

Eine Anwendung des Reinforcement Learning zur Regelung dynamischer Systeme

Aktueller Status

Jonas Helmut Wilinski

Statusgespräch, Mittwoch 18. Juli 2018

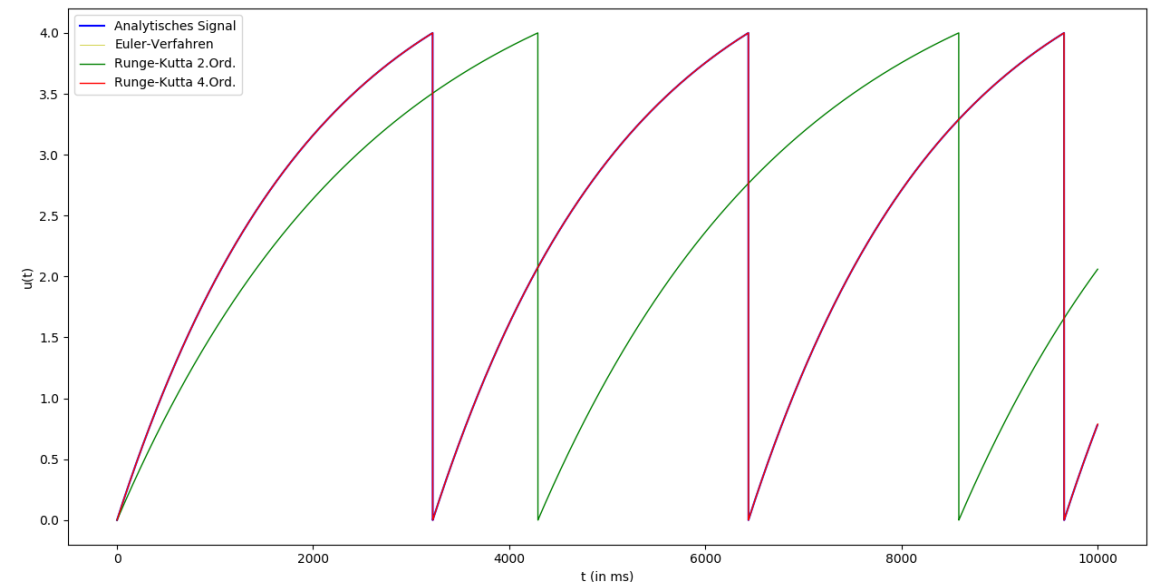
Literaturphase

- Mehrere Paper & Fachliteratur gelesen und zusammengefasst
 - Bücher:
 - Stuart Russell, Peter Norvig - Artificial Intelligence - A Modern Approach (2010, Prentice Hall)
 - Raúl Rojas (auth.) - Theorie der neuronalen Netze - Eine systematische Einführung (1993, Springer-Verlag Berlin Heidelberg)
 - Steven H. Strogatz - Nonlinear Dynamics and Chaos - With Applications to Physics, Biology, Chemistry, and Engineering (1994, Westview Press)
 - **Wulfram Gerstner, Werner M. Kistler, Richard Naud, Liam Paninski - Neuronal Dynamics - From Single Neurons to Networks and Models of Cognition (2014, Cambridge University Press)**
 - Fachartikel:
 - Lechner (et al) - Worm-level control through search-based reinforcement learning
 - Lechner (et al) - Neuronal Circuit Policies
 - SIM-CE - An advanced Simulink platform for studying the brain of C. elegans
 - ...
 - Kurse:
 - Reinforcement Learning by David Silver (Google DeepMind – UCL)

