UNIVERSITÄT ZU LÜBECK

Bachelorprojekt

Optimierung biologisch-realistischer Neuronenmodelle

Institut für Robotik und Kognitive Systeme

Dokumentation

Szymon Bereziak
Moritz Dannehl
Chris Girth
Can Kalelioglu
Julian Wolff

15. September 2015

Inhaltsverzeichnis

1 Einführung

1.1 Ziel des Projektes

Mithilfe von künstlichen neuronalen Netzen wird versucht, Strukturen des menschlichen Gehirns zu simulieren. Die verwendeten Simulationen basieren dabei auf sehr vielen Parametern, deren Optimierung eine sehr langwierige Aufgabe darstellt. Es wurde im Rahmen dieses Projektes ein Framework erstellt, welches als Schnittstelle zwischen solchen Simulationen und Optimierungsalgorithmen fungiert. Dabei waren vor allem eine Modularisierung zwecks Erweiterbarkeit sowie eine einfache Bedienbarkeit über eine grafische Oberfläche Zentrum der Entwicklung.

2 Features

- einfache Anbindung eines (neuen) Frontends möglich durch ein simples Nachrichtenprotokoll
- einfaches Hinzufügen von neuen Algorithmen
- Modularisierung erlaubt komplettes Austauschen einzelner Module ohne die Kernlogik zu kennen
- verteiltes Arbeiten möglich durch Anbindung des Simulationsclusters über SSH oder andere Protokolle
- Multithreading bereits nativ in Kernlogik implementiert; Kern verwaltet verschiedene Läufe von Algorithmen selbstständig
- kein aktives Warten notwendig; bei Terminierung eines Algorithmus' wird das Frontend automatisch benachrichtigt

