LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

Kooperative Neuroevolution in RoboCup2D

Alexander Isenko

Entwurf vom 20. Oktober 2016

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

Kooperative Neuroevolution in RoboCup2D

Alexander Isenko

Aufgabensteller:	Prof. Dr. Claudia Linnhoff-Popien
Betreuer:	Thomas Gabor, M.Sc Lenz Belzner, Dipl.-Medieninf.
Abgabetermin:	11. November 2016

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 11. November 2016

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Wir untersuchen in dieser Bachelorarbeit verschiedene Ansätze zur Entwicklung von neuronalen Netzen am Beispiel der Cross Entropy Method, genetische Algorithmen und CoSyNE unter Einschränkung von spärlichen Fitnesssignalen, hochdimensionalen kontinuierlichen Zustandsräumen und simulationsbasierter Optimierung.
Der Suchraum wird durch diskrete Cosinustransformationen (DCT) unter der Annahme reduziert, dass benachbarte Gewichte zueinander korelliert stehen. Die Domäne ist ein Fußballsimulator, Half Field Offense (HFO), der Teams aus dem weltweiten Wettbewerb RoboCup zum Vergleichen bereitstellt. Die Implementierung erfolgte in Haskell und Python.

Inhaltsverzeichnis

1. Einführung

- Finden von Lösungen ohne große Anpassung von Hyperparamteren
- Domänen mit starken Einschränkungen (sparse Fitness, hochdimensionale kontinuierlicher Zustandsraum)
- Maschinelles Lernen ist eine potenziell große Hilfe zur Entwicklung von besseren Systemen

Die Nützlichkeit von Automatisierung ist offensichtlich Viele benutzen Neuronale Netze mit Backpropagation, andere Möglichkeiten wie man Netze trainiert fallen ausm Fokus Ich spreche ein paar Möglichkeiten an und untersuche sie auf ihre Nützlichkeit in Fälle wo Backprop nicht möglich ist

1.1. Aufgabenstellung

- Verschiedene Techniken auf multi agenten systeme anzuwenden
- Kooperation und homogenität von Gewichten im Netz untersuchen
- Schauen ob ichs hinkriege

In dieser Arbeit beschäftige ich mich mit der Implementierung von verschiedenen Machine Learning Algorithmen für eine Domäne Half Field Offense mit sparsen Fitnesssignalen und einem hochdimensionalen kontinuierlichen Zustandsraum und stelle diese gegenüber. Aus dem Vergleich kann man die Nützlichkeit für andere Domänen mit ähnlichen Einschränkungen erahnen.

1.2. Motivation

- Praktikum - DARTS
- Gespräche mit Kommilitonen
- Abstrakte Zusammenhänge nutzen um Abhängigkeiten auf der Domänenseite zu abstrahieren

Im Siemens Praktikum bin ich das erste Mal mit Genetischen Algorithmen in Kontakt gekommen Es kam die Frage auf welche Domänen ein Schwierigkeiten für Machine Learning Tasks darstellen und in der Industrie anwendbar sind Verteilte Systeme, Roboter die zusammen etwas lernen, etc Fitness Engineering, also die Anpassung von dem Rewardsignal, ist oft nötig für Aufgaben wo der Reward von viele Zeitschritten und kontinuierlichen Aktionsketten abhängt Deshalb kam die Idee dies auf die Probe zu stellen, mit einem Fußballsimulator und neuen Algorithmen, die einen Durchbruch erreicht haben (CoSyNE)

1.3. Aufbau der Arbeit

- Erklärung der Grundlagen, GA, NN, DCT, CoSyNE, CE
- Aufbaut der Domäne, Implementierung der Algorithmen
- Präsentation der Resultate
- Ausblick
- Die Implementierung und Problematiken sind im Appendix
- Veränderungen während der Arbeit

Im Rahmen dieser Arbeit werden im Kapitel 2 die Grundlagen von Genetischen Algorithen und deren Verknüpfung zu neuronalen Netzen und der Cross Entropy Method erklärt und anschaulich dargestellt. Kapitel 3 beschäftigt sich mit der Domäne, Parametrisierung der Algorithmen und der jeweiligen Resultate. Das Kapitel 4 gibt einen Ausblick in weitere Verbesserungsmöglichkeiten und legt verwandte Felder dar. Im Appendix wird die Implementierung von dem gesamten System überschlagen, Problematiken angesprochen und Lehren die für eine zukünftige Arbeit gezogen werden können angesprochen.