Implementierung des Leaky Integrate and Fire - Modells

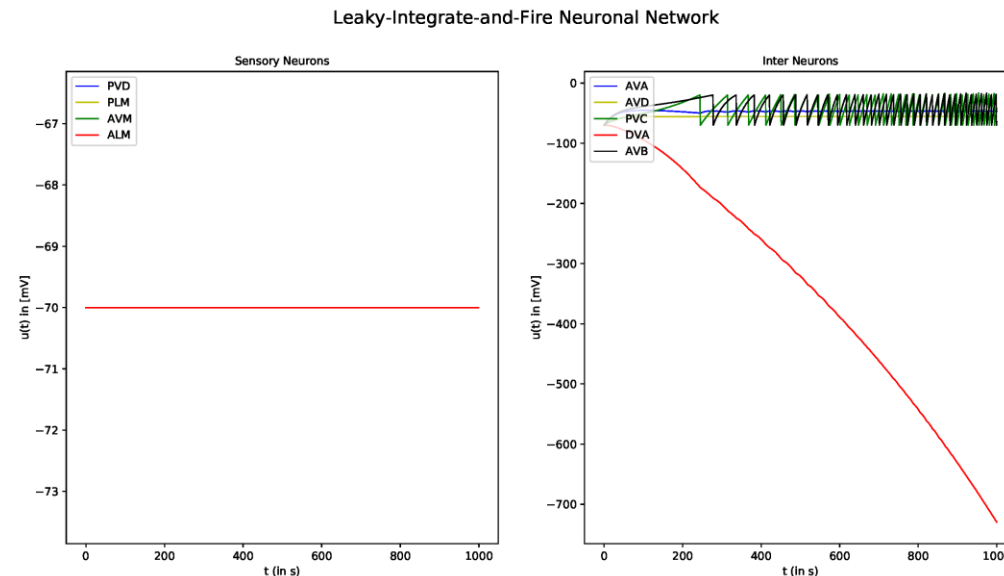
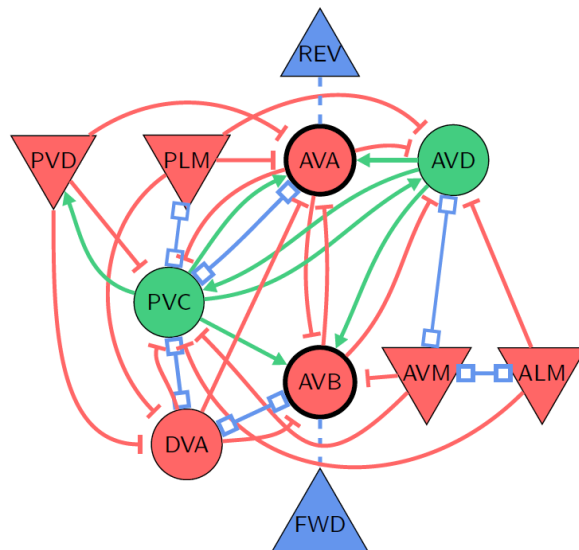
- Programmiersprache: Python
- Zusätzliche Libraries:
 - Mathematische Berechnungen & Matrizen durch die Python-Library NumPy
 - Darstellung durch die Python-Library Matplotlib
- Leaky Integrate and Fire - Modell:
 - Lösung der Differentialgleichung durch numerische Verfahren:
 - Euler-Verfahren
 - **Runge-Kutta 2. & 4. Ordnung**

Simple "Leaky-Integrate-and-Fire Model"