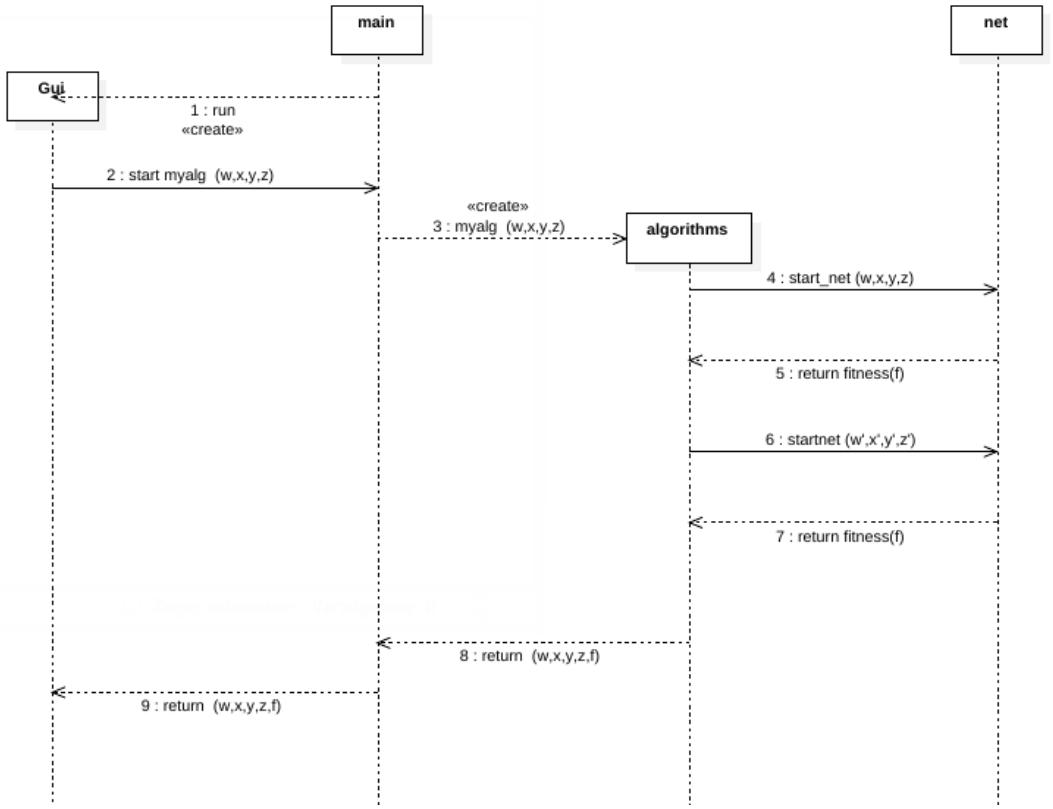
3 Architektur

Das Framework ist in zwei Teile gegliedert. Der *Kern* verwaltet selbstständig alle Algorithmen und spricht die Simulation an. Die Kommunikation zwischen dem Kern und der *Benutzerschnittstelle* erfolgt über eine nachrichtenbasierte *Kommunikationsschnittstelle*. Die Benutzerschnittstelle kann eine grafische Benutzeroberfläche oder ein Kommandozeileninterface sein. Über die Kommunikationsschnittstelle werden dem Kern Befehle erteilt, die das Einladen von Konfigurationen, Setzen von Parametern, Starten und Stoppen von Algorithmen sowie Statusabfragen umfassen.

3.1 Kern

Der Kern besteht aus den folgenden Komponenten:
main. Verarbeitet eingehende Befehle von der Benutzerschnittstelle.

net. Führt über eine SSH-Verbindung ein neuronales Netz und die zugehörige Analyse aus.
algorithms. Stellt Optimierungsalgorithmen bereit.



3.2 GUI

Die GUI besteht aus den folgenden Komponenten:
mainframe. Dient dem Starten, Stoppen, Überwachen und Speichern der Berechnungen.
addframe. Ermöglicht die Auswahl der Optimierungsalgorithmen und das Setzen der Startparameter.
sshframe. Oberfläche zur Eingabe der SSH-Daten.

4 Benutzerhandbuch

4.1 Ausführen der Anwendung

Zur Ausführung sind die Pakete `python3` und `sshpass` notwendig.
Die Anwendung kann als Kommandozeilenanwendung mithilfe des Befehls `python3 main.py` oder alternativ mit grafischer Oberfläche mithilfe des Befehls `python3 main.py --gui` ausgeführt werden.

4.2 CLI

Wird die Anwendung als Kommandozeilenanwendung gestartet, so kann sie mittels der folgenden Befehle gesteuert werden:

Befehl	Auswirkung
`help`	Liefert eine Liste möglicher Befehle, ähnlich dieser Tabelle
`get algorithms`	liefert eine Liste von implementierten Algorithmen
`get algorithms <name>`	liefert die möglichen Parameter für den angegebenen Algorithmus
`get config`	liefert alle in der Konfigurationsdatei vorhandenen Sektionen
`get config <sec>`	liefert alle Optionen unter der gegebenen Sektion
`get config <sec> <opt>`	liefert den Wert der angegebenen Option
`set config <sec> <opt> <val>`	Setzt die angegebene Option auf den angegebenen Wert
`set password`	Öffnet eine Passworteingabe zur Eingabe des Passworts der SSH-Verbindung zum neuronalen Netz
`save config`	Speichert die Änderungen in der Konfigurationsdatei
`start <algorithm> <params...>`	Startet einen Optimierungsalgorithmus mit den gegebenen Parametern

Bevor eine Optimierung gestartet wird sollten folgende Befehle aufgerufen werden:

Befehl	Auswirkung
`set password`	Setzt das SSH Passwort
`set config SSH host <user@ip>`	Setzt die Serveradresse
`set config SSH net <cmd>`	Setzt den Befehl zur Ausführung des neuronalen Netzes
`set config SSH analysis <cmd>`	Setzt den Befehl zur Ausführung der Analyse

Anschließend können beliebige Optimierungsalgorithmen gestartet werden.

Beispiel:

```
>set config SSH host "bachelor1@localhost"
>set config SSH net "cd ~/acnet2 && genesis acnet2.g"
>set config SSH analysis "cd ~/acnet2 && python ./analysis.py"
>save config
>set password
>start random_search 4
```

4.3 GUI

Wird das Programm mittels des Befehls

```
python3 main.py --gui
```

gestartet, öffnet sich das Hauptfenster der grafischen Benutzeroberfläche. Abbildung 1 zeigt schematisch die Möglichkeiten, welche die GUI bietet:

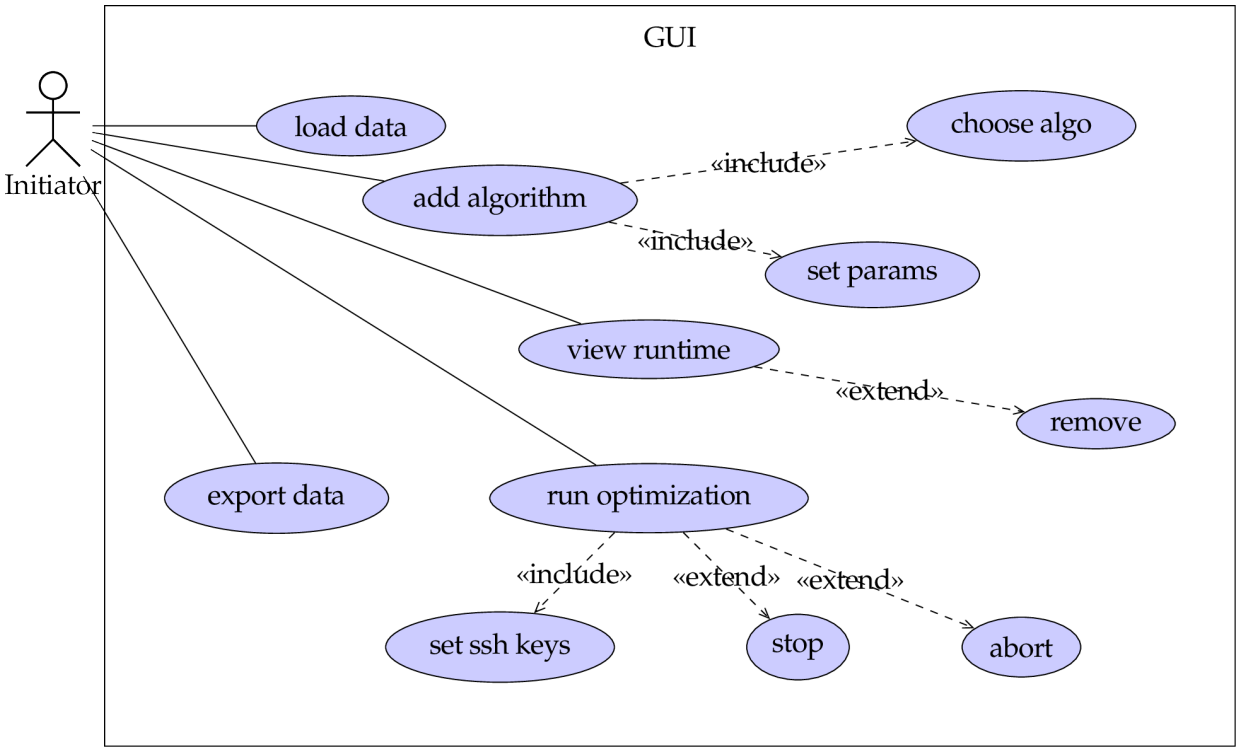


Abbildung 1: Use-case Diagramm GUI

Die folgende Tabelle liefert eine kurze Beschreibung der Funktionen:

Bezeichner	Funktion
`load session`	stellt die Daten einer zuvor gespeicherten Sitzung wieder her.
`add algorithm`	öffnet das addframe
`choose algorithm`	zeigt eine Auswahl der möglichen Algorithmen, von denen eine zu wählen ist.
`set params`	ermöglicht die Einstellung algorithmenspezifischer Parameter.
`view runtime`	liefert einen Überblick über die momentan gewählten Algorithmen und deren gegenwärtigen Status
`remove`	entfernt den ausgewählten Algorithmus
`run`	startet die Berechnung
`set ssh Keys`	Öffnet das Eingabefenster zur Eingabe des Passworts und anderer benötigter Parameter der SSH-Verbindung
`stop`	stoppt die Berechnung
`abort`	verwirft die Berechnung
`export data`	speichert die aktuelle Sitzung

4.3.1 SSH-Verbindung

Direkt nach dem Aufruf der Applikation erscheint neben dem Hauptfenster das Fenster `ssh-Gate` in welchem die Credentials und Einstellungen der SSH-Verbindung.

