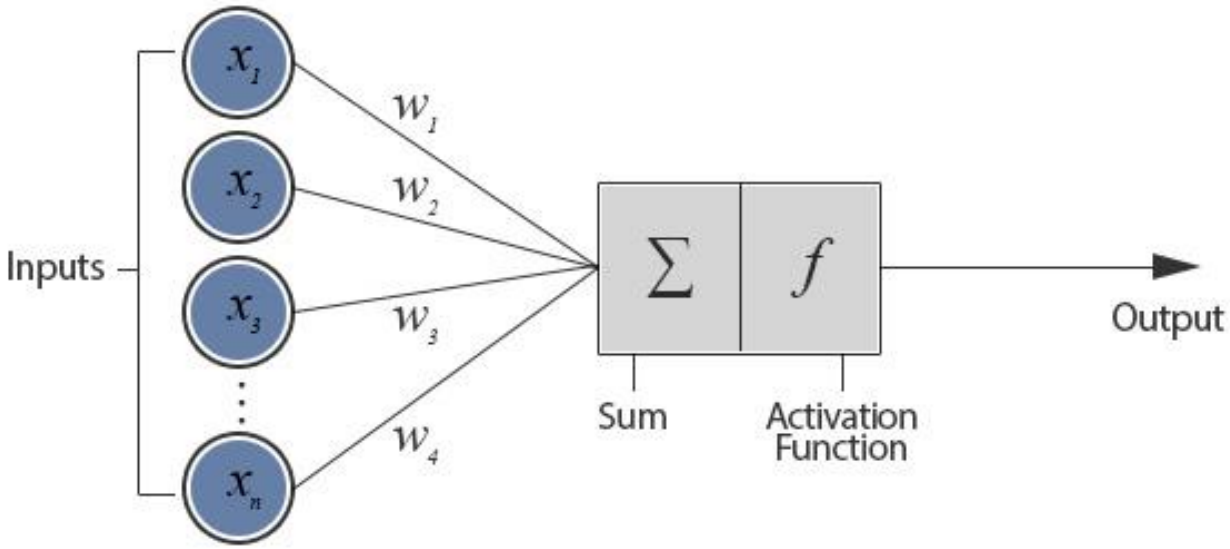
Simulated Annealing für Künstliche Neuronale Netze

Gabriel Gavrilas

Künstliche Neuronale Netze

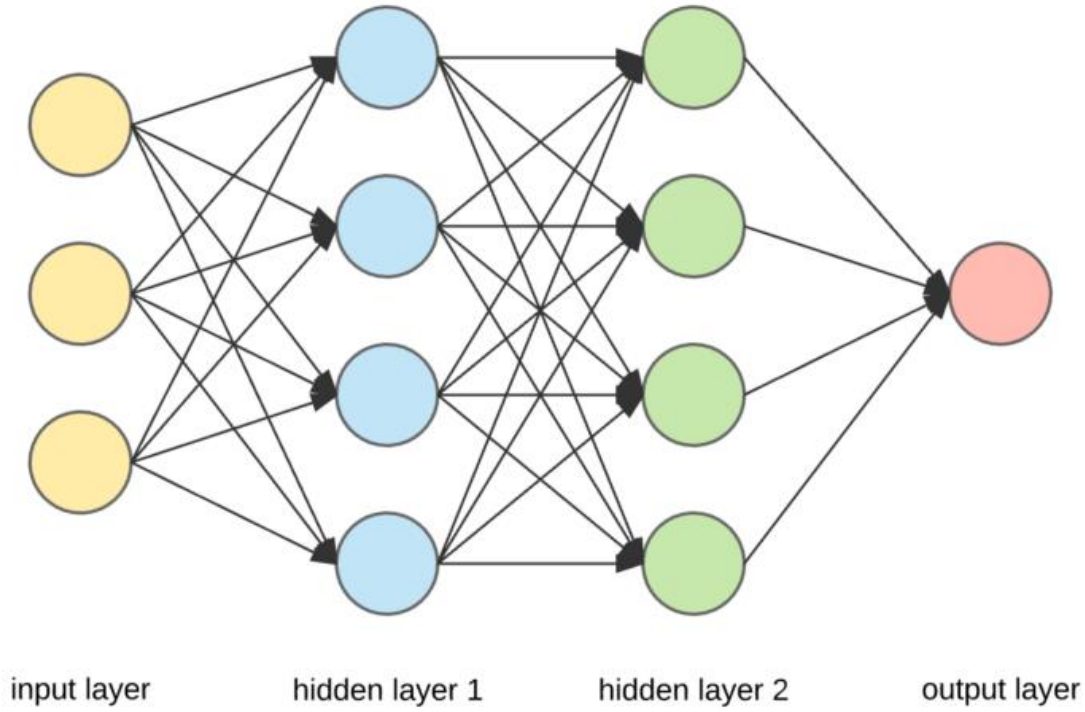
Aufbau Neuronaler Netzwerke

- Grundlegende Struktur: Neuron
- Output = $f\left(\sum_i w_i x_i\right)$
- Feedforward
- «Lernen» durch Optimierung der Gewichte.