2. Definitionen

Dieses Kapitel bietet Einblick in die Grundlagen von **Genetischen Algorithmen** im Zusammenhang mit **neuronalen Netzen** und der **Cross Entropy Method**. Außerdem werden einige Verbesserungen zu den naiven Methoden besprochen, wie die Reduzierung des Suchraums durch **Fouriertransformationen** und die Einführung von einer **kooperativen Evolution** durch Hinzufügen von einer neuen Methode zum GA.

2.1. Genetische Algorithmen

“Die Motivation von Genetischen Algorithmen kam aus der Natur bla blubb”
“Im Folgenden behandeln wir die grundlegenden Operationen die einen GA ausmachen”

2.1.1. Selektion

“Die besten x prozent werden genommen”

2.1.2. Kreuzung

“Es gibt viele Kreuzungsoperationen, wir behandeln nur welche am Beispiel von einer Liste”
“one-point, two-point und n-point crossover”

2.1.3. Mutation

“Mutation ist hilfreich um die Varianz der Population etwas zu erhöhen”

2.1.4. Repopulation

“Ist ein Schritt der oft implizit beim Crossover passiert, (zitat finden), wird nicht nur benutzt um Kinder und Eltern zu verknüpfen, sondern auch völlig neue Individuen hinzuzufügen”

2.2. Neuroevolution

“Neuroevolution beschäftigt sich mit der Verknüpfung von Genetischen Algorithmen und Neuronalen Netzn”

2.2.1. Neuronale Netze

“Neuronale Netze sind eine riesige Gleichung”
“Formel von Machine Learning zeigen”
“Backpropagation wird angesprochen aber nicht ausführlich erklärt”

Dense Ebene

“Kleines Beispielnetz aufmalen und durchrechnen”

LSTM Ebene

“Kleines Beispielnetz aufmalen und durchrechnen”

Softmax Ebene

“Erklärung bieten wieso es sowas gibt und was für eine Formel angewendet wird”

2.2.2. Verbindung mit genetischen Algorithmen

“Die Verknüpfung findet in der Kodierung von den Individuen statt - wir nehmen eine (naive) Darstellung von allen Gewichten” “Problematik -> Folgerung zu DCT”

2.3. Diskrete Cosinus Transformation - DCT

“Der Suchraum kann durch die sog. DCT auf beliebige Dimensionalität eingeschränkt werden, wenn bestimmte Annahmen getroffen werden können”
“Fouriertransformationen machen folgendes...”
“Es gibt eine Inverse die aus einem n-stelligen liste eine m-stellige macht, wo die Zahlen korelliert sind (Beispiele)”

2.3.1. Kodierung des Suchraums

“Die Kodierung besteht aus eine 20-stellingen Liste wie im Paper (Cosyne), wo damit 2k Gewichte entwickelt wurden”

2.4. Cooperative Synapsen Neuroevolution - CoSyNE

“CoSyNE wurde vom Prof.Dr.Schmidhuber an der ETH Zürich entwickelt und hat damit sehr viele anderen Algorithmen in den Schatten gestellt”
“Methodik, ist wie GA bloß mit einer Aktion mehr die statt spielbare ‘Policies’, nur im Koeffizientraum entwickle”
“Dies erlaubt eine (zitat) Kooperative Entwicklung von Koeffizienten für die nachfolgenden Inv.DCT”
“Kein Crossover, geringe Mutation”
“Beispiele vom Erfolg”