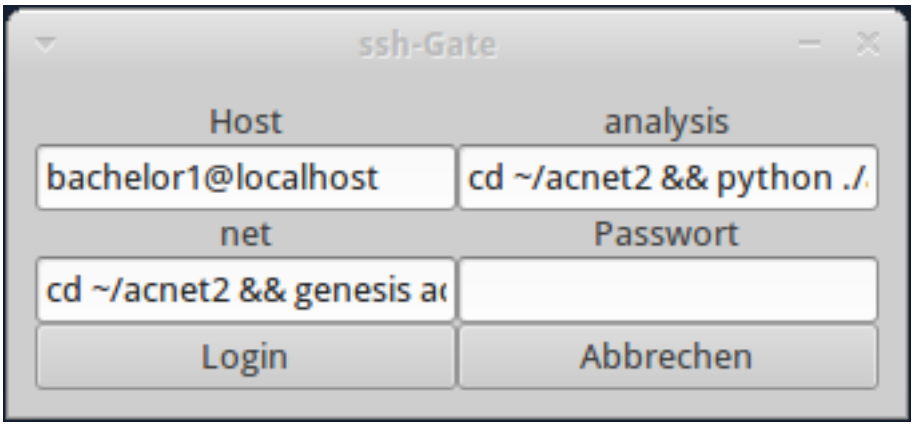


Abbildung 2: Fenster zur Einstellung der SSH-Verbindung

Schlägt die Verbindung fehl, öffnet sich dieses Fenster erneut.

Durch clicken auf **Abbrechen** wird der dieses Fenster geschlossen, es können dann die Sessions weiter bearbeitet werden.

4.3.2 Algorithmen hinzufügen

Durch clicken auf **add algorithm** im Hauptfenster, erscheint das Menü zum Hinzufügen eines Algorithmus-Runs.

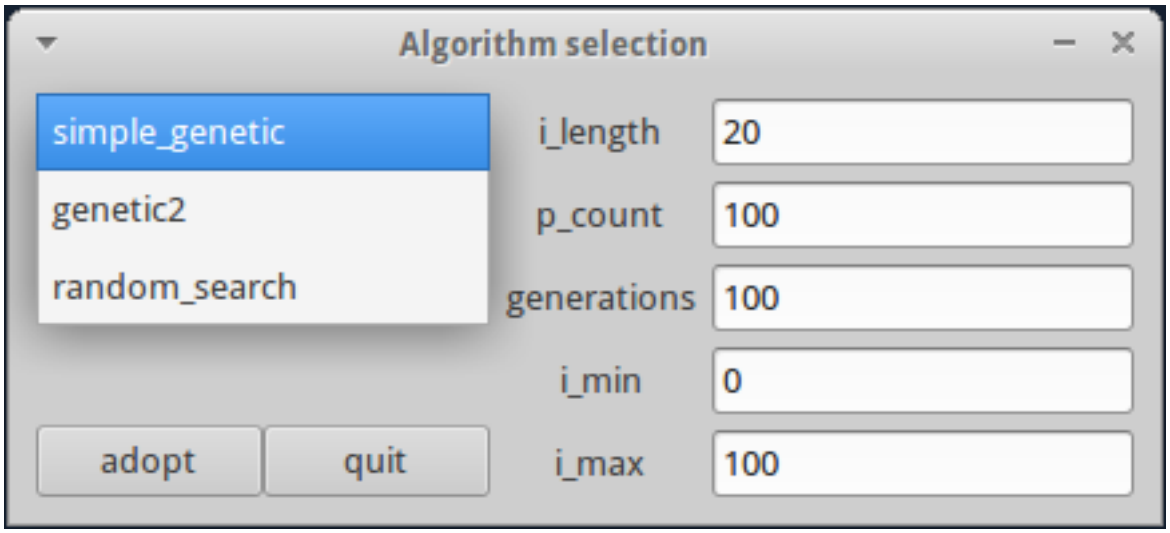


Abbildung 3: Fenster zur Hinzufügen eines Algorithmus-Runs zur Session

Nach einem klick auf **adopt** werden die Einstellungen zur Session hinzugefügt.
Im linken Pull-Down-Menü sind die vorhandenen Algorithmen auswählbar.

4.3.3 Session Starten / Abbrechen

Sind Algorithmen nun in der Session hinzugefügt, kann die Session nun über den orangenen Button **Run** gestartet werden.

