Computer spielen Computerspiele mit Hilfe von reinforcement learning am Beispiel Vier Gewinnt

Pablo Lubitz

BACHELORARBEIT

eingereicht am
Universitäts-Bachelorstudiengang

Informatik

in Bremen

im September 2019

Betreuung:

Holger Schultheis, Thomas Barkowsky

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Bremen, am 10. September 2019

Pablo Lubitz

Inhaltsverzeichnis

Vorwort

TODO

Kurzfassung

TODO

Abstract

TODO engl.

Kapitel 1

Einleitung

In der Bachelorarbeit soll ein Programm entwickelt werden das mit Hilfe von reinforcement learning trainiert um das Spiel Vier Gewinnt zu erlernen. Es wird ein reinforcement learning Ansatz gewählt da hierdurch automatisiert individuelle Gegner geschaffen werden können die das können unterschiedlicher Spieler widerspiegeln.

1.1 Motivation

Warum spielen wir? Wir spielen um Fähigkeiten zu erlernen, ein Kind lernt Objekte nach Formen zu sortieren oder eine Katze lernt spielerisch das Jagen von Beute. Doch auch erwachsene Menschen spielen noch und können hierdurch ihre Fähigkeiten verbessern. Um den bestmöglichen Effekt zu haben muss das Spiel eine gewisse Schwierigkeit haben die aber noch zu bewältigen sein muss. Da aber jeder Mensch, egal wie erfahren, das Spiel spielen soll ist es oft Hilfreich einen Gegner zu haben der ein ähnliches Level an Erfahrung mitbringt. Wenn nun diese Rolle des Gegners nicht von einem Menschen besetzt werden kann ist es naheliegend einen Computergegner zu erschaffen. Die Frage die ich in dieser Bachelorarbeit versuche zu beantworten ist: Ist es möglich durch reinfocement learning automatisiert verschieden starke Computergegner aus einer Simulation zu extrahieren.

1.2 Problemstellung

Ein Computergegner der für das Spiel Vier Gewinnt erschaffen wird kann auf verschiedenen Algorithmen basieren er könnte zum Beispiel in jedem Zug einfach zufällig eines der sieben möglichen Einwurflöcher bedienen oder er könnte durch einen Minimax Algorithmus zu jedem Zeitpunkt den bestmöglichen Zug errechnen. Das Problem hieran ist, dass jeder einzelne Algorithmus programmiert werden muss und somit ein enormer Arbeitsaufwand entstehen kann. Ein Lösungsansatz für diese Problem sind selbstlernende Algorithmen die mit Hilfe von reinforement learning und neuronalen Netzen selbstständig ähnlich wie ein Mensch erlernen wie das Spiel zu gewinnen ist. Hierraus können dann automatisch Computergegner generiert werden.

1.3 Umsetzung

Es wird eine digitale Version des Spiels Vier Gewinnt genutzt und eine Schnittstelle erschafft die es ermöglicht auszuwählen ob ein Mensch oder der Computer die Rolle der Spieler übernimmt. Spielt der Computer soll zwischen verschiedenen Algorithmen ausgewählt werden können. Dies ist wichtig da ein selbstlernender Algorithmus viele Spiele durchlaufen muss um einen Lernfortschritt aufzuzeigen. Somit kann in der Simulation dann trainiert werden ohne jedes einzelne Spiel gegen einen Menschen spielen zu müssen. Als Trainingsgegner wird ein Minimax Algorithmus dienen. Nachdem die Simulation dann durchlaufen ist können verschieden gute Varianten des selbstlernenden Algorithmus extrahiert werden indem die aktuelle Version zu verschiedenen Trainingsstadien abgespeichert wird. Die Auswahl der Varianten wird anhand der prozentualen Gewinnchance durchgeführt.

1.4 Evaluation

Um zu testen wie gut der maschine learning Algorithmus das Spiel in seinen verschiedenen Trainingsstadien gelernt hat werden die Varianten gegen Menschen antreten. TODO Hierdurch soll sich dann zeigen, dass durch das maschine learning Gegner auf automatisierte Weise geschaffen werden können die einen fein granularen Schwierigkeitsanstieg bieten können.

1.5 Related work

TODO Arbeit von Lukas und Arbeit von dem anderen

Kapitel 2

Grundlagen

In dieser Arbeit werden einige Technologien genutzt um die Problemstellung zu lösen. Die wichtigsten dieser Technologien werden hier erklärt um das weitere Verständniss des Lesers zu gewährleisten.