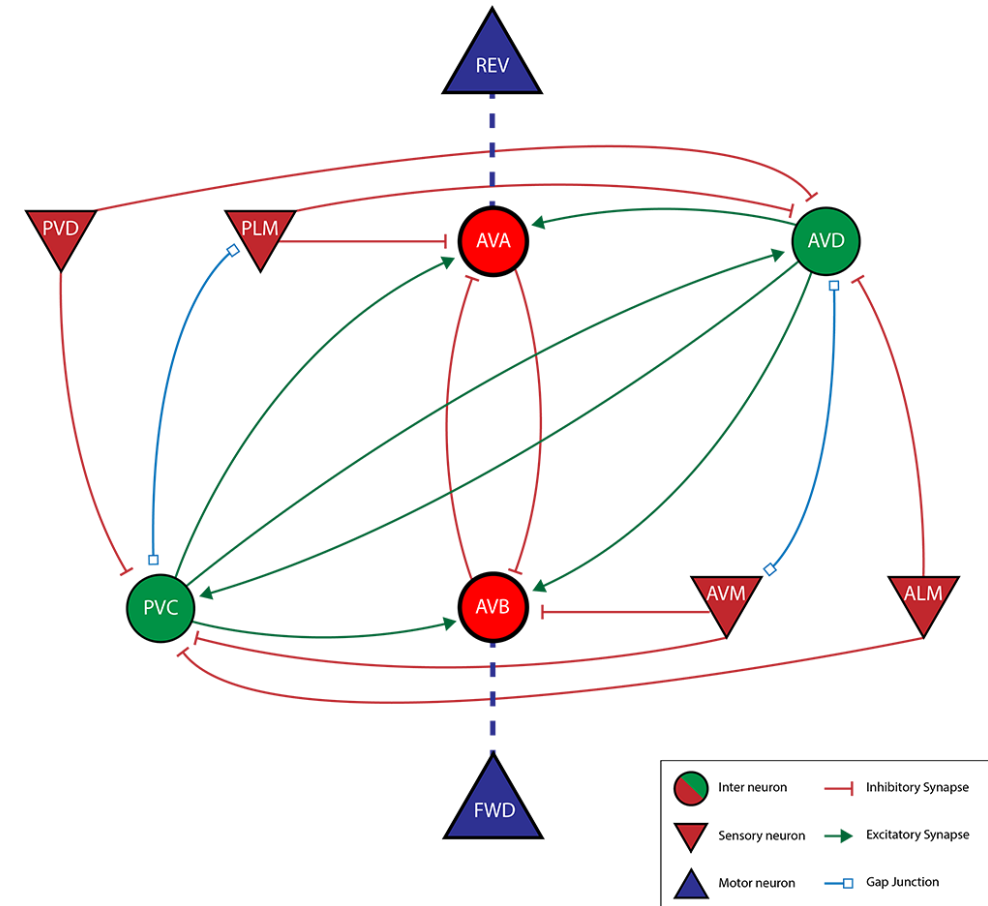
Implementierung des Neuronalen Netzes

- Implementierung durch Nutzung von Transitionsmatrizen A, A_{Gap}, B, B_{Gap} , um die Verbindungen zwischen Neuronen darzustellen
- Parameter $U_{leak}, w, \sigma, C_m, G_{leak}$ sind ebenfalls anhand der Transitionsmatrizen angeordnet
- Das `compute`-Modul berechnet durch die Modellgleichungen die Ströme der Synapsen bzw. Gap-Junctions und folglich die Membranpotentiale der Neuronen
- Dadurch kommt es zu den fire-Ereignissen und die Motor-Neuronen werden angeregt