2.4.1. Permutation

“Wir transponieren, shuffeln und transponieren zurück”

2.5. Cross Entropy

“Cross Entropy ist eine andere Möglichkeit um die Individuen darzustellen, anstatt von Zahlen, hab ich nun pro Coeffizient eine Normalverteilung habe mit Mean und STD” [**?**]

2.5.1. Normalverteilung

“Was ist eine Normalverteilung”
“Wie programmiere ich eine Normalverteilung selber, Box-Muller, Randomness, Haskell-code Beispiel”

3. Umsetzung in RoboCup2D

“Was ist RoboCup”
“Was für Ligen gibt es”
“Was für eine Domäne ist es im Vergleich zu anderen”

3.1. Half Field Offense

“Was ist Half Field Offense”
“Wie ist das Spielfeld aufgebaut”

3.1.1. Zustandsraum

“Unterschiedliche Zustandsräume, kommt drauf an wie viele Spieler aufm Feld sind (zitat vom HFO Paper)”
“Angepasst für Maschinen (zitat HFO)”
“Umrechnung für Menschen”

3.1.2. Aktionsraum

“Gibt 5 nicht parametrisierte Aktionen, die wir benutzt haben”
“Gibt noch andere parametrisierte”

3.1.3. Einschränkungen

“Sparse Fitness”
“Simulation Learning”
“Hochdimensional Kontinuierlich”
“Auch genannt Black Box RL”

3.2. Implementierung der Algorithmen

“Server in Haskell, HFOServer in C++, Agenten in Python”

3.2.1. Wahrscheinlichkeitsverteilung von Aktionen

“Einfache Kodierung von Aktionen durch eine Wahrscheinlichkeitsverteilung, der Agent sampelt raus”
“Parametrisierung”

3.2.2. Cross Entropy mit DCT

“Parametrisierung”

3.2.3. Neuroevolution mit DCT

“Parametrisierung”

3.2.4. CoSyNE mit DCT

“Parametrisierung”

3.3. Resultate

“Die Resultate wurden an einem PC mit Specs ausgeführt”

3.3.1. 1v1

“Domäne nochmal erklären, Zustandsraum spezifizieren, sodass alles replizierbar ist”

Wahrscheinlichkeitsverteilung

“Graph zeigen, Interpretation bzw. Erklärung” “Nach 25 Episoden stagniert”

Cross Entropy

“Graph zeigen, Interpretation bzw. Erklärung” “Aggressivität” “Nach 50 Episoden stagniert”

Neuroevolution

“Graph zeigen, Interpretation bzw. Erklärung”

CoSyNE

“Graph zeigen, Interpretation bzw. Erklärung”

3.3.2. Vergleich

“Sicherheit <-> Aggressivität steht im Kontra zur Stabilität der Algorithmen”
“Wenn Zeit ein Faktor wäre”
“Wenn Sicherheit ein Faktor wäre”

4. Diskussion

“Im Folgenden bespreche ich die Nützlichkeit der Ergebnisse”

4.1. Anwendungsmöglichkeiten

“Wenn starke Einschränkungen bestehen sind, waren diese Algorithmen zielführend”

4.2. Coevolutionärer Aspekt

“Die Annahmen für Netze zu treffen ist interessant gewesen, hat anscheinend auch für 30k Gewichte geklappt, sollte man später zum checken”

4.3. Ausblick

“Im Folgenden gebe ich meinen Senf zu den Parametrisierungen an und wenn die Welt perfekt wäre, was ich dann machen würde”

4.3.1. Genetische Algorithmen

“Hab nicht an den Parameteren gespielt”

4.3.2. Aufbau des neuronalen Netzes

“Hab nicht verschiedene Netze ausprobiert”

4.3.3. Cross Entropy

“Hab die naivste Implementierung genommen, könnte viel viel besser werden, wenn man Grips reinsteckt”

4.3.4. Aktionsraum

“Hätte bessere Aktionen definieren können, die schlauere Spielzüge erlauben, da ich die Vermutung habe dass die jetzigen ein Hardcap an Qualität bieten”

