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

27. 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.

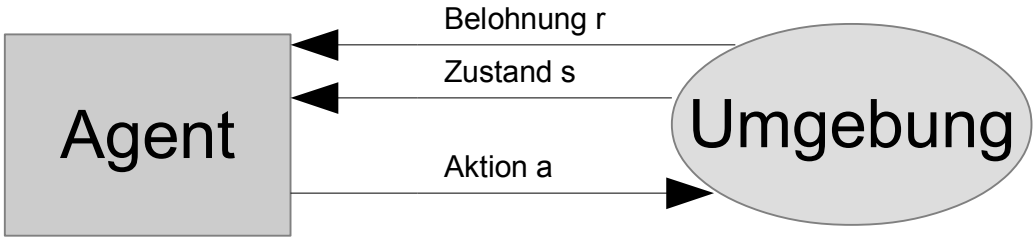

Abbildung 2.1 Der Agent und seine Wechselwirkung mit der Umgebung

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 Die optimale Strategie

Der Zustand $s_t \in S$ beschreibt die Position eines Agenten in einer Umgebung zu einem festgelegten Zeitpunkt t. S ist eine menge von möglichen Zuständen. Die Aktion $a_t \in A$ wird vom Agenten, zum Zeitpunkt t, ausgeführt und verändert die Umgebung(siehe Abbildung 2.1). Diese Aktion überführt den vorherigen Zustand s_t in einen neuen zustand $s_t + 1$. Der neue Zustand $s_t + 1 = \delta(s_t, a_t)$ wird von der Übergangsfunktion δ bestimmt. Die Übergangsfunktion wird von der Umgebung bestimmt und kann von dem Agenten nicht beeinflusst werden. Führt ein Agent eine Aktion a in einem Zustand s zu einem Zeitpunkt t aus, dann erhält der Agent eine direkte Belohnung(engl. immediate reward) $t_t = r(s_t, a_t)$. Diese direkte Belohnung ist abhängig von dem aktuellen Zustand und der ausgeführten Aktion.

Viele praktische Probleme hingegen haben für jede Aktion in jedem Zustand keine direkte Belohnung. Bei einem Schachspiel zum Beispiel erhält der Agent erst eine positive Belohnung, wenn das Spiel gewonnen ist oder eine negative Belohnung, wenn das Spiel verloren ist. Die zahlreichen vorherigen Aktionen des Agenten erhalten kein Feedback oder eine direkte Belohnung von $r_t(s_t, a_t) = 0$. Eine Schwierigkeit ist diese verspätete Belohnung am Ende einer Aktionssequenz auf die einzelnen Aktionen des Agenten aufzuteilen(engl. credit assignment problem).
[Ert16, S. 290]

2.1.2 Zeitliche Beschränkung

Endliches Horizont Modell mit h Schritten

Hat der Agent nur eine endliche Anzahl von Schritten h zur Verfügung, dann ist sein Aktionsradius beschränkt d.h. sein Verhalten verändert sich hinsichtlich der Anzahl der Schritte die dem Agenten zur Verfügung stehen.

$$E(\sum_{t=0}^{h} r_t) \tag{2.1}$$

Unendliches Horizont Modell

$$E(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2.2}$$

2.1.3 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.

2.1.4 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

$$\sum_{d=0}^{50} 30^d = \frac{1 - 30^{51}}{1 - 30} = 7{,}4 \times 10^{73} \tag{2.3}$$

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 Tic Tac Toe

Das klassische Tic Tac Toe ist ein Spiel, welches mit genau zwei Spielern gespielt wird. Jeder dieser Spieler setzt abwechselnd entweder ein Kreuz oder einen Kreis. Während eines gesamten Spiels darf ein Spieler nur Kreuze setzen und der andere Spieler nur Kreise. Wir können uns die Kreuze und Kreise als Spielfiguren vorstellen, die sobald sie auf das Spielfeld gesetzt wurden, nicht mehr verändert oder verschoben werden können. Das Spielfeld ist eine drei mal drei große Matrix, also können maximal neun Spielfiguren in diese Matrix gesetzt werden. Im ersten Spielzug stehen dem Spieler neun mögliche Positionen zur Verfügung. Die Anzahl der möglichen Positionen reduziert sich jede Runde um eins. Folglich ist die maximale Länge einer Spielzugsequenz, bei einem drei mal drei Spielfeld, neun. Maximal deshalb, weil das Spiel auch vor der neunten Runde bereits entschieden sein kann.