Implementierung des Neuronalen Netzes mit symmetrischen Komponenten

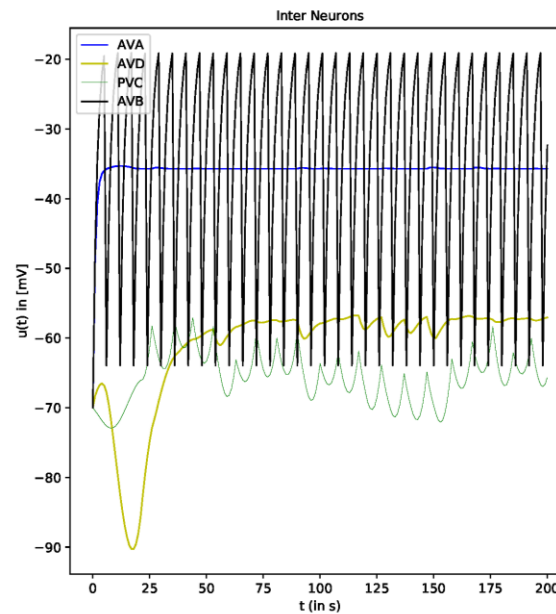
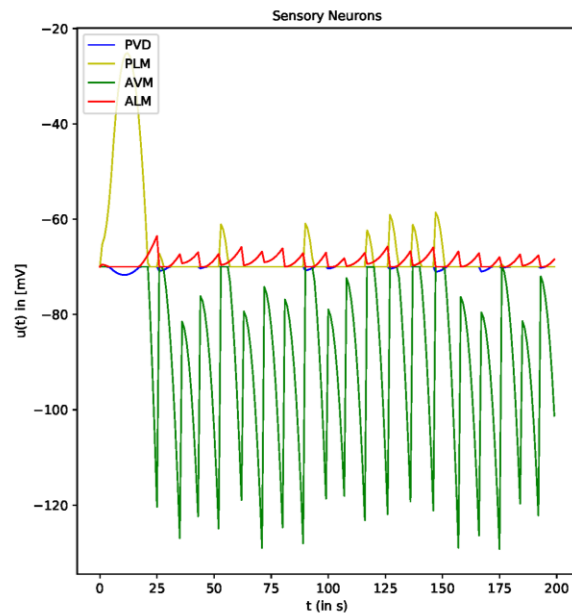
- Aufgrund unsymmetrischen Verhalten der Neuronen wurde das bestehende neuronale Netz leicht verändert und symmetrisch aufgestellt
- Transitionsmatrizen A , A_{Gap} , B , B_{Gap} wurden entsprechend angepasst
- Parameter U_{leak} , w , σ , C_m , G_{leak} werden nun via Random Search erzeugt
- Eine Rückführung der Observation von Winkel φ des Pendels und Geschwindigkeit v des Carts in die Eingangsneuronen bildet das geschlossene Simulationsmodell
- Durch 10.000 Episoden werden mittels Reinforcement Learning die besten Parametermatrizen herausgefiltert



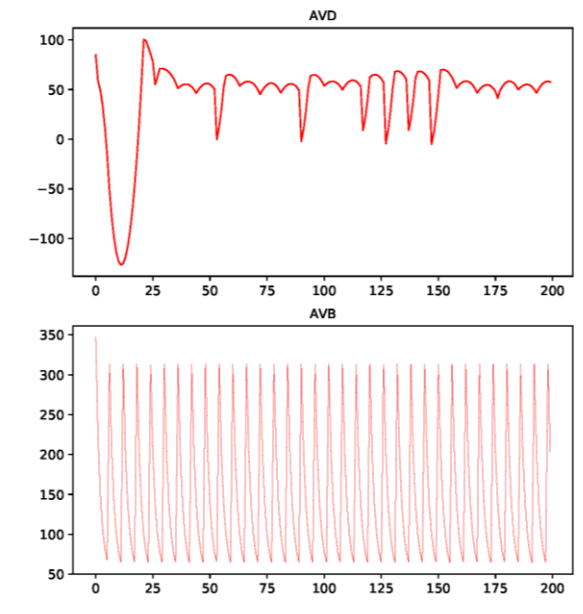
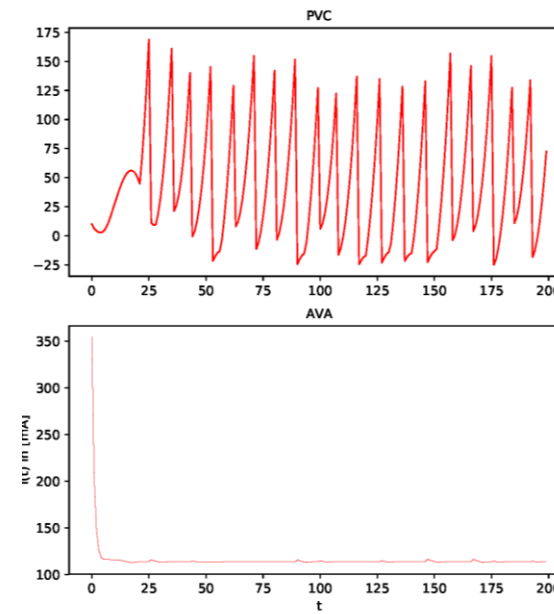
Implementierung des Neuronalen Netzes mit symmetrischen Komponenten

