Untersuchung der Lernfähigkeit verschiedener Verfahren am Beispiel von Computerspielen

Abschlussarbeit
zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

Thilo Stegemann
s0539757
Angewandte Informatik

21. Januar 2017

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Erstprüfer: Prof. Dr. Burkhard Messer
Zweitprüferin: Prof. Dr. Adrianna Alexander

Todo list

Abkürzungsverzeichnis

ID Identifikator

UI User Interface

vgl. Vergleich

vs. Versus, Gengenüberstellung

Abbildungsverzeichnis

Tabellenverzeichnis

Inhaltsverzeichnis

Kapitel 1

Projektvision

1.1 Motivation

Sind Sie ein (angehender) Softwareentwickler und programmieren aktuell ein Computerspiel, welches lernfähige Verfahren unterstützen soll? Benötigen Sie innerhalb einer beliebigen Anwendung einen lernfähigen Algorithmus und sie kennen die Schwächen, Stärken, Grenzen und Anwendungsgebiete der Lernverfahren nicht?

Haben Sie sich auch schon mal eine der nachfolgenden Fragen gestellt oder interessieren Sie diese Fragen generell?

Wie lernt ein Programm Strategien? Was sind die elementaren Schritte die ein Programm während des Lernprozesses durchläuft? Wie anwendbar und leistungsfähig sind die Lernverfahren hinsichtlich verschiedener Spielgrundlagen? In wie fern wird ein Lernverfahren von einem Computerspiel ausgereizt? Wenn zwei unterschiedliche Lernverfahren untersucht und verglichen werden, welches Lernverfahren ist dann effizienter, schneller oder besser?

Diese wissenschaftliche Arbeit könnte dann sehr interessant für Sie sein. Innerhalb dieser Arbeit werden bestimmte Lernverfahren, am Beispiel verschiedener Computerspiele, auf Ihre Funktionsweise, Schwächen, Stärken und Grenzen untersucht, implementiert, und getestet.

1.2 Vorläufige Zielsetzung

Das Ziel der Arbeit ist die Untersuchung des Lernverhaltens, der Grenzen, der Schwächen und der Stärken verschiedener Lernverfahren am Beispiel von mindestens zwei eigens implementierten Computerspielen. Die Lernverfahren sollen trainiert werden und dadurch mehr oder weniger eigenständige Siegesstrategien und Spielzugmuster entwickeln. Die Lernverfahren könnten sich gegenseitig trainieren

oder sie trainieren indem sie gegen einen Menschen spielen. Der Fokus der wissenschaftlichen Arbeit liegt hierbei auf der Untersuchung der verschiedenen Lernverfahren und nicht auf der Implementierung besonders komplexer Computerspiele, daher sollen nur sehr simple Computerspiele implementiert werden. Ein vollständiges Dame Spiel wird zum Beispiel nicht implementiert, aber eine absichtlich verkleinerte Dame Variante mit veränderten Spielregen, für ein schnelleres Spielende, wäre durchaus möglich. Zudem wären auch ein vier mal vier Tic-Tac-Toe ein Vier Gewinnt oder ein Black Jack Computerspiel

1.3 Nutzen/Zweck des Projektes

1.4 Zielgruppe

Kapitel 2

Grundlagen

Schreiben der Einführung des Kapitels ?? Grundlagen

2.1 Verstärkendes Lernen(Reinforcement Learing)

Beim bestärkenden Lernen ist der Lerner ein entscheidungstreffender Agent, der in einer Umgebung Handlungen ausführt und Belohnung (oder Bestrafung) für seine Aktionen beim Versuch, das Problem zu lösen, erfährt. Nach einer Menge an Versuch-und-Irrtum-Durchläufen sollte er die beste Vorgehensweise lernen, welche der Sequenz an Aktionen entspricht, durch welche die Gesamtbelohnung maximiert wird[Alp08, S. 397].