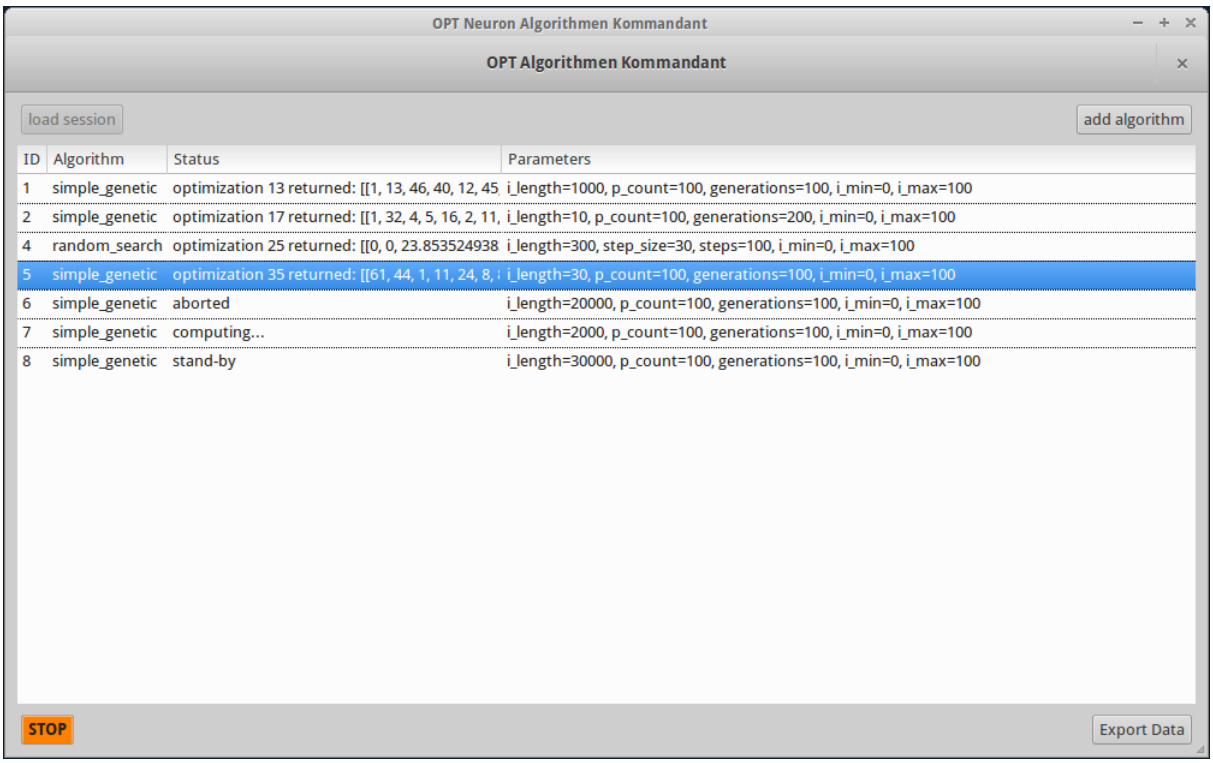


Abbildung 4: Fenster zur Hinzufügen eines Algorithmus-Runs zur Session

Eine Zeile in der Session-Tabelle ist eine Berechnung.

Läuf die Berechnung, ändert sich der button zu **STOP**, über welchen die laufende Session abgebrochen werden kann.

Es ist möglich, eine teilweise berechnete Session wieder aufzunehmen (weiterer Click auf **Run**).

4.3.4 Session-Tabelle

Die Tabelle gliedert sich in vier spalten, ÏD", Älgorithm", SStatusünd "Parameters".

ID is eine fortlaufende Zahl.

Algorithm enthält den Namen des Algorihmuses.

Status zeigt den Status dieser Berechnung an. Wobei ßtand-byëin Algorithmus ist, der noch nicht berechnet wurde, "computing..."bedeutet die Berechnung findet gerade statt, äborted"bedeutet die Berechnung wurde abgebrochen. Wurde eine Berechnung beended, befindet sich hier die Ausgabe.

Parameters enthält die Parameter die für die gegebene Berechnung ausgewählt wurden.

Bei einem Rechtsclick auf eine Tabellenzeile öffnet sich ein Kontextmenü mit den Optionen einen Algorithmus aus der Liste zu entfernen, einen abgebrochenen Algorithmus zu resetten und somit bei einem weiteren Durchgang wieder zu berechnen. Wird die Berechnung gerade durchgeführt, kann man diese einzeln abbrechen, die restliche Session aber nicht. Eine weitere Berechnung findet statt.

4.3.5 Session laden / speichern

Ein Click auf **load session** erlaubt es eine vorher abgespeicherte Session in die Tabelle zu laden. Bestehende Tabelleninhalte werden dabei gelöscht! Es ist hierbei unerheblich, ob die Berechnungen bereits beendet sind, abgebrochen wurden, oder noch nicht durchgeführt worden sind.

Ein Click auf **Export Data** erlaubt es diese Session abzuspeichern. Eine exportierte Session kann hinterher über **load session** wieder geladen werden.
Die Daten werden als einfache CSV-Datei abgespeichert und können problemlos in Textverarbeitungsprogrammen genutzt werden.