Feste Parameter - Simulation der CartPole-Environment

Leaky-Integrate-and-Fire Neuronal Network



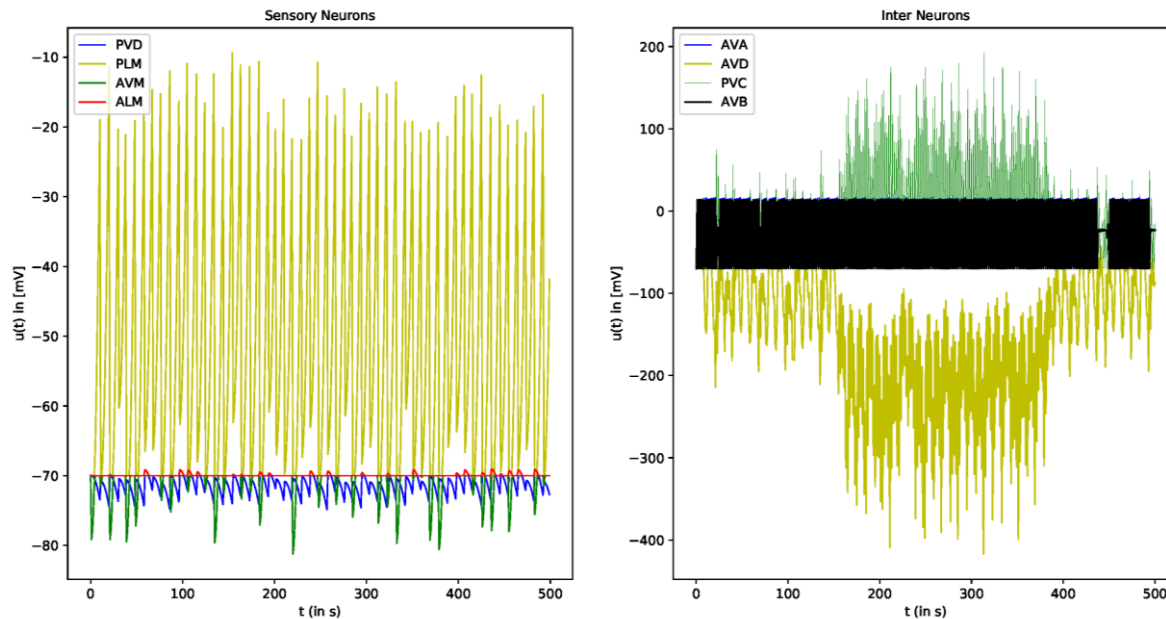
Neuron Currents



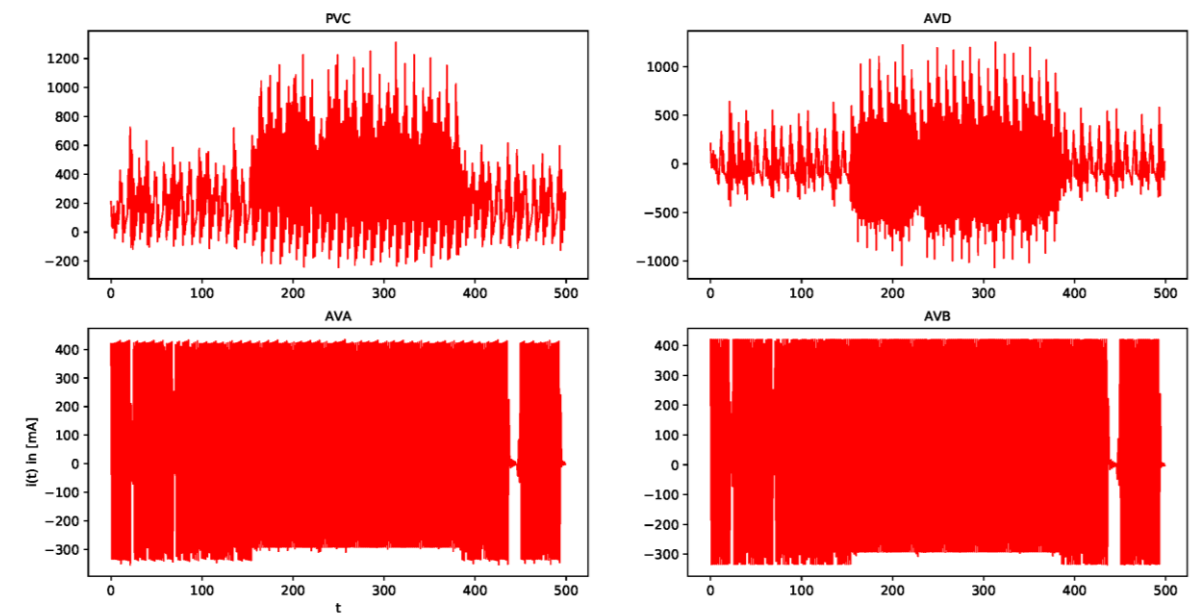
Implementierung des Neuronalen Netzes mit symmetrischen Komponenten