Aufbau Neuronaler Netzwerke

- Komposition vieler Neuronen in Netz
- Verschiedene «layers» mit mehreren Neuronen
- «layers» vollständig verbunden untereinander
- «Lernen» wiederum durch Optimierung der Gewichte



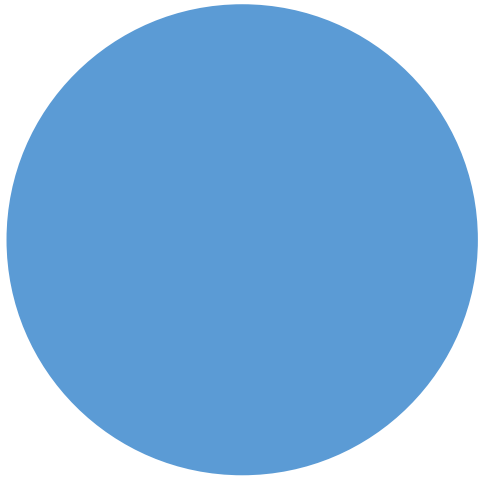
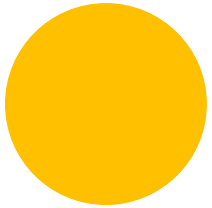
Gradient Descent

- Standard Verfahren für Neurale Netze
- Optimierung mithilfe der Gradienten
- Backpropagation
- Probleme
  - Vanishing gradients
  - Lokale Minima

Mein Projekt

Mein Projekt

- Implementation eines Neuronalen Netzes mit Simulated Annealing als Optimierungsverfahren
- In Pyhton mit Jupyther Notebooks
- Implementationsverlauf:
    1. Struktur Neuronales Netzwerk & Datengenerierung
    2. Implementation GD Versionen
    3. Implementation SA Versionen
    4. Parametrisiertes SA

Simulated Annealing

Simulated Annealing

- Let $s = s_0$
- For $k = 0$ through $k_{\max}$ (exclusive):
  - $T \leftarrow \text{temperature}(\ k_{\max}/(k+1)\ )$
  - Pick a random neighbour, $s_{\text{new}} \leftarrow \text{neighbour}(s)$
  - If $P(E(s), E(s_{\text{new}}), T) \geq \text{random}(0, 1)$:
    - $s \leftarrow s_{\text{new}}$
- Output: the final state s

Simulated Annealing für Neuronale Netze

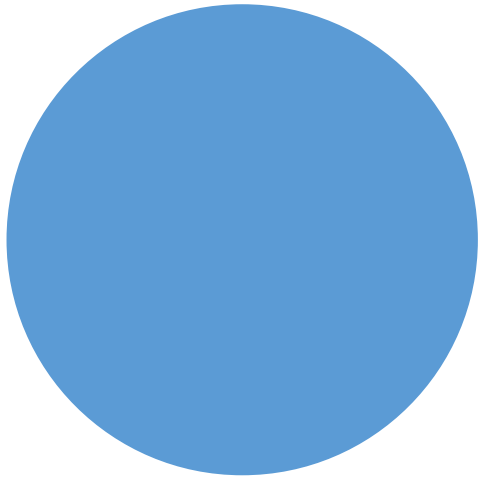
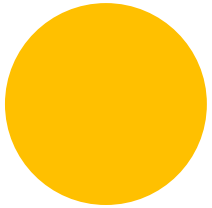
1. Zustandsraum & Startzustand s_0:
   - Zustandsraum: Werte der Gewichte
   - Startzustand: Zufällige Werte zwischen -1 und 1
2. Energy function:
   - Loss function des Neuronalen Netzes
   - Squared loss, absolute loss
   - Regularisierung: L1, L2

Simulated Annealing für Neuronale Netze

3. Kandidaten generierende Funktion $neighbour(s)$
   - Zufälliger kleiner Wert zu Gewichten addiert
   - Abhängig von der Temperatur
4. Annahmewahrscheinlichkeit $P(e, e_{new}, t)$:
   - Falls $e_{new} < e$: 1
   - Sonst mehrere Möglichkeiten:
     - Konstant: 0
     - Linear: t
     - Boltzmann: $\exp(-\frac{e_{new} - e}{t})$
     - Boltzmann: $(1 + \mathrm{i})^{\left(-\frac{e_{new} - e}{t}\right)}$

Simulated Annealing für Neuronale Netze

5. Abkühlungsprogramm: $temperature()$ & Starttemperatur
   - $temperature()$: Stufenweise linear, linear, exponentiel
   - Starttemperatur: 1, von Funktion abhängig
6. Abbruchbedingung
   - Max. Anzahl Iterationen
7. Rückgabe
   - Letzter Zustand
   - Bester Zustand

Kommentare

Vergelich SA-GD

- Keine Gradienten
- Globale maxima
- Aber:
  - Keine Zeitgarantien
  - GD kann verbessert werden -> SGD

Andere Methoden

- Stochastic Gradient descent SGD:
  - Approximieren des Gradienten
  - Lernrate
  - Verhält sich wie SA: https://leon.bottou.org/publications/pdf/nimes-1991.pdf
- EA, GA

Demo

Ende

Vielen Dank!