Ein Agent führt Entscheidungen (Aktionen) innerhalb einer ihm unbekannten Umgebung aus. Der Agent soll in dieser Umgebung ein Ziel erreichen. Dieses Ziel ist oftmals die einzige Belohnung für den Agenten. Ob ihn seine Aktionen dem Ziel näher bringen oder ihn vom Ziel entfernen ist dem Agenten nicht bekannt. Daher probiert und scheitert der Agent solange, bis er eine Möglichkeit findet, sein Ziel zu erreichen. Ein Problembereich des verstärkenden Lernens ist es, diese gefunden Möglichkeit, sprich die zielführenden Aneinanderreihung von Aktionen, zu optimieren.

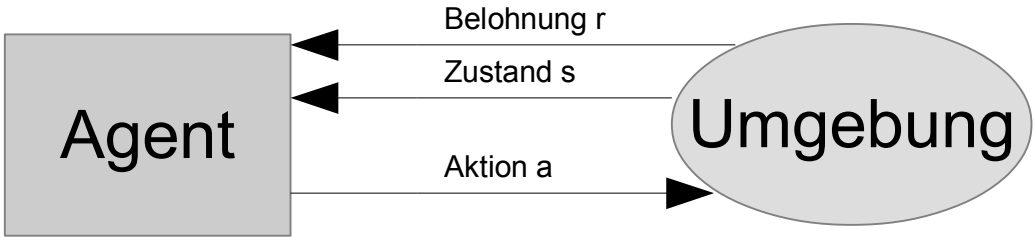

Abbildung 2.1 Der Agent und seine Wechselwirkung mit der Umgebung

Verstärkendes oder auch bestärkendes Lernen beschäftigt sich mit dem Problem, dass ein entscheidungstreffender Agent innerhalb einer unbekannten Umgebung Aktionen (Entscheidungen) ausführen soll (siehe Abbildung 2.1 [Alp08, vgl. 398] und [Ert16, vgl. 290]). Der Agent lernt dann, durch Belohnung oder Bestrafung, seine Aktionen, hinsichtlich seiner Zielstellung, an die Zustände der Umgebung anzupassen. Die Belohnung oder Bestrafung erfolgt jedoch meistens erst nach einer Sequenz von Aktionen, daher ist dem Agenten nicht bekannt ob eine einzelne Aktion positiv oder negativ, hinsichtlich der Erreichung seiner Ziele, ist. Die Optimierung einer solchen Aktionssequenz ist besonders schwierig und daher ein bedeutsames Problem des verstärkenden Lernens.

Ein Agent im Labyrinth

Nehmen wir an es existiere folgender Agent. Dieser Agent kann vier Aktionen ausführen, bewege dich nach oben, unten, rechts oder links. Er wird in einem ihm unbekannten Labyrinth ausgesetzt. Das Labyrinth ist die Umgebung und die Zustände der Umgebung verändern sich durch die Aktionen des Agenten, das heißt verändert der Agent seine Position innerhalb des Labyrinths, dann wechselt er von einem Ausgangszustand, durch eine Aktion, in einen neuen Zustand der Umgebung.

Der Agent lernt also, dass die Aktion 'bewege dich nach oben' den Ausgangszustand in einen neuen Zustand transformiert. Führt der Agent die Aktion 'bewege dich nach oben' aus, dann ist jedoch die Zustandsveränderung abhängig von der individuellen Umgebung in der sich der Agent befindet. In einem Labyrinth kann der Agent nicht immer alle seiner vier Aktionen ausführen, denn er ist umringt von Mauer die seinen Aktionsradius beschränken. Würde er trotzdem eine unzulässige Aktion ausführen, dann verändert sich der Zustand der Umgebung nicht, denn der Agent würde sprichwörtlich gegen die Wand fahren. Das bedeutet für den Agenten, dass er genau differenzieren muss in welchem Zustand er welche Aktion durchführt und wie er sein Ziel erreichen kann. Das Ziel des Agenten, in diesem Beispiel, ist das erreichen des Labyrinth Ausgangs.