Dynamische Parameter - Simulation der CartPole-Environment

Leaky-Integrate-and-Fire Neuronal Network

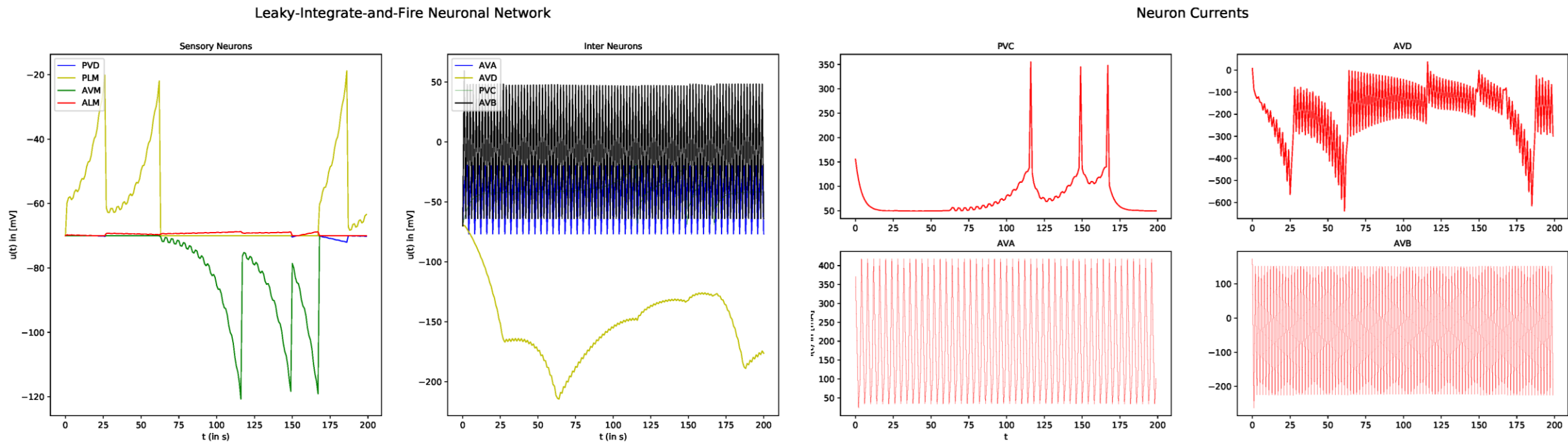


Neuron Currents



Implementierung des Neuronalen Netzes mit symmetrischen Komponenten

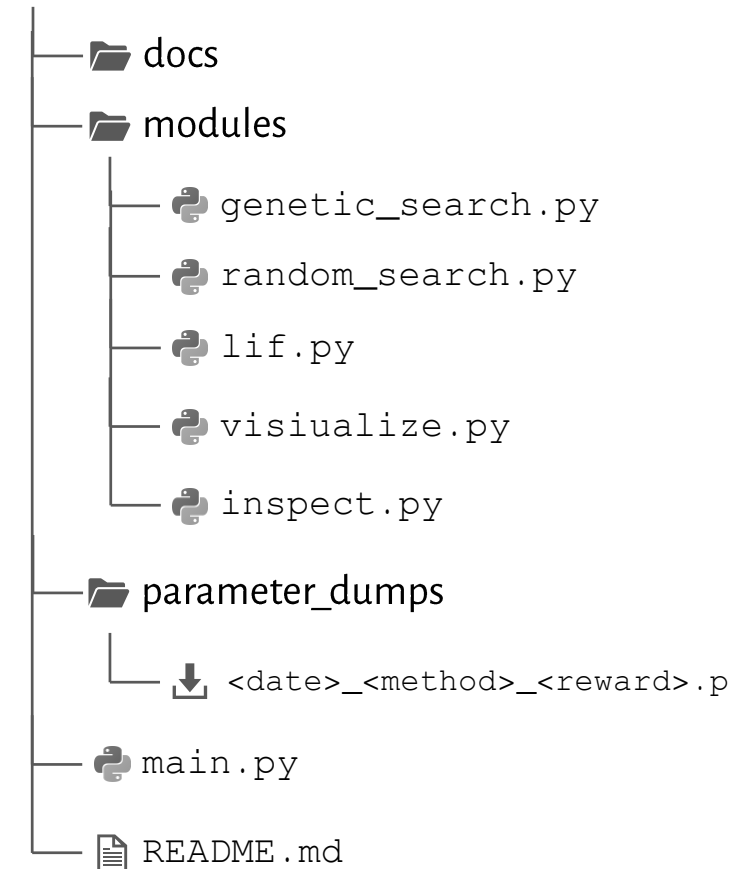
- Der bisher beste Parametersatz hat einen Score in der OpenAI Gym Library von 31/200
- Dieser Score ist nicht sehr gut – jedoch wurde er lediglich durch Random Search herausgefunden:



Erweiterung der Lernmethode

- Durch RandomSearch werden zufällige Werte generiert – es liegt kein tieferer Sinn hinter dieser Methode
- State of the Art Algorithmen nutzen verschiedene Methoden, um gezielter zu suchen bzw. zu lernen:
 - Gradient-Based Methoden schauen nach der Tendenz der Parameter und suchen in eine gezielte Richtung
 - Genetische Algorithmen nutzen die gut verstandene Evolutionstheorie um Populationen zu bilden und diese durch Fit und Mutation in eine Richtung zu lernen
 - Kostenfunktionen
- Geplant ist, den RandomSearch Score mit diesen Algorithmen zu vergleichen und einen besseren Score zu erzielen
 - Dies erfordert eine erweiterte und umfangreiche Programmierung in Python (da dieser Ansatz des Reinforcement Learning) keine Toolboxes hat

BA Repository





**Lehrstuhl für Regelungstechnik
Technische Fakultät der CAU Kiel**



Programme und Code
auf GitHub
(JOnasW/BA)



Environment:
OpenAI Gym
(CartPole_v0)



Programmiersprache:
Python 2.7
(Atom Editor)