Erklärungen zum Programm

Anne Kloskowski

11. April 2014

In diesem Dokument möchte ich eine kurze Einführung in mein Programm zur Optimierung von Parametern in elektrophysiologischen Neuronenmodellen mithilfe eines Genetischen Algorithmus geben. Das gesamte Programm ist in der Programmiersprache Python verfasst und verwendet dessen Eingabehilfe Idle und das Neuronensimulationsprogramm *neuro*Construct, sowie das Modul *inspyred*[1].

1 Das Hauptprogramm

Das Hauptprogramm führt den Genetischen Algorithmus mit einer gewissen Anzahl an Individuen in einer bestimmten Anzahl an Generationen durch.

1.1 Aufbau

Das Programm besteht aus verschiedenen Modulen. Zu Anfang ist da die Startdatei `multiEA.py`. Diese startet die Hauptdatei `GenAlg.py` mit dem Genetischen Algorithmus und ermöglicht die Übergabe aller notwendigen Eingaben und Parameter. Das Hauptprogramm benötigt die Module `Analysis.py`, `diptTestInst.py`, `MultiConductance.py` und `MultiCurrent.py`. Die letzten beiden Module enthalten Methoden zum Aufruf der Simulation der Neurone mit verschiedenen Leitfähigkeiten und gleichen Inputströmen bzw. unterschiedlichen Inputströmen und gleichen Leitfähigkeiten. Zusätzlich gibt es noch ein paar Textdateien: `config.txt`, `channel.txt`, `location.txt`, `density.txt` und `index.txt`. Diese speichern die Konfiguration, also die Eingaben aus der Startdatei, und die Daten der Ionenkanäle (Name, Ort und Leitfähigkeit) zur Übergabe an die Simulation.

1.2 Eingaben und Parameter

Vor dem Start des Programms hat man die Möglichkeit gewünschte Einstellungen und notwendige Parameter (siehe Tabelle 1) zu übergeben. Dies geschieht in der Datei `multiEA.py`

[1] `http://inspyred.github.io/`

mit dem Aufruf GenAlg.start(...) des Moduls `GenAlg.py`. Hier können mehrere Durchläufe gestartet und jedem einzelnen andere Einstellungen übergeben werden. Die Reihenfolge der Parameter spielt dabei keine Rolle, wobei jedoch die richtige Schreibweise eingehalten werden muss. Die Einträge werden durch Kommas getrennt und Text muss in Anführungszeichen stehen.

BEACHTE: Dies ist ausschließlich **vor** dem Start möglich.

1.3 Start

So, starten wir nun das Programm. Dafür wechselt man in der Konsole in den Ordner mit der Startdatei. Anschließend sollte die IDE von Python *Idle* mit dem Befehl `idle` in der Konsole geöffnet werden. In *Idle* gibt man nun den Befehl `execfile("multiEA.py")` ein. Daraufhin startet das Programm mit den in `multiEA.py` festgelegten Einstellungen. Wurde show = 1 gewählt, so informiert euch *Idle* nun fortlaufend über den Fortschritt der Evolution. Die eigentlichen Ausgaben der Simulation etc. findet ihr in der Konsole, die nun massiv arbeiten sollte.

Wollte man *Idle* nicht benutzen, kann man auch `python` in der Konsole aufrufen und dort den Startbefehl geben. Die Ausgaben, die in *Idle* produziert worden wären, gehen nun zwischen den Ausgaben der Simulation etc. unter.

1.4 Ergebnisse

Zum Schluss spuckt das Programm Ergebnisse aus. Dabei handelt es sich um die abschließende Generation, wobei die Individuen in ihrem Rang absteigend sortiert sind. Ich habe mir immer die ersten fünf Individuen ausgeben lassen. Der Rest verfällt. Diese Individuen werden in der Ergebnisdatei in Textform gespeichert. Dabei werden im Kopf alle relevanten Eingaben aus der Konfigurationsdatei mit Populationsgröße, Mutationsrate usw. gespeichert, dazu Anfangs- und Endzeit der Evolution und darunter die Werte der 12 Ionenkanalleitfähigkeiten der besten Individuen. Zusätzlich werden in zwei Analyseverfahren (bereitgestellt von *inspyred*[2]) die besten, mittleren und durchschnittlichen Fitnesswerte der Individuen bzw. die Eigenschaften des besten, mittleren und durchschnittlichen Individuums über die Generationen hinweg gespeichert. Diese erstellen bei extra Aufruf dieser Dateien zwei unterschiedliche Grafiken (dazu mehr in „2. Auswertung der Ergebnisse“)