4.1.1 Spielregeln

Ziel des Spiels ist es vier Kreuze oder vier Kreise in einer bestimmten Position anzuordnen. Es wird davon ausgegangen, dass zwei Spieler Alice und Bob existieren

und gegeneinander Tic Tac Toe spielen. Alice setzt ausschließlich Kreuze und Bob ausschließlich Kreise. Wir gewähren Alice in unseren Beispielen immer 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 in gleicher Höhe (Nullsummenspiel).

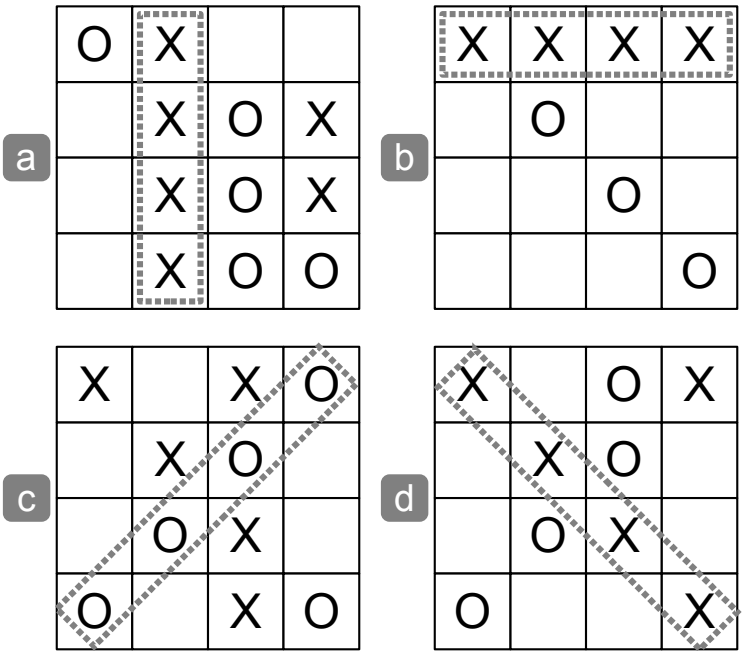

Abbildung 4.1 Veranschaulichung der vertikalen (a), horizontalen (b) und diagonalen (c, d) Siegesbedingung.

In Abbildung 4.1 a, gewinnt Alice knapp gegen Bob mit einer ununterbrochenen vertikalen Anordnung ihrer Spielfiguren. Bob hätte fast eine diagonale Reihe aus Kreisen verbunden, diese wurde jedoch von Alice mit einer Spielfigur geblockt. Zudem hätte Bob auch fast eine vertikale Spalte ohne Unterbrechungen vervollständigt. Eine horizontale Siegesanordnung entsteht, wenn vier Spielfiguren eines Spielers in einer horizontalen Zeile angeordnet sind. Alice gewinnt das zweite Spiel gegen Bob durch eine horizontale Siegesanordnung (siehe Abbildung 4.1 b). In jeder Spalte des Spielfeldes ist genau eine vertikale und in jeder Zeile des Spielfeldes ist genau eine horizontale Siegesanordnung möglich. 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 4.1 c, gewinnt Bob mit einer diagonalen Anordnung von vier Spielfiguren ohne Unterbrechung einer gegnerischen Spielfigur. Alice gewinnt ebenfalls eine Party, durch einen diagonale Siegesanordnung (Abbildung 4.1 d). Eine Party wird, in diesem Kontext, als ein abgeschlossenes Spiel, mit anschließendem Ergebnis, verstanden.

Bei einem vier mal vier Spielfeld existieren vier vertikale, vier horizontale und zwei diagonale Anordnungen der Spielfiguren, welche einen Sieg herbeiführen würden. Insgesamt zehn verschiedene Siegesanordnungen für beide Spieler. Was passiert jedoch, wenn keine der zehn möglichen Siegesanordnungen auftritt, dann gewinnt beziehungsweise 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.

4.1.2 Suchbaum

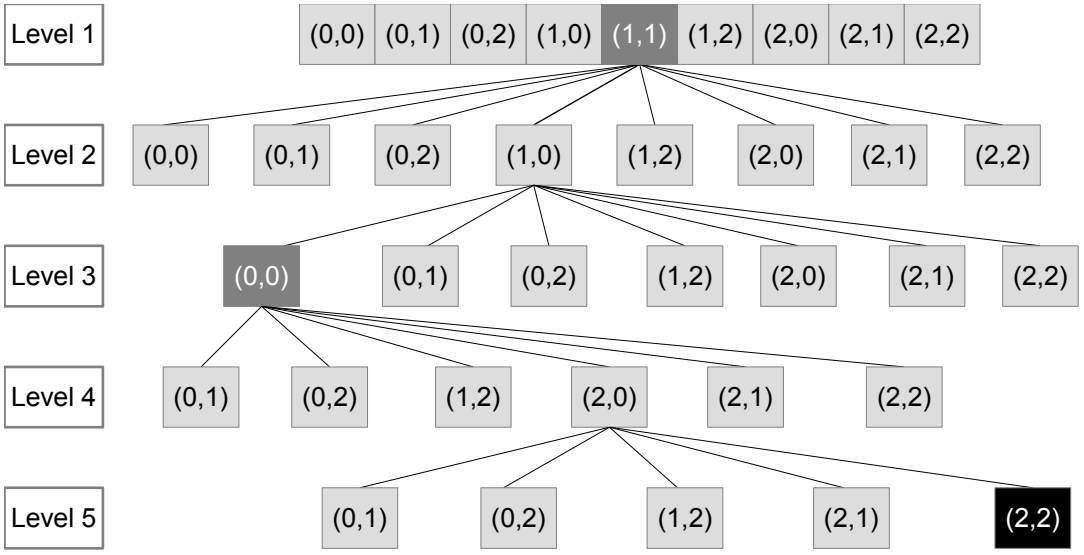

Abbildung 4.2 TicTacToe Spielbaum, eines drei mal drei Spielfeldes, der in Level fünf terminiert.

Ein Suchbaum für ein drei mal drei Spielfeld, hat maximal eine Tiefe (Level) von neun. Der Verzweigungsfaktor im ersten Level ist äquivalent zur maximalen Tiefe des Suchbaums. Für jedes weitere Level reduziert sich der Verzweigungsfaktor um eins. In Level eins ist der Verzweigungsfaktor gleich neun, in Level fünf ist er gleich fünf und in Level neun ist er gleich eins. Der Verzweigungsfaktor reduziert sich deshalb in jedem Spielzug, weil in jedem Spielzug eine Position mit einer Spielfigur be-

setzt wird und eine bereits besetzte Position darf nicht erneut besetzt werden. Der Sieg mit der kürzesten Spielzugsequenz ist nach fünf Spielzügen, für den beginnenden Spieler, zu erreichen. Bob der erst als zweites seinen Spielzug ausführen darf, kann erst ab einer Spielzugsequenz der Länge sechs gewinnen. Abbildung 4.2 zeigt einen schwarz markierten terminierenden Endzustand (2,2) des Spielbaums in Level fünf. Alice erreicht diesen Endzustand, weil sie ihren Spielstein in Zeile zwei und Spalte zwei setzt. Die terminierenden Endzustände des Spielbaumes werden allgemein als Blattknoten bezeichnet. Ein Blattknoten charakterisiert, dass es keine weiteren Verzweigungen oder Knoten unterhalb dieses Blattknotens geben darf. Es gibt zwei Möglichkeiten für das erreichen eines Blattknotens(Bedingung der Terminierung):

Ist jeder Blattknoten ein terminierender Endzustand?

1. Eine Siegesanordnung ist entstanden.
2. Alle möglichen Positionen des Spielfeldes sind besetzt und es wurde keine Siegesanordnung erreicht.

In Level fünf bis Level neun können terminierende Endzustände auftreten, das heißt Alice kann in Level fünf, Level sieben und in Level neun eine Siegesformation erreichen. Bob hingegen kann in Level sechs und Level acht eine Siegesanordnung erhalten. Theoretisch fehlt Bob, dem Spieler der als zweites beginnt, ein Zug gegenüber dem zu erst beginnenden Spieler. Tritt ein terminierender Endzustand von Level fünf bis Level acht auf, dann wurde dieser Endzustand immer durch das erreichen einer Siegesanordnung hervorgerufen, sprich durch die 1. Bedingung der Terminierung. Ein Blattknoten in Level neun kann durch die 1. und 2. Bedingung der Terminierung entstehen, das heißt entweder ist das Spielfeld voll besetzt und es wurde keine Siegesanordnung gefunden oder das Spielfeld ist voll besetzt und es wurde eine Siegesanforderung gefunden. Folglich besteht Level neun ausschließlich aus Blattknoten.

5 * 8 * (3 * 3) 40 terminierende Endzustände

Wie viele verschiedene Zustände kann ein drei mal drei Tic Tac Toe Spielfeld haben? Eine beliebige regelkonforme Anordnung von Spielsteinen auf dem Tic Tac Toe Spielfeld bezeichnen wir als einen Zustand s_1. Eine Menge die alle möglichen Zustände s enthält bezeichnen wir als die Menge S. Somit gilt $s_1, s_2 ... s_x \in S$. Der gesamte Spielbaum eines drei mal drei Tic Tac Toe Spielfeldes (Spielbaum ähnelt Abbildung 4.2) hat

$$\prod_{i=1}^{9} 9 - (i-1) = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 362880 \tag{4.1}$$

Blattkonten. In dieser Rechnung sind keine Positionsredundanzen und terminierende Endzustände berücksichtigt.

$$\prod_{i=0}^{5-1} 9 - i = 9 \times 8 \times 7 \times 6 \times 5 = 15120 \tag{4.2}$$

Um die Anzahl der möglichen Spielzüge zu erhöhen und um die mittlere unfaire Position in einem drei mal drei Spielfeld zu entfernen, wird das Spielfeld des klassischen Tic Tac Toe auf eine vier mal vier Matrix erweitert.

4.1.3 Lernen mit Strategieverbesserung

Die nachfolgend aufgestellte Strategie ist keine optimale Strategie, das heißt es ist nicht möglich mit dieser Strategie immer zu gewinnen oder mindestens ein Unentschieden zu erreichen. Das Ziel des Lernverfahrens ist es, diese nicht optimale Strategie in eine optimale Strategie oder zumindest eine annähernd optimale Strategie zu transformieren. Eine optimale Strategie würde gegen unaufmerksame oder unerfahrene Gegner verhältnismäßig oft Gewinnen und nicht gegen diese Verlieren, Unentschieden können trotzdem vorkommen. Ist der Gegner ein perfekter TicTacToe Algorithmus oder ein TicTacToe Großmeister, dann sollte die optimale Strategie überwiegend Unentschieden hervorbringen. Eine optimale Strategie sollte in 100 Spielen gegen einen Großmeister oder eine andere optimale Strategie (dieselbe optimale Strategie oder möglicherweise eine andere) 100 Unentschieden erringen. Siege sind theoretisch höherwertiger als Unentschieden, aber innerhalb der TicTacToe Spielwelt sind diese gegen einen Großmeister oder einen perfekten Algorithmus äußerst unwahrscheinlich.

Unser Lernalgorithmus oder im Kontext des verstärkenden Lernens unser Agent, erhält eine Belohnung von +1 wenn er eine Party (eine komplette Spielzugsequenz bis ein Spielergebnis feststeht) TicTacToe gewinnt. Verliert er eine Party, dann wird er bestraft mit dem numerischen Wert -1. Bei einem Unentschieden wird der Agent ebenfalls belohnt, aber die Belohnung ist nicht so hoch wie bei einem Sieg, denn wir wollen das Verhalten des Lernverfahrens so trainieren, dass es eher einen Sieg erlangen wird als ein Unentschieden, aber auf jeden Fall eine Niederlage vermeidet. Der Agent soll immer versuchen diesen nummerischen Wert zu maximieren, darum wird er eine Niederlage vermeiden. Der Agenten könnte auch so trainiert werden, dass er immer absichtlich verlieren würde, dafür müsste jeder, vom Agenten ausgeführte, Spielzug (egal welcher Spielzug) eine hohe negative nummerische Bestrafung hervorbringen. Das Ziel des Agenten wäre dann, schnellstmöglich ein Ende des Spiels zu provozieren und weil verlieren sicherer und kürzer ist als gewinnen oder ein Unentschieden, würde der Agent lernen absichtlich zu verlieren.

Interessant wäre das Spielergebnis, wenn zwei Agenten gegeneinander antreten würden, die immer schnellstmöglich verlieren wollen, dann entstünden theoretisch ausschließlich Unentschieden.

Kombination von überwachtem und verstärkenden Lernen
Dem Lernverfahren eine Strategie vorzugeben, welche für jeden Zustand festlegt welche Aktion ausgeführt werden soll, ähnelt stark dem überwachten Lernen. Bei einem überwachten Lernverfahren würde der Agent eine Liste von Zuständen und Aktionen als Eingabeparameter bekommen, in dieser Liste sind den Zuständen Aktion zugeordnet. Der Agent lernt die Liste von Zustand-Aktions-Paaren und dadurch lernt er, bei welcher Spielfiguren Anordnung (Zustand), welche Position mit seiner Spielfigur belegt werden sollte (Aktion). Die Zustände wären in diesem Trainingsset Eigenschaften (Features) und die Aktionen wären Zielvariablen (Target Values). In der Theorie wird überwachtes Lernen mit verstärkendem Lernen kombiniert, um ein schnell konvergierendes Lernverfahren zu erhalten. Die Konvergenz bezieht sich auf die Strategie, welche das Lernverfahren optimieren soll. Der Agent versucht diese nicht optimale Strategie schrittweise in eine optimale Strategie umzuwandeln und wenn ihm dies gelingt, dann hat die verbesserte nicht optimale Strategie eine Konvergenz mit der optimalen Strategie erreicht, dass bedeutet die Ausgangsstrategie wurde zu einer optimale Strategie weiterentwickelt.

Warum eigentlich kein reines überwachtes oder verstärkendes Verfahren? Eine optimale Strategie nur durch überwachtes Lernen zu erstellen, würde ein Trainingsset voraussetzen, indem jeder Zustand der Spielwelt erfasst ist und jeder dieser Zustände müsste genau eine optimale Aktion abbilden. Bei komplexeren Probleme kann das sehr hohe Kosten verursachen oder es ist überhaupt nicht möglich, denn für viele Zustände ist die optimale Aktion nicht bekannt. Das vier mal vier TicTacToe ist noch ein recht simpler Vertreter der Strategiespiele und soll dazu dienen, die Anwendung und Modellierung der Lernverfahren und der Strategien zu veranschaulichen. Ob ein Strategiespiel im Kontext der Lernverfahren simpler oder komplexer ist ,hängt von seinen Zustands- und Aktionsdimensionen ab. Die Größe das Spielfelds, die Anzahl verschiedener Aktionen pro Spielsituation und die durchschnittliche Anzahl der Spielzüge, entscheiden über die Komplexität des Strategiespiels. Wenn das überwachte Lernen einer optimalen Strategie aufgrund größerer Dimensionen nicht möglich ist, können wir dann nicht ein verstärkendes Lernverfahren implementieren?

Unterabschnitt 4.1.4 behandelt die Thematik des reinen verstärkenden Lernens. Daher werden die Vor- und Nachteile des reinen verstärkenden Lernens jetzt nicht weiter analysiert. Wir konzentrieren uns in diesem Abschnitt darauf, welche Strate-

gien entwickelt, wie diese Strategien praktisch angewendet und wie die angewendeten Strategien optimiert werden können. Bei der Entwicklung der Strategie muss besonders darauf geachtet werden, worauf der Agent aufmerksam gemacht werden soll. Unter Umständen ist es besonders Wahrscheinlich zu gewinnen, wenn eine besondere Spielstellung erreicht wurde oder einzelne Spielfelder erleichtern einen Sieg und sollten daher eher besetzt werden als andere möglicherweise unwichtigerer Spielfelder. Die Ausgangsstrategie ist also ein wichtiger Faktor für die Konvergenz-Geschwindigkeit.

Das Spielfeld

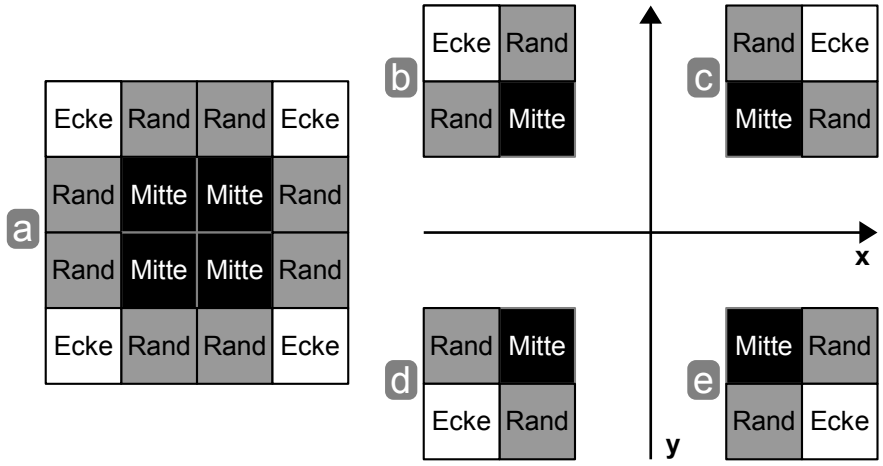

Abbildung 4.3 Symmetrie Eigenschaften des vier mal vier Tic Tac Toe Spielfelds.

Um eine Strategie zu entwickeln werden wir uns zuerst das Spielfeld ansehen und dieses analysieren. Das vier mal vier TicTacToe Spielfeld hat 16 Spielfelder, vier Eckfeldern, vier Mittelfeldern und acht Randfeldern (siehe Abbildung 4.3 a). Ziehen wir eine horizontale und eine vertikale Achse durch das Spielfeld, dann sind bestimmte Symmetrieeigenschaften zu erkennen, Abbildung 4.3 zeigt b und c, sowie d und e sind symmetrisch zur y-Achse, b und d, sowie c und e sind symmetrisch zur x-Achse und b und e, sowie d und c sind symmetrisch, wenn man sie an der x-Achse und der y-Achse spiegelt. Diese Symmetrieeigenschaften sind wichtig für die Reduktion der Strategien, das heißt wenn eine Strategie auf b angewendet werden kann, dann auch auf c, d und e.

Kontrolliere die Mitte

Um über bestimmte Spielfelder reden zu können, müssen wir eine konkrete Identifikation der einzelnen Spielfelder vornehmen. In Abbildung 4.4 wird daher jedem der Spielfelder ein Index zugewiesen.

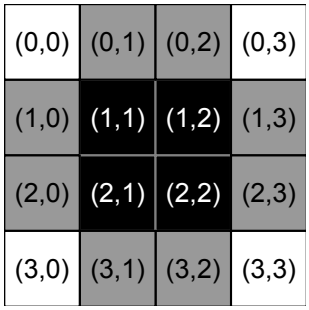

Abbildung 4.4 Die Indizes der einzelnen Spielfelder.

Die Eröffnungsstrategie konzentriert sich auf die Kontrolle der Mittelfelder. Hat der Agent beziehungsweise das Lernverfahren das Recht auf den ersten Zug befindet er sich immer in Zustand s_0 (siehe Abbildung 4.5 1). In diesem Zustand sind alle Spielfelder leer und der Agent kann durch Zufall eines der vier mittleren Felder wählen. Der Agent trifft diese Entscheidung zufällig, weil zu diesem Zeitpunkt und in diesem Zustand alle vier Mittelfelder die gleiche positive numerische Belohnung erbringen. Die Rand- und Eckfelder haben keine numerische Belohnung, aber auch keine numerische Bestrafung für den Agenten, so wird sichergestellt, dass das Lernverfahren sich für ein mittleres Spielfeld entscheidet. Nachdem der gegnerische Spieler seine Spielfigur gesetzt hat, soll der Agent die zweite Spielfigur ebenfalls auf ein mittleres Feld setzen, jedoch eher auf ein mittleres Spielfeld, welches sich in einer Reihe ohne eine gegnerische Spielfigur befindet.

Verteidigung ist der beste Angriff

Von jetzt an betrachten wir die beiden Kontrahenten Alice (Spielfiguren X) und Bob (Spielfiguren O) als zwei Instanzen des selben Lernverfahrens, dass heißt der Agent spielt gegen einen anderen Agenten mit exakt dem selben Verhalten.

Strategiezusammenfassung

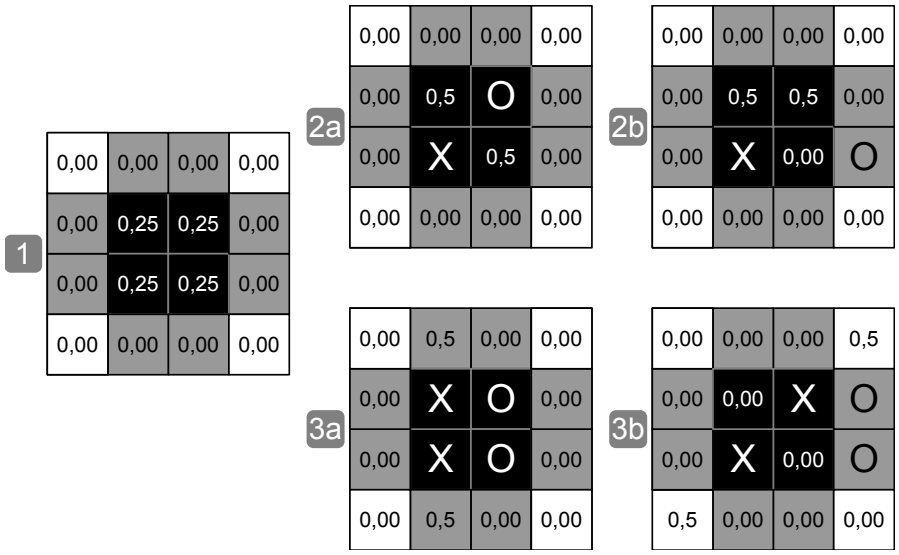

Abbildung 4.5 Strategie um die Mitte zu kontrollieren.

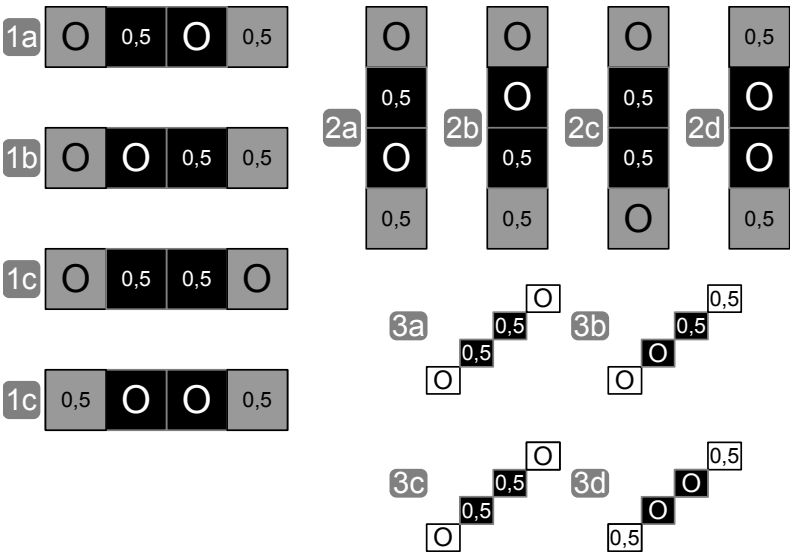

Abbildung 4.6 TicTacToe Angriffsstrategien (aus der Sicht von Bob) und Verteidigungsstrategien (aus der Sicht von Alice).

4.1.4 Lernen ohne Strategie

4.1.5 Benutzerschnittstellen

Alice kann ihre Kreuze auf das Spielfeld setzen, indem sie ein, vorher vom Spiel definiertes, Zahlentupel über die Tastatur eingibt. Welche Zahlentupel ein Kreuz an welche Stelle setzt ist in Abbildung 4.7 definiert. Sollte Alice keines der erlaubten Zahlentupel eingeben, dann wird sie darauf hingewiesen, welche Steuerungsmöglichkeiten ihr, zum setzen der Spielfiguren, zur Verfügung stehen.

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

Abbildung 4.7 Tic Tac Toe Spielfiguren setzen

4.2 Reversi

Spielprinzipien

Spielregeln

Benutzerschnittstellen

4.3 Lernverfahren

4.3.1 Analyse und Auswahl der lernfähigen Algorithmen

4.3.2 Anwendung der Algorithmen auf Computerspiele

4.3.3 Konzeptuelles Training der Algorithmen

4.3.4 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!