4.3.5. Multi-Agenten Systeme

“Hatte noch keine Möglichkeit in diese Domäne ausführlich zu testen”

4.4. Verwandte Felder

“Folgende Reads sind nice”

4.4.1. Implementierung für OpenAI Gym

“Update für OpenAI Gym, meine Impementierung wird folgen”
“Keine Multi-Agenten Setting möglich, dsw ist meine Arbeit nicht umsonst gewesen”

4.4.2. Bestärkendes Lernen - Black Box RL

“Wenn man die Netze als Policy betrachtet ist meine Arbeit eine Anwendung im Reinforcement Learning Bereich gewesen”

4.4.3. Convolutional neuronale Netze und CoSyNE

“Warum ist die Annahme für homogene Netzstrukturen besser für Convolutional Networks”
“Evtl ist CoSyNE dort richtig geil”

Roboterarm

“Bilder, Graphen und Parametrisierung zeigen + Link + Zitat”

Rennspiel

“Bilder, Graphen und Parametrisierung zeigen + Link + Zitat”

A. Appendix

“Da der Fokus sehr stark auf dem Machine Learning Bereich lag, aber die Domäne viele Tücken hatte und die Implementierung einen Großteil der Zeit in Anspruch genommen hat, wird es hier behandelt”

A.1. Architektur

“Die Architektur des gesamten Projektes wurde in Haskell geplant, da ich mit externen und unsicheren Implementierungen arbeite”
“Automatische Docs, weil ich gute Kommentare schreibe” “Hat geholfen Fehler schneller zu finden und Codereuse ist super geil gewesen”
“Bild von der Kommunikation zeichnen”

A.1.1. Haskell Server

“Module”
“Typklassen sind geil”
“Globale Config ist nice”
“Automatische Serialisierung mit Aeson”

A.1.2. Python Agent

“Module”
“CMD Parser”
“Keras”

A.1.3. Kommunikation

“Haskell <-> Python: JSON ist von Python nativ als Dict unterstützt” “Python <-> Server: FFI Python to C++ (HFO) + (Zitat)”

A.1.4. Parallelisierungsmöglichkeiten

“Bottleneck ist die Kommunikation über JSON-Files”

A.2. Statistik

“Für Cross Entropy hab ich Varianz und STD mit Seed gebraucht, für MonadRandom gabs keine Implementierung, dsw hab ich eine eigne gemacht”

A.2.1. Lineares Laufzeit und Speicherkomplexität für Evaluation

“Folds sind supernice, minimale Erklärung, Links zu Gabriels Blog, Vorzeigen von Effizienz”

A.2.2. Stabile Varianzfunktion

“Catastrophic Cancellation, erste Lösung, zweite Lösung von Gabriel mit E-Mail Austausch”

A.3. Problematiken

“Server ist scheiße, nicht besonders gut dokumentiert”

A.3.1. HFO Server

“Erfolgreicher Durchlauf ist abhängig vom Computer, bzw. von der Leistung”
“Undefinierte Lags zwischen Step 2k-8k”
“Bedienung vom Visualizer ist nicht richtig erklärt”
“Visualizer produziert nicht benutzbare logs, nur das erste Spiel ist ‘abspielbar’, rest ist korrupt ”
“Einloggen von Spielern nimmt sehr viel Zeit in Kauf, dafür muss ein Austausch von Policies on-the-fly passieren”
“Visualzer bricht bei 24k Steps ab, aber ohne ihn funktioniert die Simulation nicht, laggt nur rum”
“Es gibt keine Zeitangaben wie lange ein Agent nicht aktiv sein darf, wenn er keine Entscheidung abgibt wird er ignoriert, dieser Fall sollte behandelt werden für kompetative Benutzung”

A.3.2. HFO Python Library

“Kodierung von Zuständen in denen sich der Agent befindet gibt ein Hexdump zurück, man muss per Hand die ENUMS herausfinden und hardcoden”