NeuGenGo

Kann unser neuronales Netz besser Go spielen als wir?

Lennart Braun, Armin Schaare, Theresa Eimer

Praktikum Parallele Programmierung
Fachbereich Informatik
Universität Hamburg

03.06.2015

Gliederung (Agenda)

Ziele

- Ein gut spielendes Netz als Ergebnis
- Spiele und Netze visualisierbar machen
- Einen guten Vererbungsmechanismus finden

Go - Das Spiel

- Asiatisches Brettspiel
- Wird auf Brettern mit 19x19 Knoten gespielt
- Ziel: gleichzeitig Gebiet einkreisen und gegnerische Steine schlagen
- Spielende: wenn beide Spieler passen

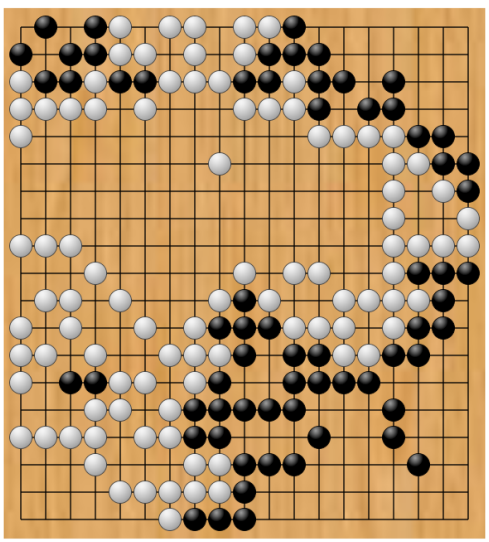

Figure : Beispiel eines Bretts

Quelle: http://www.13thmonkey.org/Artdraw/images/kifu2.png

Go - Die Umsetzung

- Gespielt wird auf kleineren Brettern (bis 9x9)
- Repräsentation des Brettes als eigener Datentyp, der den Spielzustand speichert
- Das Spiel endet, wenn es keine gültigen Züge mehr gibt

Das Netz

Neuronales Netz mit...

- ... n Input-Neuronen für die Knoten auf dem Spielfeld
- ... beliebig vielen hidden layers mit jeweils beliebig vielen Knoten
- ... n Output-Neuronen für die Knoten auf dem Spielfeld (oder $n + 1$ zum Passen)
- Pro Knoten eine Präferenz, auf die höchste wird gesetzt
- Bei ungültigen Zügen wird die nächste Präferenz gezogen und es gibt einen Punktemalus.

Was das Netz kann

- Die Eingangssignale für die nächste Schicht an Neuronen berechnen
- Die Signale zu einer gültigen Ausgabe auswerten
- Den Aufbau des Netzes ausgeben
- Ein Netz als Datei ausgeben
- Mit dem Backpropagation Algorithmus supervised learning betreiben

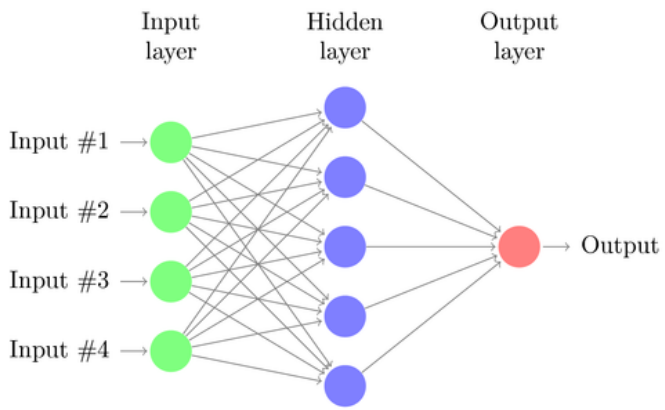

Figure : Beispielnetz

Quelle:

Lernen mit Backpropagation

- Supervised learning mit Eingabedaten und erwarteten Werten
- Trainiert wird auf regelkonforme Züge
- Ziel: Präferenz für falsche Züge soll 0 sein
- Vergleich von Ausgabe und erwarteten Werten
- Gemäß dem Fehler werden die Kantengewichte von hinten nach vorne angepasst

Der genetische Algorithmus

- Je mehr Spiele ein Netz gewinnt, desto wahrscheinlicher überleben dessen Eigenschaften
- Verschiedene Möglichkeiten die Vererbung zu gestalten:
  - Variable Lebensdauer von Netzen
  - Verschiedene Mutationswahrscheinlichkeiten
  - Crossovers
- Finden der besten Kombination durch Ausprobieren

Lösungsansatz

Sequentieller Algorithmus

1: Generiere ein zufällige Menge N_0 an neuronalen Netzwerken
2: Trainiere sie mit Backpropagation damit sie regelgerecht spielen
3: **for** Runde $i = 0$ bis ... **do**
4: **for** $n, m \in N_i$ **do**
5: lass die Netze n, m gegeneinander spielen
6: und zähle die Anzahl der Siege
7: **end for**
8: generiere die Menge N_{i+1} mit Hilfe eines genetischen Algorithmus abhängig von N_i und den Ergebnissen der Spiele
9: **end for**
10: Speichere die Netze

I/O und Visualisierung

I/O

- Smart Game Format
  - Speichern und Analyse von Partien
  - Generieren von Trainingsdaten für die Netze
- Go Text Protocol
  - Kommunikation mit anderer Software
- Dumps der neuronalen Netze

Visualisierung

- Abspielen von im SGF gespeicherten Partien

Parallelisierungsschema

Hybride Parallelisierung

- verteilter Speicher
  - mehrere Partien auf verschiedenen Knoten
  - Backpropagation auf verschiedenen Knoten
- gemeinsamer Speicher
  - parallele Berechnung der Ausgaben der neuronalen Netze
  - ggf. Parallelisierung innerhalb des Go Moduls