Nach einer endlichen Sequenz von Aktionen gelingt es dem Agenten den Ausgang des Labyrinths zu erreichen und er erhält eine nummerische Belohnung für die Zielerreichung. Der Weg, sprich die Sequenz von Aktionen, den der Agent, durch probieren und scheitern, gewählt hat wird nicht der schnellstmögliche Weg sein und auch nicht der kürzeste. Wie kann dies Aktionssequenz hinsichtlich der Zielerreichung optimiert werden?

2.1.1 Überwachtes vs. verstärkendes Lernen

Im Gegensatz zu den Lernverfahren beim überwachten Lernen fehlt dem Agenten beim verstärkenden Lernen ein Lehrer der dem Agenten genau sagt ob seine Aktion richtig oder falsch ist. Zudem wäre ein Lehrer, der dem Agenten für jeden Zustand genau sagt welches die richtige Aktion ist, sehr kostspielig. Oftmals ist für einen Zustand gar keine optimale Aktion möglich, erst die Aneinanderreihung von Aktionen und Zustandsübergängen kann optimal sein oder optimiert werden[Alp08, vgl. 397].

Ein überwachtes Lernverfahren wird zuerst mittels eines Trainingssets kontrolliert belehrt. Das Trainingsset für ein überwachtes Lernverfahren besteht aus verschiedenen Eigenschaften (Features) und einer den Eigenschaften zugeordneten Klasse beziehungsweise Zielvariable (Target Variable). Die Qualität der Zielwerte kann mittels Testsets ermittelt werden. Ein Testset ist ein Datenset bestehend aus Eigenschaften ohne die dazugehörigen Klassen. Abbildung 2.2 zeigt, wie ein Trainingsset aussehen könnte[Har12, S. 8]. Die Spalten Gewicht, Flügelspanne, Schwimmhäute und Rückenfarbe sind die Eigenschaften und die Spalte Spezies beinhaltet die Zielvariablen. Das überwachte Lernverfahren lernt mittels des Trainingsdatensets die Beziehungen zwischen den Eigenschaften und der Klassen.

	Gewicht (g)	Flügelspanne (cm)	Füße mit Schwimmhäuten?	Rückenfarbe	Spezies
1	1000,1	125,0	Nein	Braun	Buteo jamaicensis
2	3000,7	200,0	Nein	Grau	Sagittarius serpentarius
3	3300,0	220,3	Nein	Grau	Sagittarius serpentarius
4	4100,0	136,0	Ja	Schwarz	Gavia immer
5	3,0	11,0	Nein	Grün	Calothorax lucifer
6	570,0	75,0	Nein	Schwarz	Campephilus principalis

Abbildung 2.2 Vogelspezies Klassifikation basierend auf vier Eigenschaften

Verstärktes Lernen umfasst die Probleme, bei denen in der Regel kein solches Trainingsdatenset vorliegt, welches genau festlegt ob eine Aktion in einem Zustand korrekt oder falsch ist. Bezogen auf ein Testdatenset wäre eine Aktion des Agenten, dass nennen einer Klasse hinsichtlich der gelernten Zusammenhänge aus den Trainingsdaten. Teilt man die Anzahl der korrekten Vorschläge durch die Anzahl aller Versuche, dann erhält man eine Kennzahl für die Qualität der Vorhersage. Sind alle Klassen der Testinstanzen richtig vorhergesagt, dann ist die Kennzahl genau Eins. Jede Zeile eines Testsets und jede Zeile eines Trainingssets ohne die Zielvariablen ist eine Instanz. Verstärkendes Lernen kann trotzdem vom überwachten Lernen profitieren, denn es ist möglich den Agenten in der Anfangsphase des verstärkten

Lernens explizit zu programmieren und ihn dadurch auf bestimmte Auffälligkeiten oder Muster aufmerksam zu machen. Ist es zu kompliziert dies explizit zu programmieren, dann kann auch ein Mensch dem Agenten die richtigen Aktionen vorgeben.