2.1 Vier Gewinnt

Vier Gewinnt ist ein Spiel für 2 Spieler in dem abwechselnd Spielsteine in ein vertikales Spielfeld eingeworfen werden. Jeder Spieler kann sich in seinem Zug zwischen einem von sieben Einwurflöchern entscheiden. Wird ein Spielstein eingeworfen so fällt dieser auf die niedrigste der sechs Positionen die noch nicht durch andere Spielsteine gefüllt ist.Hierdurch ergibt sich ein Spielfeld mit 42 Feldern. Sind schon sechs Spielsteine in ein bestimmtest Einwurfloch gesteckt wurden so darf hier kein weiterer Spielstein eingeworfen werden. Ein Spieler gewinnt das Spiel wenn er es geschafft hat, dass sich vier seiner Spielsteine in einer Reihe befinden. Eine Reihe kann horizontal, vertikal oder diagonal gebildet werden. Werden alle 42 Positionen mit Spielsteinen befüllt ohne eine Reihe von vier Steinen des selben Spielers zu bilden geht die Partie unentschieden aus. TODO BILD

2.2 Minmax Algorithmus

Ein Minmax Algorithmus beschreibt einen rekursiven Ansatz zum finden einer optimalen Lösung für Probleme von zwei Parteien mit widersetzlich Zielen. Dies sind in der Regel Nullsummenspiele mit perfekter Information, es gibt also für jeden Gewinn des einen Spielers genauso viel Verlust für den anderen und es gibt kein Spielelement, dass nur für einen Spieler einsehbar ist. Der Minmax Algorithmus berechnet sich hierfür den Suchbaum des Spiels welcher alle möglichen Züge beider Spieler in nachvollziehbarer Reihenfolge enthält. Hierbei werden in jedem Rekursionsschritt alle möglichen Züge des derzeitigen Akteurs betrachtet und bewertet. Wenn der derzeitige Zug nicht zum Gewinn führt, findet eine Bewertung durch das Betrachten des nächsten Rekursionsschrittes statt. Der Name Minmax ergibt sich durch das Betrachten der abwechselnden Züge von den Akteuren wobei jeweils der maximal beste Zug für den einen der minimal beste Zug für den anderen bedeutet. Dieses Verfahren führt bei einfachen Spielen wie

Tic-Tac-Toe zu einem perfekten Spieler, da hier eine überschaubare Menge an möglichen Zügen betrachtet wird (9 Felder die von 2 Spielern gefüllt werden). Für komplexere Probleme gibt es die Möglichkeit eine Suchtiefe zu bestimmen wodurch die Rekursion nur bis zu dieser Tiefe durchgeführt wird. Dies ist Sinnvoll da die Berechnung einzelner Züge sonst sehr lange dauern kann.

2.2.1 Negamax

Negamax ist eine Variante des Minmax Algorithmus die TODO.

2.2.2 Alpha-Beta?

TODO.

2.3 Reinforcement Learning

Reinforcement learning (Bestärkendes Lernen) ist eine Maschine-Learning Methode bei der ein Agent mit Hilfe eines Environments, Policy, State, Actions und eines Rewards lernt eine ihm gegebene Aufgabe zu lösen.
Die Aufgabe ist in diesem Fall das Spiel Vier Gewinnt zu meistern. TODO Bild state action agent environme

2.3.1 Environment

Das Environment beschreibt das gesamte Problem dass der Agent versucht zu lösen. Es beinhaltet alles was für das Spielen wichtig ist außer dem Spieler welcher von dem Agenten behandelt wird. Dies ist bei den meisten Spielen ein Spielfeld mit den Positionen aller Akteure, Spielsteine oder sonstige Elemente die zum Spielen genutzt werden. Dazu kümmert sich das Environment auch noch um die Spiellogik. Die Spiellogik beschreibt alle Abläufe die durch die Regeln des Spiels definiert wurden. In dem Beispiel Vier Gewinnt wäre das unter anderem die Funktion dass die Spielsteine immer auf das untereste freie Feld fallen oder die überprüfung ob ein Spieler Gewonnen hat. Das letzte wichtige Element im Environment der meisten Spiele ist der Gegenspieler welcher im Falle von physikalischen Spielen normalerweise die selben Möglichkeiten besitzt wie der Spieler. Für das Verhalten des Gegenspielers werden im Normalfall Algorithmen genutzt die den bestmöglichen Zug errechnen, wie hier der Minmax Algorithmus. Für sehr komplexe Spiele bei denen sich kein optimalerZug berechnen lässt wird oft auf menschliche Spieler zurückgegriffen. Da ein Computer aber viel schneller im spielen ist und reinforcement learning viele Durchläufe absolvieren muss um Wirkung zu zeigen wird versucht dies zu vermeiden oder bestehende Datensätze von Spielen mit Menschen genutzt.
TODO Bsp Alpha Star etc.