2 Auswertung der Ergebnisse

In der Auswertung ging es mir in erster Linie darum, die Neurone der finalen Population zu rekonstruieren und alle ihre Eigenschaften separat noch einmal berechnen zu können. Es ist mir leider bis zum Schluss meiner Arbeit nicht gelungen, dies konsistent umzusetzen. Die benötigten Module habe ich euch jedoch zur Verfügung gestellt. Dabei gibt es die Startdatei `startAuswertung.py` und in dem Ordner *Auswertung* die Dateien, die die Leitfähigkeiten verarbeiten und speichern (`CALCdens.txt`,`chrom_in_dens.py`

[2] siehe `http://inspyred.github.io/`

Variablenname	**Erklärung**	**Default**
proj_name	Name des verwendeten *neuro*Construct -Projekts	„Pyr_RS“
proj_path	absoluter Pfad des *neuro*Construct -Projekts auf dem Computer	–
sim_config	Simulationskonfiguration, anhängig von verwendetem Projekt	„Default Simulation Configuration“
stimulation	Name der Stimulation laut Projekt	„Input_0“
cell	Name der Zelle laut Projekt	„L5TuftedPyrRS “
duration	Dauer der Simulation in Millisekunden	500
dt	Schrittweite der Simulation in Millisekunden	0.05
currents	Inputströme, mit denen ein Neuron simuliert werden soll: [Anzahl, Start, Schritt] (z.B. [2,0.5,0.3] ergibt die drei Ströme 0.5 und 0.8nA)	[3,0.2,0.3]
pop_size	Größe der Startpopulation	50
max_generations	maximale Anzahl an Generationen	30
mode	Art der zu suchenden Nervenzelle (1:RS, 2:FS, 3:IB, 4:CH)	Pyr_RS: 1, Pyr_IB: 3
crossover_rate	Wahrscheinlichkeit eines Crossovers	1
mutation_strength	Parameter in der Uniformen Mutation	0.4
mutation_rate	Wahrscheinlichkeit für Mutation	1/12.0
num_selected	Anzahl der Individuen, auf die genetischen Operatoren angewandt werden sollen; wegen des paarweisen Crossovers bietet sich eine gerade Anzahl an	pop_size
num_co_points	Anzahl der Crossover-Punkte (zwei Teilen das Chrosomom in drei Teile, wobei das Mittlere ausgetauscht wird)	1
num_elites	Anzahl der besten Individuen, die auf jedenfall in die nächste Generation kommen	1
tournament_size	Anzahl der Individuen, die gegeneinander antreten sollen	2
thrFourier	STrafe für unzulässigen Fourierwert	-10000
weights	Gewichte der Eigenschaften [apw,ibf,ir,ai,slope], wichtig für die Bewertung	[1,1,1,1,1]
penalty_ai_X	Strafwert für einen unzulässigen Wert für *ai*, X: RS oder FS	-3500
penalty_Y_X	Strafwert für einen unzulässigen Wert für Y:*ibf* oder *ir*, X: CH oder IB	IB:-2500, CH:-3500
custom	individuelle Kombination von Ersetzen und Selektion? Ja: 1, Nein: 0	0
BS	Betriebssystem: Linux (1), Windows (2)	1
anhang	Notizen, die zum Schluss bei den Ergebnissen stehen sollen	„“
show	aktiviert (1) das Protokoll in der IDE *Idle*	1

Tabelle 1: Mögliche Eingabeparameter

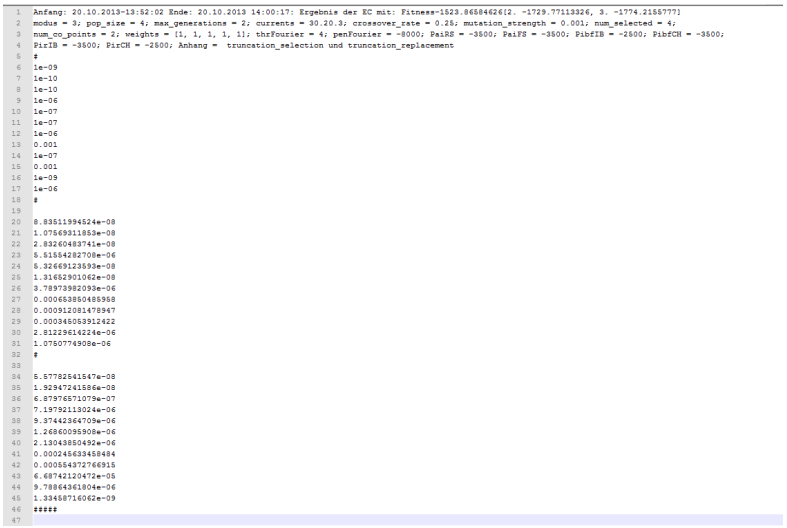

Abbildung 1: Beispiel für die Ergebnisdarstellung in der Speicherdatei

und `calc_dens.py`). Damit kann nun die Simulation und Analyse des Neurons gestartet werden.