Sehr nützlich werden diese beiden Unterstützungen der Anfangsphase des verstärkenden Lernens, sobald die Dimensionen der Umwelt oder des Agenten eine bestimmte Größe überschreiten. Eine Aktionsdimension kann sehr wenige Aktionen beinhalten zum Beispiel die vier Aktionen bewege dich nach oben, unten, rechts oder links. Roboter die dem Menschen nachempfunden sind verfügen über bis zu 50 verschiedene Motoren für die einzelnen Gelenke. Diese müssen gleichzeitig angesteuert werden, was zu einem 50-dimensionalen Zustandsraum und einem 50-dimensionalen Aktionenraum führt[Ert16, vgl. 305 f.]. Bei solch großen Dimensionen können die Laufzeiten einiger verstärkender Lernverfahren massiv ansteigen, bis sie praktisch nicht mehr anwendbar sind. Gerade in der Anfangsphase des verstärkten Lernens kann darum ein Eingriff mittels überwachtem Lernen sehr Laufzeit schonend sein.

2.1.2 Das Strategiespiel Schach

Das Strategiespiel Schach hat viele Ähnlichkeiten mit dem Strategiespiel Reversi und auch Ähnlichkeiten zum Spiel Tic Tac Toe, darum wird nachfolgend die Anwendungsmöglichkeit des verstärkenden Lernens auf das Schachspiel ausführlicher behandelt.

Schachspiel Komplexität

Während eines Schachspiels gibt es für eine typische Stellung über 30 mögliche Züge und eine durchschnittliche Dauer von 50 Halbzügen ist noch relativ kurz. Würden wir einen Suchbaum für ein Schachspiel aufstellen wollen, dann hätte dieser Suchbaum?

Suchbaum

$$\sum_{d=0}^{50} 30^d = \frac{1 - 30^{51}}{1 - 30} = 7,4 \times 10^{73} \tag{2.1}$$

Blattkonten. Angenommen wir hätten 10.000 Computer, von denen jeder eine Milliarde Schlussfolgerungen pro Sekunde schaffen würde und wir könnten die Arbeit ohne Verlust auf alle Rechner verteilen. Die gesamte Rechenzeit für $7,4 \times 10^{73}$ Schlussfolgerungen wäre dann $\approx 2,3 \times 10^{53}$ Jahre, was wiederum etwa 10^{43} mal so lange dauert wie unser Universum alt ist[Ert16, S. 93 f.].

Schach ist also äußert komplex und die Dimensionen der Aktions- und Zustandsräume sind sehr groß. Diverse Suchalgorithmen würden an dieser Komplexität scheitern, weil die Rechenzeit dieser Algorithmen exponentiell zu den Dimensionen ansteigt.

2.2 Spielentwicklung

2.3 Lineare Algebra

2.4 Heuristik

Kapitel 3

Problemanalyse und Anforderungsdefinition

In diesem Kapitel:

Grundlagen maschinelles Lernen einfügen in Kapitel 2!

Schreibe die Einführung von Kapitel 3!

3.1 Problemanalyse

3.1.1 Die Problematik

Welches lernfähige Verfahren ist auf die Brettspiele anwendbar?
Welche Daten müssen die Brettspiele liefern, sodass die lernfähigen Verfahren diese Daten verwenden können?
Wie muss das Format dieser Daten angepasst werden?
Wie wird eine Spielsituation dargestellt?

Was genau ist die Problematik?
Kurzgefasst sollen innerhalb dieser wissenschaftlichen Abschlussarbeit lernfähige Verfahren mittels Daten von Brettspielen trainiert werden. Nach Abschluss der Trainingsphase sollen die lernfähigen Verfahren bestenfalls einen menschlichen Gegenspieler schlagen können. Schlussendlich soll die Lernfähigkeit dieser Verfahren untersucht werden. Um diese größere Problematik zufriedenstellend zu bearbeiten müssen wir sie in kleinere Teilprobleme aufteilen.

Das erste Teilproblem ist die Auswahl der lernfähigen Verfahren. Die Schwierigkeit hierbei besteht in der Kompatibilität der Verfahren zu den von den Brettspielen produzierten Daten und in der Anzahl der zur Verfügung stehenden lernfähigen Verfahren. Es existieren drei Gattungen von lernfähige Algorithmen die ihre stärken und schwächen haben und die nur auf bestimmte Daten angewendet werden können. Eine genauere Beschreibung dieser Gattungen ist im Unterabschnitt **??** die-

ser Arbeit vorhanden.

Das erste Teilproblem führt unmittelbar zum zweiten Teilproblem. Welche Daten können die zu implementierten Brettspiele liefern und wie sollten diese Daten strukturiert werden? Sollten Daten in Form von Eigenschaften und Zielwerten generiert werden können, dann sind überwachte lernfähige Algorithmen für Klassifizierung möglicherweise eine gute Wahl.

Spiele für zwei Spieler wie zum Beispiel Schach, Dame, Reversi oder Go sind deterministisch, denn jede Aktion(Spielzug) führt bei gleichem Ausgangszustand immer zum gleichen Nachfolgezustand. Im Unterschied dazu ist Backgammon nichtdeterministisch, denn hier hängt der Nachfolgezustand vom Würfelergebnis ab. Diese Spiele sind alle beobachtbar, denn jeder Spieler kennt immer den kompletten Spielzustand.[Ert16, S. 114]

Beide Brettspiele werden durch das Modell des endlichen Horizonts(engl. finite-horizon model) Das Modell des optimalen Verhaltens?

Sebastian Raschka schreibt: Ein schönes Beispiel für verstärkendes Lernen ist ein Schachcomputer. Hier bewertet der Agent nach einer Reihe von Zügen die Stellung auf dem Schachbrett(die Umgebung), und die Belohnung kann am Ende des Spiels als Sieg oder Niederlage definiert werden.[Ras16, S. 27]

3.1.2 Bereits existierende Softwarelösungen

3.1.3 Gegenstandsbereich des Projektes abgrenzen

3.1.4 Abgrenzung gegenüber der künstlichen Intelligenz

3.2 Anforderungsdefinition

Innerhalb dieses Abschnittes werden die funktionalen und nicht-funktionalen Anforderungen für die Software definiert. Die funktionalen Anforderungen sollen das Verhalten der Software festlegen, sprich welche Aktionen die Software ausführen kann. In verschiedenen Softwareprojekten können sich die funktionale Anforderungen stark voneinander unterscheiden, dahingegen sind nicht-funktionale Anforderungen in verschiedenen Softwareprojekten oftmals sehr ähnlich.
Nichtfunktionale Anforderungen beschreiben wie gut eine Software seine Leistung erbringen soll.

3.2.1 Funktionale Anforderungen

Eine User Interface (UI) ist eine Benutzerschnittstelle diese definiert die Interaktion zwischen dem Benutzer der Anwendung und der Anwendung. Die Identifikator

Tabelle 3.1 Funktionale Anforderungen

ID	Titel	Beschreibung
1	Brettspiele	Es sollen zwei Brettspiele implementiert werden.
2	Brettspiel UI	Es soll eine Benutzersteuerung für die Brettspiele implementiert werden.
3	Gleichartigkeit der Brettspiele	Die Spielweise der Brettspiele soll hinsichtlich bestimmter Punkte gleich sein.
4	Eindeutigkeit der Brettspiele	Die Brettspiele sollen immer einen Gewinner und einen Verlierer oder ein Unentschieden hervorbringen.
5	Endlichkeit der Brettspiele	Die Brettspiele sollen immer nach einer angemessenen endlichen Anzahl von Spielzügen enden.
6	Spieleranzahl der Brettspiele	Die Brettspiele sollen genau von zwei Spielern in einer direkten Konfrontation(Eins-gegen-Eins-Situation) ausgetragen werden.
7	Lernfähige Verfahren	Es sollen zwei lernfähige Verfahren implementiert werden. Diese Verfahren sollen Strategien, wie die Brettspiele gewonnen werden können, entwickeln.
8	Training der Lernverfahren	Die lernfähigen Verfahren sollen durch einen menschlichen Spieler trainiert werden oder durch das jeweils andere Lernverfahren.
9	Wissen Speichern	Das von den Lernverfahren ermittelte Wissen(z.B. mögliche Spielzüge, Gewichtungen besserer und schlechterer Spielzüge) soll persistent gespeichert werden.

(ID) soll die nachfolgend aufgestellten Anforderungen eindeutig identifizieren.

3.2.2 Nicht-funktionale Anforderungen

Tabelle 3.2 Nichtfunktionale Anforderungen

ID	Titel	Beschreibung
1	Programmlogik	Die Programmlogik der Brettspiele und der lernfähigen Verfahren soll in der Programmiersprache Python geschrieben sein.
2	Testen der Programmlogik	Die Tests der Programmlogik sollen mit den, in der Standard-Bibliothek von Python enthaltenen, Modultests(Unit testing framework) durchgeführt werden.
3	Brettspielgrafik	Die Grafik der Brettspiele soll mit der Bibliothek PyGameïmplementiert werden.

Kapitel 4

Modellierung und Entwurf

In diesem Kapitel werden die funktionalen Anforderungen aus dem Abschnitt 3.2 spezifiziert. Modelle für die einzelnen funktionalen Anforderungen sollen entwickelt werden. Die Modelle veranschaulichen das geforderte funktionale Verhalten der Software. Voraussetzung für die Entwicklung der Modelle ist die Konkretisierung der funktionalen Anforderungen. Diese Konkretisierung beinhaltet die Festlegung der Bestandteile die für eine Implementierung der funktionalen Anforderung benötigt werden. Ziel des Kapitels ist es, die wichtigsten Bestandteile einer funktionalen Anwendung herauszubilden, dieses Bestandteile zu definieren und zu veranschaulichen.

4.1 Computerspiele

In diesem Abschnitt werden die Bestandteile der Computerspiele festgelegt und veranschaulicht.

4.1.1 Tic Tac Toe

Das klassische Tic Tac Toe ist ein Spiel, welches mit genau zwei Spielern gespielt wird. Jeder dieser Spieler zeichnet abwechselnd entweder ein Kreuz oder einen Kreis in eine Matrix auf ein Blatt Papier. Während eines gesamten Spiels darf ein Spieler nur Kreuze zeichnen und der andere Spieler nur Kreise. Das Spielfeld ist eine drei mal drei große Matrix, also können maximal neun Symbole in diese Matrix eingetragen werden. Um die Anzahl der möglichen Spielzüge zu erhöhen wird das Spielfeld des klassischen Tic Tac Toe auf eine vier mal vier Matrix erweitert.

Spielregeln

Ziel des Spiels ist es vier Kreuze oder vier Kreise in einer bestimmten Position anzuordnen. Im nachfolgenden wird davon ausgegangen, dass der menschliche Spieler Kreuze verwendet und der Computergegner Kreise. Die Kreise und Kreuze sind

Spielfiguren, welche den jeweiligen Spieler repräsentieren. Der menschliche Spieler hat zusätzlich, in den Nachfolgenden Siegesszenarien, das Anrecht auf den ersten Zug. Es existieren drei unterschiedliche Anordnungen von Spielfiguren, die das Spiel beenden und einen Sieg herbeiführen. Gewinnt ein Spieler mit einer Siegesanordnung seiner Spielfiguren, dann verliert der andere Spieler dadurch automatisch.

Eine horizontale Siegesanordnung entsteht, wenn vier Spielfiguren eines Spielers in einer horizontalen Reihe, veranschaulicht in Abbildung 4.1, angeordnet sind. In jeder Reihe des Spielbretts ist ein horizontaler Sieg möglich.

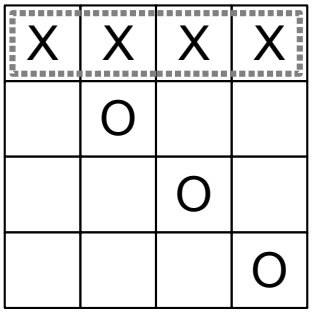

Abbildung 4.1 Tic Tac Toe horizontaler Sieg

In Abbildung 4.2 gewinnt der menschliche Spieler knapp gegen den Computergegner mit einer ununterbrochenen vertikalen Reihe. Der Computergegner hätte fast eine diagonale Reihe aus Kreisen verbunden, die jedoch von dem menschlichen Spieler mit einer Spielfigur geblockt wurde. Zudem hätte der Computergegner auch fast eine vertikale Reihe ohne Unterbrechungen vervollständigt.

Die dritte und letzte Anordnungsvariante der Spielfiguren, welche zu einem Sieg eines Spielers führt, ist die diagonale Verbindung von vier Spielfiguren eines Spielers. In Abbildung 4.3 gewinnt der Computergegner mit einer diagonalen Anordnung von vier Spielfiguren ohne Unterbrechung einer gegnerischen Spielfigur.

Insgesamt existieren vier vertikale, vier horizontale und zwei diagonale Anordnungen der Spielfiguren, welche einen Sieg herbeiführen würden, also zehn verschiedene Siegesanordnungen. Was passiert jedoch, wenn keine der zehn möglichen Siegesanordnungen auftritt?

Dann gewinnt bzw. verliert keiner der beiden Spieler und es entsteht ein Unentschieden. Sind die beiden Kontrahenten gleich gut, erfahren oder verwenden die selben Strategien, dann tritt ein Unentschieden möglicherweise öfter oder andauernd ein.

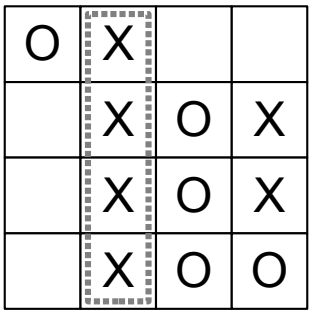

Abbildung 4.2 Tic Tac Toe vertikaler Sieg

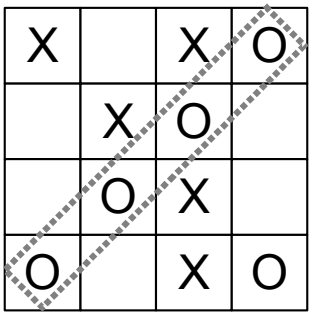

Abbildung 4.3 Tic Tac Toe diagonaler Sieg

Benutzerschnittstellen

Der Benutzer kann seine Kreuze auf das Spielbrett setzen indem er ein vorher vom Spiel definiertes Zahlentupel über die Tastatur eingibt. Welche Zahlentupel ein Kreuz an welche Stelle setzt ist in Abbildung 4.4 definiert. Sollte der menschliche Spieler keines der erlaubten Zahlentupel eingeben, dann wird er darauf hingewiesen, welche Steuerungsmöglichkeiten zum setzen der Spielfiguren ihm zur Verfügung stehen.