2.3.2 Agent

Bei dem Agenten handelt es sich um den Akteur der die ihm gegebene Aufgabe meistern soll. Er sieht das Environment und wählt mittels der Policy die ihm am besten erscheinende Action aus. https://es.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html

2.3.3 Policy

Mit der Policy wird das Verhalten beschrieben nach dem der Agent entscheidet welche der möglichen Aktionen die aktuell beste ist um den Reward zu maximieren.
TODO

2.3.4 State

Der State beschreibt den aktuellen Zustand des Environments welcher in jedem Zug an den Agenten weitergereicht wird damit dieser seine Auswahl treffen kann.

2.3.5 Actions

Actions sind die möglichen Aktionen zwischen denen sich der Agent je nach State entscheiden muss. Diese sind alle möglichen Züge die ein Spieler zu einer bestimmten Zeitpunkt ausführen kann.

2.3.6 Reward

Der Reward ist die Belohnung die der Agent für seine Aktionen bekommt. Diese Belohnung beschreibt wie gut es ist eine bestimmte Aktion auszuführen. Hieran passt der Agent dann wärend des Trainings seine Policy an. Der Reward wird also benutzt um das Verhalten des Agenten zu beeinflussen. Möchte man ein bestimmtes verhalten wird hierfür ein positiver Reward vergeben möchte man ein anderes Verhalten nicht wird ein negativer Reward vergeben. Je besser die Reward-Funktion gewählt wird desto eher wird der Agent gewolltes Verhalten zeigen. Bei der Wahl einer Reward-Funktion wird zwischen kontinuierlichen und diskreten Funktionen unterschieden.

> Continuous vs Discrete Rewards hier Discrete? Every Step vs at the end
> TODO

2.3.7 Q-Funktion

TODO

2.3.8 Markov Decision Process

TODO

2.3.9 psychologischer Hintergrund

Reinforcement learning funktioniert nach dem Prinzip von Trial-and-Error.
TODO

2.4 Software

Um die Problemstellung zu bearbeiten wurde eine Implementierung des Spiels in Python genutzt welches als ein Gym Environment fungiert. An diesem Environment trainiert dann ein Agenten der mit Hilfe von Keras erschaffen wurde. TODO genaue versionen

2.4.1 Python

Als Programmiersprache wurde sich für Python entschieden da ein gutes Vorwissen in dieser Sprache vorhanden war und alle für diese Arbeit wichtigen Packages in dieser Programmiersprache existieren. Der Schöpfer von Python beschreibt es selbst als einfach zu erlernende objektorientierte Programmiersprache welche Leistung mit einer klaren Syntax verbindet.
TODO https://dl.acm.org/doi/book/10.5555/1593511

2.4.2 Tensorflow

Bei Tensorflow handelt es sich um ein open source Framework für Maschine-Learning welches ursprünglich für den internen Bedarf bei Google, für zum Beispiel Spracherkennung oder Google Maps, entwickelt wurde. TODO mehr?

2.4.3 Keras

Keras ist eine open source Bibliothek die auf Tensorflow aufbaut welche die möglichkeit bietet vereinfacht neuronale Netze zu erstellen um diese in reinforcement learning Algorithmen zu benutzen. TODO mehr?

2.4.4 OpenAI Gym

Gym ist ein Werkzeug zum entwickeln und vergleichen von reinforcement learning Algorithmen welches von OpenAI, einem Forschungslabor aus San Francisco, zur Verfügung gestellt wird. Es bietet eine angleichende Schnittstelle zwischen dem Environment und dem Agenten. Hierdurch können neue Agenten und Environments nach gewissen Vorgaben erstellt werden. Somit ist es mit Gym einfacher möglich verschiedene Agenten an einem Environment zu Trainieren und diese zu vergleichen oder den selben Agenten an verschiedenen Environments auf seine Anpassungsfähigkeit zu testen.