Zur Darstellung der Aktionspotenzialverläufe existiert das Modul `Plotten.py`. Diese kann Aktionspotenziale von beliebig vielen unterschiedlichen Inputströmen darstellen.

Eine weitere Analyse ermöglicht `FFT.py`. Aufgerufen über `FFTstart.py` berechnet dieses Modul die Fouriertransformation des Aktionspotenzials und zeichnet diese.

Wie oben schon erwähnt, stellt uns *inspyred* Funktionen zum Darstellen der Eigenschaften der Individuen bereit. Die beiden Dateien, in denen die Daten gespeichert werden, heißen 'inspyred-statistics-file-<timestamp>.csv' und 'inspyred-individuals-file-<timestamp>.csv'. Beide werden jeweils über den Befehl *inspyred.ec.analysis.X* aufgerufen[3], wobei *X* entweder für *generation_plot(filename)* bzw. *allele_plot(filename)* steht. Dafür muss aber vorher die Datei `analysis.py` aus dem Installationsordner von *inspyred* importiert werden. Beispiele, für solche Grafiken, sind in den Abbildungen 2 und 3 dargestellt.

[3]Näheres dazu unter `http://inspyred.github.io/`

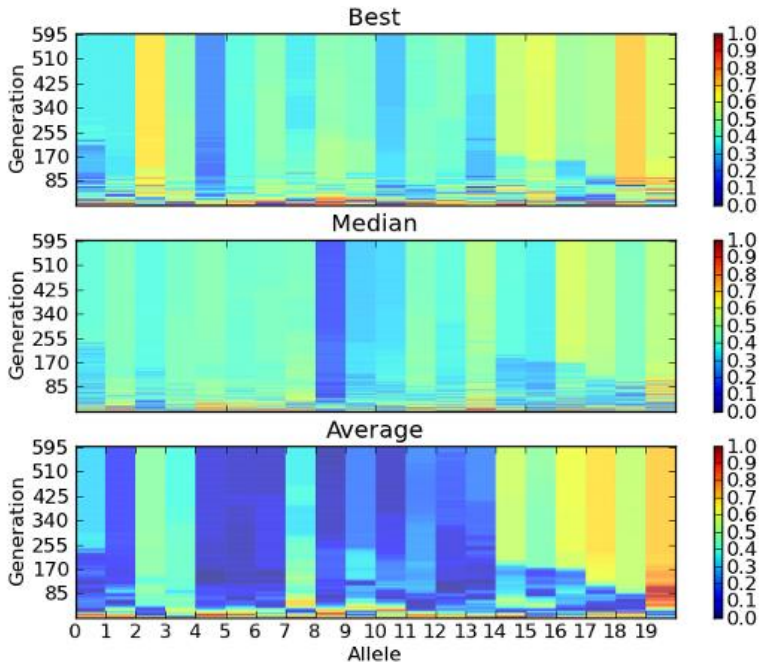

Abbildung 2: Beispiel für einen Allel-Plot

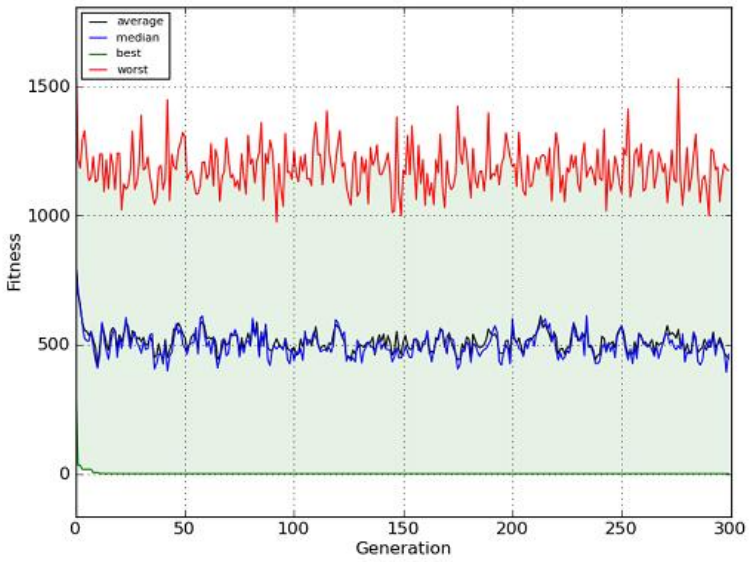

Abbildung 3: Beispiel für einen Generation-Plot