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

Abbildung 4.4 Tic Tac Toe Spielfiguren setzen

4.1.2 Reversi

Spielprinzipien

Spielregeln

Benutzerschnittstellen

4.2 Lernverfahren

4.2.1 Analyse und Auswahl der lernfähigen Algorithmen

4.2.2 Anwendung der Algorithmen auf Computerspiele

4.2.3 Konzeptuelles Training der Algorithmen

4.2.4 Persistenz der Trainingsdaten

Kapitel 5

Implementierung

In diesem Kapitel: //TODO Einführung in das Kapitel

5.1 Computerspiele

5.2 Lernverfahren

5.3 Alternative Lernverfahren

Kapitel 6

Validierung

In diesem Kapitel: //TODO Einführung in das Kapitel

6.1 Messbare Testkriterien

6.2 Modultests

6.2.1 TicTacToe

6.2.2 Reversi

6.2.3 Lernverfahren 1

Empirisches Protokoll

6.2.4 Lernverfahren 2

Empirisches Protokoll

6.2.5 Persistenz

6.3 Systemtest

Kapitel 7

Auswertung

In diesem Kapitel: //TODO Einführung in das Kapitel

7.1 Belastbarkeit und Grenzen der Lernverfahren

7.2 Optimale Anwendungsspiele für die Lernverfahren

7.3 Gegenüberstellung der Lernverfahren

7.4 Bewertung der Strategien

7.5 Menschlicher oder mechanischer Trainer?

Literatur

[Alp08] Ethem Alpaydin. *Maschinelles Lernen*. 1. Aufl. Oldenbourg, 2008.

[Bei14] Christoph Beierle. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. 5. Aufl. Springer, 2014.

[Ert16] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz: Eine prakmatische Einführung*. 4. Aufl. Springer, 2016.

[Har12] Peter Harrington. *Machine Learning: IN ACTION*. 1. Aufl. Manning, 2012.

[Lö93] Jan Löschner. *Künstliche Intelligenz: Ein Handwörterbuch für Ingenieure*. 1. Aufl. VDI, 1993.

[Ras16] Sebastian Raschka. *Machine Learning mit Python*. 1. Aufl. MIT Press, 2016.

[Rus12] Stuart J. Russell. *Künstliche Intelligenz: Ein moderner Ansatz*. 3. Aufl. Pearson, 2012.

Anhang A

I am an appendix!

Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country, in which roasted parts of sentences fly into your mouth. Even the all-powerful Pointing has no control about the blind texts it is an almost unorthographic life One day however a small line of blind text by the name of Lorem Ipsum decided to leave for the far World of Grammar. The Big Oxmox advised her not to do so, because there were thousands of bad Commas, wild Question Marks and devious Semikoli, but the Little Blind Text didn't listen. She packed her seven versalia, put her initial into the belt and made herself on the way. When she reached the first hills of the Italic Mountains, she had a last view back on the skyline of her hometown Bookmarksgrove, the headline of Alphabet Village and the subline of her own road, the Line Lane. Pityful a rethoric question ran over her cheek, then

A.1 Section in appendix!

Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country, in which roasted parts of sentences fly into your mouth. Even the all-powerful Pointing has no control about the blind texts it is an almost unorthographic life One day however a small line of blind text by the name of Lorem Ipsum decided to leave for the far World of Grammar. The Big Oxmox advised her not to do so, because there were thousands of bad Commas, wild Question Marks and devious Semikoli, but the Little Blind Text didn't listen. She packed her seven versalia, put her initial into the belt and made herself on the way. When she reached the first hills of the Italic Mountains, she had a last view back on the skyline of her

hometown Bookmarksgrove, the headline of Alphabet Village and the subline of her own road, the Line Lane. Pityful a rethoric question ran over her cheek, then

Anhang B

Another appendix? What the heck?

B.1 Section in appendix!

Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country, in which roasted parts of sentences fly into your mouth. Even the all-powerful Pointing has no control about the blind texts it is an almost unorthographic life One day however a small line of blind text by the name of Lorem Ipsum decided to leave for the far World of Grammar. The Big Oxmox advised her not to do so, because there were thousands of bad Commas, wild Question Marks and devious Semikoli, but the Little Blind Text didn't listen. She packed her seven versalia, put her initial into the belt and made herself on the way. When she reached the first hills of the Italic Mountains, she had a last view back on the skyline of her hometown Bookmarksgrove, the headline of Alphabet Village and the subline of her own road, the Line Lane. Pityful a rethoric question ran over her cheek, then

Acknowledgements

Acknowledgements go to the back!