Kapitel 3

Konzept

In diesem Kapitel wird beschrieben was mit dieser Bachelorarbeit erreicht werden soll und wie dafür vorgegangen werden soll.TODO

3.1 Plan

TODOmehr
Es wird ein Programm geschrieben, dass durch reinforcement learning erlernt das Spiel Vier Gewinnt zu meistern. Des weiteren werden dann aus diesem Lernprozess mehrere Versionen des gelernten Agenten extrahiert welche unterschiedlich lange trainiert haben. Es soll dann mit einer Studie in der Menschen gegen die verschiedenen Varianten antretten beobachtet werden ob sich diese für Computergegner eignen und somit automatisch verschieden starke Computergegner mit hilfe von reinforcement learning erstellt werden können.keine Studie mehr

Kapitel 4

Implementierung

Vorgehensweise bei der Implementierung

DQN Algoritmus zum lernen des Spiels

Trainiert gegen Random/Minmax

Auswertung: ???

Es wurde sich für eine bestehende grundlegende Implementierung entschieden welche schon die Spiellogik von Vier Gewinnt sowie einem sehr einfachen Reinforcement Learning Ansatz enthält. In dieser Version lernt der Agent gegen einen Gegner zu gewinnen der zufällige Züge macht.
Diese Implentierung wurde um einen Gegner erweitert der seine Züge nach dem Minmax Algorithmus auswählt.
Weitergehend wurde die Reinforcement Learning Implementierung überarbeitet um gegen den Minmax Algorithmus gewinnen zu können.
Diese Implementierungen werden hier einmal genauer beschrieben:

4.1 Minmax

4.2 Reinforcement Learning

TODO link

4.2.1 Environment

TODOIn Vier Gewinnt ist dies das Spielfeld und der Gegner.

4.2.2 Agent

TODO Hier ist der Agent einer der beiden Spieler von Vier Gewinnt.

4.2.3 Policy

TODO

4.2.4 State

TODO Für Vier Gewinnt beschreibt der State welche der 42 Felder mit welchen Steinen befüllt sind.

4.2.5 Actions

TODO Die Actions in Vier Gewinnt beschreiben die sieben Löcher in die der Agent Steine werfen kann.

4.2.6 Reward

TODOIn der ursprünglichen Implementierung wird der Reward immer zum Ende einer Partie ausgegeben und beträgt beim Gewinn 1 beim Verlieren -1 und wenn die Partie unentschieden ausgeht 0. Dies wurde so erweitert, dass die Anzahl der Züge mit in die Bewertung einbezogen werden. Hierfür wird der Reward durch die Anzahl der Züge geteilt. Da es nicht möglich ist vor dem vierten Zug zu gewinnen wird erst ab diesem Zug gezählt. Hierdurch wird das schnelle Gewinnen mehr belohnt und das schnelle Verlieren mehr bestraft.
TODO credit assignment problem (bewertung nur am ende)
TODO Bild von der Rewardkurve

4.2.7 Q-Funktion

TODO

4.3 Lernen vom Gegner

TODO

Kapitel 5

Evaluierung

Was kann evaluiert werden? Muss jetzt anhand der veränderung vom Agneten gemacht werden jede veränderung vergleichen

5.1 Auswertung

Ergebnisse in Text übersetzen-> Graph, Durchschnitt,...
Schlüsse die aus den Daten gezogen werden können -> Signifikantstest?

Kapitel 6

Fazit

TODO
Was wurde festgestellt auswertung ausblick

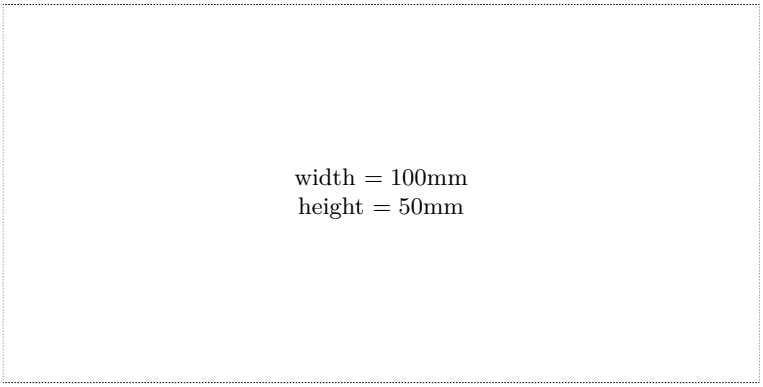
6.1 Besonderheiten

TODO

Quellenverzeichnis

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —