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

22. Februar 2017

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

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

Todo list

Abkürzungsverzeichnis

eng. Englische Sprache

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

In diesem Kapitel werden verschiedene Konzepte der Spieltheorie Abschnitt 2.1 und des maschinellen Lernens Abschnitt 2.2 genauer erklärt und veranschaulicht. Diese Konzepte sind der Grundbaustein für die spätere Anwendung der lernfähigen Verfahren. Konzepte der Spieltheorie sind z.B. Suchstrategien und deren Optimierungen. Im zweite Abschnitt "Verstärkendes Lernen"behandeln wir das Konzept eines Agenten der ohne Strategie in einer Welt ausgesetzt wird und in dieser Aktionen ausführt. Wir betrachten insbesondere das Verhalten des Agenten hinsichtlich einer Belohnung für eine durchgeführte Aktion.

2.1 Spiele mit Gegenspieler

Schach, Vier Gewinnt, Dame, Tic Tac Toe und Reversi sind deterministische Spiele für zwei Personen, die gegeneinander antreten, um nach den Regeln des Spiels, den Gegenspieler zu besiegen. Diese Spiele sind deterministisch, weil zu jedem Zeitpunkt des Spiels das Spielfeld und alle Spielzüge einsehbar sind, dass Spiel nicht vom Zufall abhängt und der gleiche Spielzug führt bei gleichem Ausgangszustand immer zum selben Spielergebnis. Wir nehmen an es existieren zwei Spieler Alice und Bob, die gegeneinander antreten. Ein nichtdeterministisches Spiel mit Gegenspieler ist z.B. Backgammon. Wie muss ein Agent programmiert werden, um ein deterministisches Spiel gegen einen menschlichen Gegenspieler zu gewinnen? In den nachfolgenden Unterabschnitten werden Konzepte der Spieltheorie erläutert, die versuchen das Problem zu lösen.

Nullsummenspiele

Ein Nullsummenspiel ist ein Spiel bei dem der Verlust eines Spielers einen gleich hohen Gewinn für den Gegenspieler bedeutet. Gewinnt Alice eine Partie eines Nullsummenspiels, dann verliert Bob automatisch. Alice erhält einen Pluspunkt und Bob einen Minuspunkt. Die Summe der beiden Ergebnisse ist Null. Gewinnt beziehungsweise verliert keiner der beiden Spieler, dann erhalten beide Spieler als

Spielergebnis eine Null. Die Summe der beiden Ergebnisse ist wiederum Null, so lässt sich der Name Nullsummenspiele"herleiten. Die beiden Spiele Tic Tac Toe und Reversi sind Nullsummenspiele.

2.1.1 Minimax

Ein Spieler wird als MAX bezeichnet und der Gegenspieler als MIN. Spieler MAX versucht einen maximalen Gewinn für sich zu erlangen und Spieler MIN versucht den erreichbaren Gewinn von MAX zu minimieren. Es wäre auch möglich anzunehmen, dass Spieler MIN einfach irgendeinen zufälligen oder dummen Spielzug auswählt und MAX einen einfachen Sieg erlangt. Diese Annahme entspricht jedoch wenig einem realen Spiel, bei dem beide Spieler versuchen das Spiel zu gewinnen.

In Abbildung 2.3 ist der Ablauf einer Minimax-Suche veranschaulicht. Nehmen wir an Spieler MAX sei Alice und Spieler Bob sei MIN. Der Minimax-Suchbaum berücksichtigt jeden Zustand indem sich die Spielwelt befinden kann. Im ersten Spielzug könnte Alice Ihr Kreuz in die obere linke Ecke setzen, daraus ergeben sich neue Zustandsmöglichkeiten. Bob könnte seinen Kreis ein Feld weiter rechts und in die selbe Reihe wie Alice setzen. Der Nutzen der einzelnen Züge ist zur Zeit der Ausführung noch nicht bekannt, erste wenn das Tic Tac Toe Spiel einen Endzustand erreicht, werden den Spielern ihre Spielergebnisse mitgeteilt. Der Minmax-Suchbaum ist somit rekursiv zu betrachten. Von seinen Blattknoten ausgehend entscheidet sich Bob für den geringsten Nutzwert und Alice für den höchsten. Die Entscheidungen stehen in direkter Abhängigkeit zur vorherigen Entscheidung des Gegenspielers.

Das größte Problem der Minimax-Suche ist die exponentielle Vergrößerung der Anzahl der Blattknoten des Suchbaums, schon bei sehr einfachen Spielen z.B. einem drei mal drei Tic Tac Toe mit Neun Spielfeldern ist der Suchbaum bereits sehr groß ungefähr 362880 Blattknoten mit einem effektiven Verzweigungsfaktor von Neun der sich nach jedem Halbzug um Eins verringert. Erweitern wir die Dimension des Tic Tac Toe Spiels, so erhalten wir ein vier mal vier Tic Tac Toe Spiel mit 16 Spielfeldern, einem Verzweigungsfaktor von 16 und ungefähr 20922789888000 Blattknoten. Der effektive Verzweigungsfaktor beim Schach liegt etwa bei 30 bis 35. Bei einem typischen Spiel mit 50 Zügen pro Spieler hat der Suchbaum dann mehr als $30^{100} \approx 10^{148}$ Blattknoten[Ert16, S. 114]. Der Minimax-Algorithmus ist in seiner Reinform praktisch nicht anwendbar, daher werden wir Erweiterungen der Minmax-Suche und andere Konzepte kennen lernen.

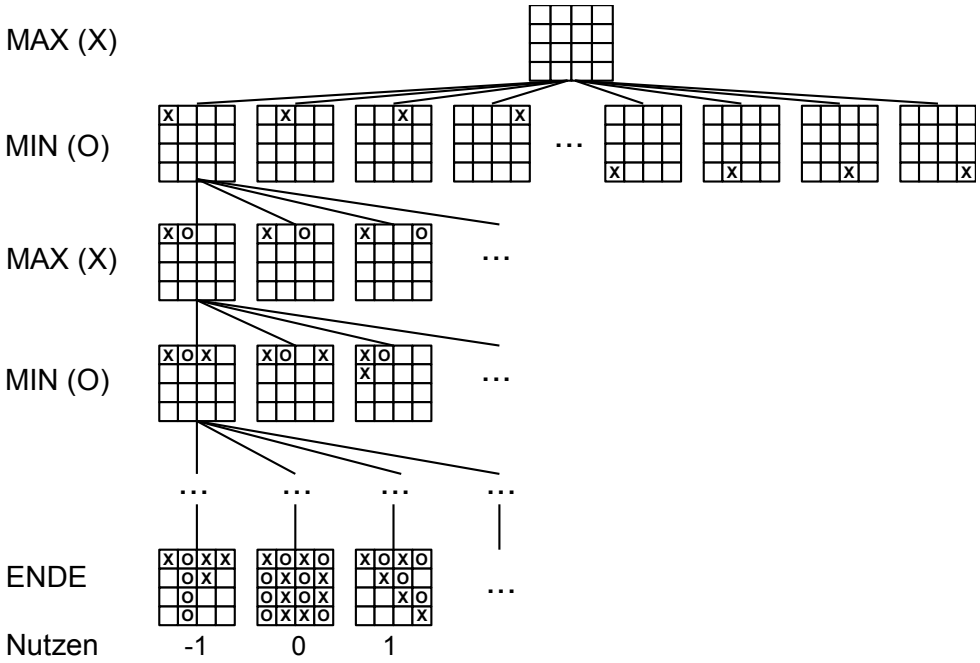

Abbildung 2.1 Ein (partieller) Suchbaum vgl. Abbildung 5.1 [RN12, S. 208]

2.1.2 Alpha-Beta-Pruning

Eine Möglichkeit die Rechenzeit der Minimax-Suche zu verbessern ist das Kürzen oder Beschneiden des Suchbaums(eng. Pruning). Beim Alpha-Beta-Kürzen wird der Teil des Suchbaums beschnitten, der keinen Effekt auf das Ergebnis der Minimax Suche hat. Der Minimax Algorithmus wird um zwei Parameter Alpha und Beta ergänzt. Die Bewertung erfolgt an jedem Blattknoten des Suchbaums. Alpha enthält den aktuell größten Wert, für jeden Maximum Knoten, der bisher bei der Traversierung des Suchbaums gefunden wurde. In Beta wird für jeden Minimum Knoten der bisher kleinste gefundene Wert gespeichert. Ist Beta an einem Minimum Knoten kleiner oder gleich Alpha ($Beta \leq Alpha$), so kann die Suche unterhalb von diesem Minimum Knoten abgebrochen werden. Ist Alpha an einem Maximum Knoten größer oder gleich Beta ($Alpha \geq Beta$), so kann die Suche unterhalb von diesem Maximum Knoten abgebrochen werden [Ert16, S. 116].

Verdeutlichen wir das Alpha-Beta-Pruning an Hand eines Beispiels aus dem Standardwerk der künstlichen Intelligenz von S. Russell und P. Norvig Abbildung 5.5 [RN12, S. 213]. Ein Dreieck mit der Spitze nach oben ist ein Maximumknoten und ein Dreieck mit der Spitze nach unten ist ein Minimumknoten. Leere Dreiecke ohne einen bezeichnenden Buchstaben und gestrichelter Umrandung sind noch nicht explorierte Knoten. Durchgängige Linien verweisen auf bereits besuchte Pfade und

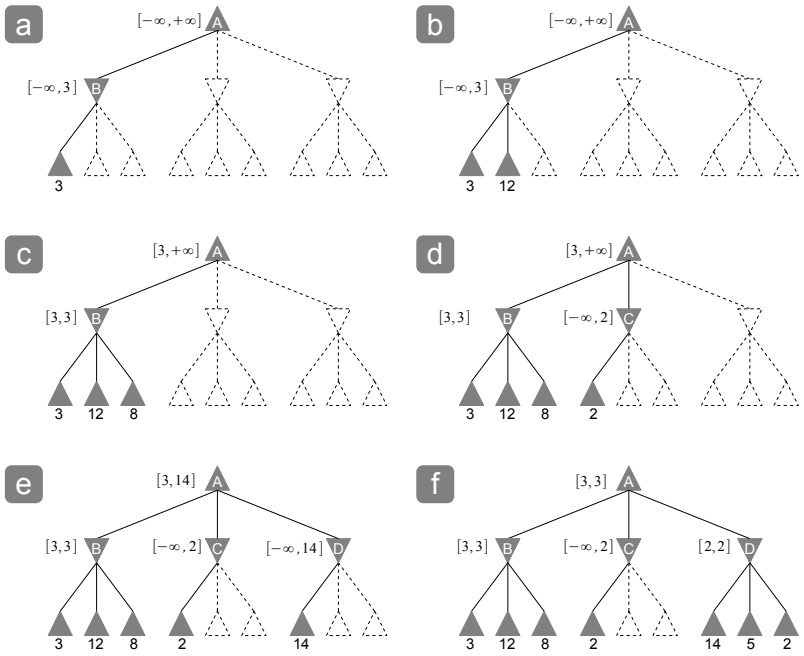

Abbildung 2.2 Ein Alpha Beta Suchbaum [RN12, S. 213].

gestrichelte Linien verweisen auf noch nicht besuchte Pfade. Die Zahlen unterhalb der Blattknoten sind die Nutzwerte die der maximierende Spieler erhält, wenn er den Pfad bis zu diesem Blattknoten durchschreitet.

(a) Minimum Knoten B findet einen Nutzwert 3, da dieser Wert der bisher kleinste gefundene Wert ist wird er in Beta gespeichert.

(b) Der Minimum Knoten B exploriert einen zweiten möglichen Nutzwert 12. Dieser Wert ist höher als der vorher gefundene und in Beta gespeicherte Wert 3, daher wird der minimierende Spieler versuchen diesen Nutzwert für den maximierenden Spieler zu vermeiden. Der neue Wert wird vom Minimum Knoten B ignoriert und Beta bleibt unverändert.

(c) Minimum Knoten B findet den Wert 8, dieser ist genau wie 12 größer als 3 und daher wird Spieler MIN vermeiden, dass Spieler MAX zu diesem Spielergebnis gelangt. Minimum Knoten B hat alle seine nachfolgenden Knoten exploriert. Maximum Knoten A wird vom Minimum Knoten B maximal den Nutzwert 3 erhalten, somit ergibt sich für den Maximum Knoten A, dass dieser mindestens den Nutzwert 3 erreichen kann.

(d) Ein weiterer Minimum Knoten ist C. Der erste Blattknoten von C liefert einen

Nutzwert von 2, weil dieser Wert der erste gefundene Wert unterhalb des Minimum Knotens C ist, wird er in Beta gespeichert. C wird Maximum Knoten A maximal einen Nutzwert 2 liefern. A wiederum kann durch Minimum Knoten B bereits einen minimalen Nutzwert von 3 erhalten und hat diesen in Alpha gespeichert. Es gilt $Beta \leq Alpha$ und es ist nicht notwendig die Knoten unterhalb von C weiter zu explorieren. Selbst wenn ein größerer Nutzwert gefunden werden würde, entscheidet sich der minimierende Spieler trotzdem für den kleineren Wert und würde ein kleinerer Nutzwert als 2 gefunden werden, dann entscheidet sich der maximierende Spieler für den Nutzwert 3, den Minimum Knoten B liefert. Folglich kann der Suchbaum an dieser Stelle abgeschnitten werden, weil weitere gefundene Nutzwerte keinen Einfluss mehr auf das Ergebnis haben.

(e) Der letzte von A zu erreichende Minimum Knoten wird exploriert. Der erste Blattknoten unterhalb des Minimum Knoten D liefert den Nutzwert 14. Dieser Wert wäre für Maximum Knoten A eine starke Verbesserung, weil dieser bisher nur maximal einen Nutzwert von 3 erreichen konnte. Der minimierende Spieler hat noch zwei weitere Möglichkeiten(Knoten) zu explorieren und daher wird er versuchen einen geringeren Nutzwert als 14 zu finden.

(f) Minimum Knoten D findet in den beiden letzten Blattknoten die Nutzwerte 5 und 2. Der minimierende Spieler wählt die Möglichkeit mit dem geringsten Nutzwert 2. Dieser Nutzwert wird zum neuen Beta Wert. Der Suchbaum wird unterhalb vom Minimum Knoten D jedoch nicht abgeschnitten, weil der Nutzwert 2 erst im zuletzt explorierten Knoten gefunden wurde. Theoretisch könnten zwei Pfade unterhalb des Minimum Knoten D abgeschnitten werden, wenn der Blattknoten mit dem Nutzwert 2 zuerst exploriert worden wäre.

2.1.3 Iterative Deepening

Die iterative Tiefensuche (eng. Iterative Deepening) ist eine Kombination der Breitensuche und der Tiefensuche. Diese Suchverfahren sind uninformierte (blinde) Suchverfahren. Die Strategien der uninformierten Suchverfahren haben keine zusätzlichen Informationen über Zustände, außer den in der Problemdefinition vorgegebenen. Alles was sie tun können, ist, Nachfolger zu erzeugen und einen Zielzustand von einem Nichtzielzustand zu unterscheiden. Die Reihenfolge der Suche ist entscheidend für die Unterscheidung der einzelnen uninformierten Suchverfahren [RN12, S. 116].

Die Breitensuche expandiert (erweitert oder vergrößert) zu erst alle Nachfolger (Knoten eines Suchbaums) die in derselben Tiefe liegen, beginnend mit dem Wur-

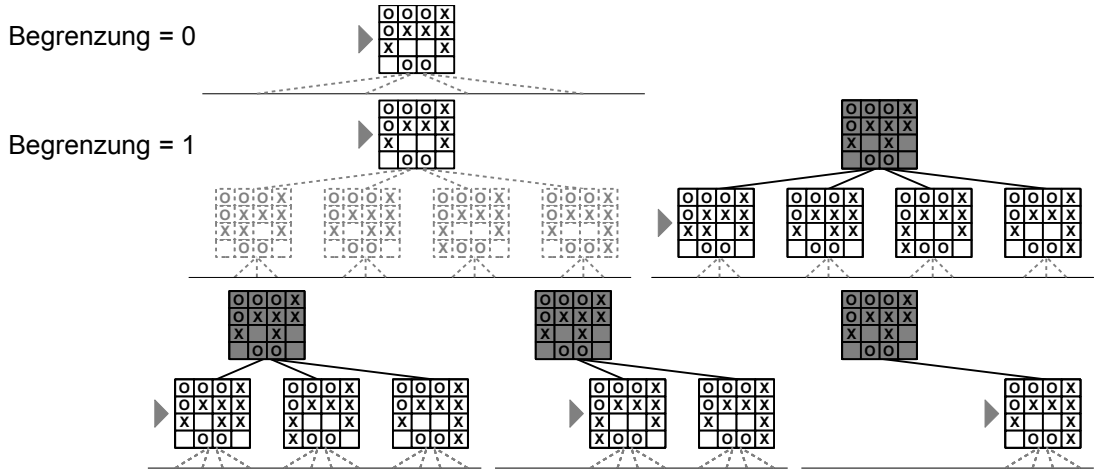

Abbildung 2.3 TODO

zelknoten. Sind alle Nachfolger einer Tiefe expandiert, dann werden deren Nachfolger nacheinander expandiert. Diesen Schritt wiederholt die Breitensuche bis ein gesuchtes Ergebnis gefunden wird.

Die Tiefensuche expandiert zuerst die tiefsten Knoten des Suchbaums(Depth-first). Erreicht die Tiefensuche einen Endknoten der nicht dem gesuchten Ergebnis entspricht, dann werden die alternativen Knoten des letzten expandierten Knotens, der sich eine Tiefenebene höher befindet, expandiert.

Kombinieren wir diese beiden uninformierten Suchverfahren miteinander und mit einer Grenze für die Suchtiefe, erhalten wir die iterative Tiefensuche. Diese expandiert zuerst die Nachfolger des Wurzelknotens der Suchtiefe 1. Sind alle Knoten auf dieser Ebene expandiert, dann wird die Schranke für die aktuelle Suchtiefe um 1 erhöht (Iteration) und die Knoten der Suchtiefe 2 werden expandiert. Diese Schritte wiederholt die Tiefensuche bis ein Ziel gefunden wird.

Anwendung findet die iterative Tiefensuche bei der Zugsortierung für die Verbesserung des Alpha-Beta Suchbaumkürzens.

2.1.4 Transition Table

Eine Transitionstabelle (Transition Table) ist eine Tabelle (z.B. in einer SQLite Datenbank) in der Spielsituationen mit verschiedenen Attributen gespeichert werden (vgl. [RN12, S. 215 f.]). Transitionen sind der Grund dafür, dass der gleiche Spiel-

zustand durch unterschiedliche Spielzugsequenzen auftritt (siehe Abbildung 2.4). Folgende Attribute werden in der Transitionstabelle gespeichert: der Spielzustand, die Alpha-Beta Werte, der bestmögliche Halbzug, der Nutzen und welcher Spieler gerade setzen muss. Alle Attribute beziehen sich auf den gespeicherten Spielzustand z.B. der bestmögliche berechnete Halbzug der für den gespeicherten Spielzustand möglich ist.

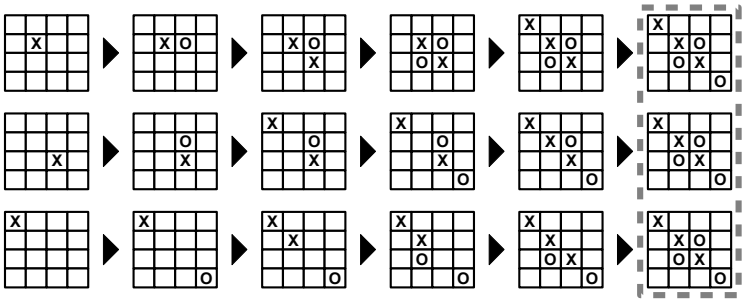

Abbildung 2.4 Drei verschiedene Spielzugsequenzen die alle zum selben Spielzustand führen.

Transitionen innerhalb des Suchbaums verursachen Redundanzen. Für jede dieser Redundanzen wird eine erneute Suche durchgeführt, falls diese nicht durch Alpha-Beta-Kürzung abgeschnitten werden. Sollten diese Transitionen vermieden werden können, dann würde sich die Rechenzeit der Suchverfahren weiter verringern, weil weniger Spielzustände durchsucht bzw. expandiert werden müssen. Wie können wir Spielzustände in einer Transitionstabelle abspeichern?

Zobrist Hash

Wenn ein Computerprogramm einen Gegenstand in einer großen Tabelle speichert, muss die Tabelle zwangsläufig durchsucht werden, um den Gegenstand wiederzuverwenden bzw. zu referenzieren. Dies gilt solange, bis eine Tabellendresse aus dem Gegenstand selbst, in systematischer weise, berechnet werden kann. Eine Funktion die Gegenstände in Adressen umwandelt ist ein Hash-Algorithmus, und die daraus resultierende Tabelle ist eine Hashtabelle [Zob70, S. 3].

In Abbildung 2.5 wird das Zobrist Hash Verfahrens auf den redundanten Spielzustand aus Abbildung 2.4 angewendet. (2.5 a) Wir weisen jedem Spielfeld zwei zufällige ganzzahlige Werte zu im Bereich von 0 bis maximal 1×10^9. Einen zufälligen Wert für den Kreuzspielstein an dieser Position und einen für den Kreisspielstein. Das 4x4 Tic Tac Toe Spielbrett sollte insgesamt 32 verschiedene Werte erhalten. (2.5

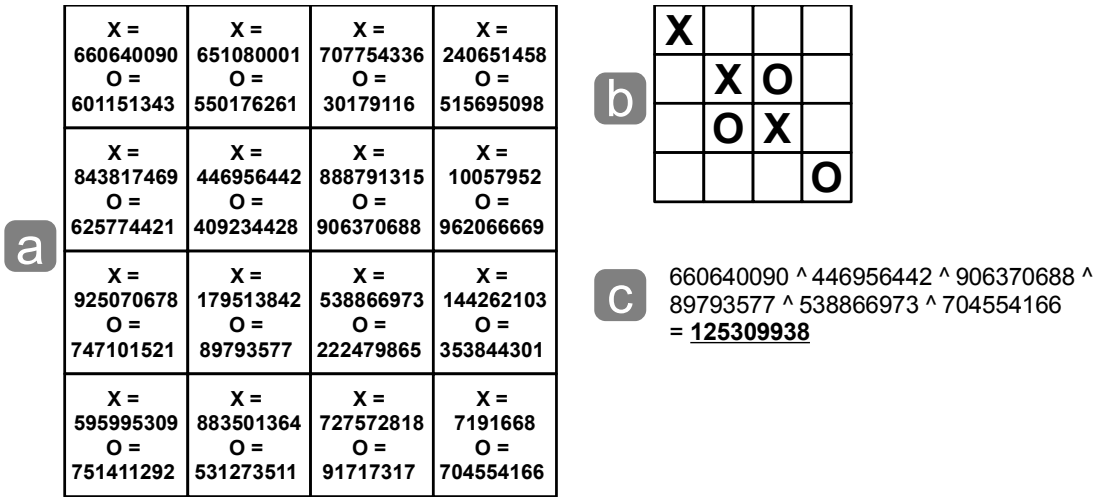

Abbildung 2.5 Zobrist Hashing von Spielzuständen.

b) Dieser Spielzustand soll in einen Zobrist-Hash umgewandelt werden.

(2.5 c) Der Zobrist-Hash berechnet sich wie folgt, ist die aktuelle Position mit einem Kreuzspielstein oder einem Kreisspielstein besetzt, dann wähle den entsprechenden Wert aus (2.5 a). Dies wiederhole für jedes besetzte Spielfeld. Auf den bereits ermittelten Wert und den hinzukommenden Wert wird ein exklusives bitweises Oder (XOR) angewendet. In Python ist das Zeichen für die Funktion des exklusiven bitweisen Oder ein Zirkumflex (siehe Operatoren in der Berechnung 2.5 c). Das Ergebnis ist eine Adresse die exakt den Spielzustand (2.5 b) referenziert.

Spielzustände die gerade expandiert werden, können in die Transitionstabelle eingetragen werden, sollten diese nicht bereits in der Tabelle vorhanden sein. Ist dieser Spielzustand bereits in der Tabelle vorhanden, dann können die vorgeschlagenen besten Halbzüge aus der Transitionstabelle ausgelesen und angewendet werden.

2.1.5 Heuristik

Eine Heuristik oder Bewertungsfunktion berechnet einen Nutzwert für einen gegebenen Spielzustand. Dieser Nutzwert gibt an, wie "wertvoll"diese Spielsituation hinsichtlich eines Sieges ist, sprich sie gibt an ob der Spieler in diesem Spielzustand eher gewinnen oder verlieren könnte. Trotz aller Optimierungen des Minimax Verfahrens (Alpha-Beta, Zugsortierung, Vermeidung von Redundanzen) wäre die Rechenzeit, für den zu durchsuchenden Baum, immer noch enorm hoch. Reale

Zeitbeschränkungen z.B. bei Schach Spielen erlauben ein überaus langes Berechnen ohnehin nicht. Die Lösung ist das verwenden einer Heuristik. Der Kompromiss bei einer Heuristik ist: das Ergebnis wird geschätzt und ist nicht mehr sicher, aber die Suche kann nach einem Zeitkriterium abgebrochen werden und die beste bisher gefundene Lösung wird zurückgegeben. Sollte die Bewertungsfunktion für einen Spielzustand z.B. einen sehr hohen Wert berechnen, dann besagt dieser, der Spieler der diesen Spielzustand erreicht wird wahrscheinlich gewinnen.

Die Qualität einer Heuristik ist ausschlaggebend für die Spielerischen Fähigkeiten eines Programms. Ein Programm welches, durch eine schlechte Stellungsbewertung (Heuristik) einen fatalen Spielzug des Gegners übersieht oder ignoriert, würde gegen ein Programm verlieren, welches diese Stellungen (Spielzustände) erkennt und ausnutzt bzw. entsprechend verhindert. Eine Bewertungsfunktion B(s) für ein Schachspiel enthält folgende Elemente, wobei s der Parameter für den Spielzustand ist[Ert16, S. 119]:

B(s) = a_1 x Material + a_2 x Bauernstruktur + a_3 x Königssicherheit
+ a_4 x Springer im Zentrum + a_5 x Läufer Diagonalabdeckung + ...,

das mit Abstand wichtigste Feature (Merkmal) Materialnach der Formel

Material = Material(eigenes Team) - Material(Gegner)

Material(Team) = Anzahl Bauern(Team) x 100 + Anzahl Springer(Team) x 300
+ Anzahl Läufer(Team) x 300 + Anzahl Türme(Team) x 500
+ Anzahl Damen(Team) x 900

In Kapitel 4 Modellierung und Entwurf werden wir ähnlich der gerade vorgestellten Heuristik, eigene Bewertungsfunktionen für Tic Tac Toe und Reversi entwerfen und in Kapitel 5 Implementierung werden die modellierten Bewertungsfunktionen praktisch angewendet.

2.2 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.

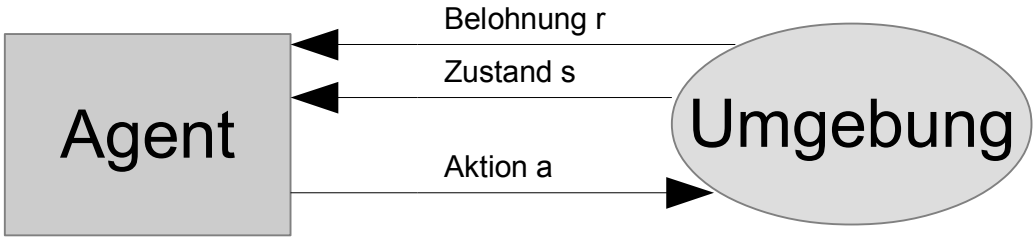

Abbildung 2.6 Der Agent und seine Wechselwirkung mit der Umgebung

Verstärkendes oder auch bestärkendes Lernen beschäftigt sich mit dem Problem, dass ein entscheidungstreffender Agent innerhalb einer unbekannten Umgebung Aktionen (Entscheidungen) ausführen soll (siehe Abbildung 2.6 [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?

2.2.1 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 2.6). 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]

2.2.2 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

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

2.2.3 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 2.7 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.

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

	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 2.7 Vogelspezies Klassifikation basierend auf vier Eigenschaften

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.

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

GNU 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

$$\sum_{d=0}^{50} 30^d = \frac{1 - 30^{51}}{1 - 30} = 7,4 \times 10^{73} \tag{3.1}$$

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.

Samuels Dame Spiel

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 (ID) soll die nachfolgend aufgestellten Anforderungen eindeutig identifizieren.

3.2.2 Nicht-funktionale Anforderungen

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.

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

Tic Tac Toe ist ein Spiel, welches mit genau zwei Spielern gespielt wird. Während eines gesamten Spiels darf ein Spieler nur Kreuze setzen und der andere Spieler nur Kreise. Ein Spieler der während einer Partie nur Kreuze setzen darf wird als Kreuzspieler und sein Gegner als Kreisspieler bezeichnet. Wir können uns die Kreuze und Kreise als Spielsteine vorstellen, die sobald sie auf das Spielfeld gesetzt wurden, nicht mehr verändert oder verschoben werden können. Das Spielbrett ist eine 4 x 4 große Matrix, also können maximal 16 Spielsteine in diese Matrix gesetzt werden. Der Kreuzspieler muss immer als erster beginnen. Im ersten Spielzug stehen dem Kreuzspieler 16 mögliche Positionen zur Verfügung. Die Anzahl der möglichen Positionen reduziert sich jede Runde um 1, weil jede Runde genau ein gesetzter Spielstein ein Spielfeld besetzt. Folglich ist die maximale Länge einer Spielzugsequenz bei einem 4 x 4 Spielfeld gleich 16. Es ist auch möglich, dass das Spiel bereits vor der 16. Runde beendet wird.

Spielzüge jeder Spieler setzt abwechselnd entweder ein Kreuz oder einen Kreis in ein Spielfeld des Spielbretts. Ein Spielstein kann in jedes der 16 Spielfelder ge-

setzt werden, außer dieses ist bereits mit einem anderen Spielstein besetzt, dann muss der Spieler ein anderes Spielfeld auswählen. Die Spieler führen solange Ihre Spielzüge aus, bis eine Siegesformation eines Spielsteintyps erreicht ist oder alle Spielfelder besetzt sind.

Ziel des Spiels ist es vier Kreuze oder vier Kreise in einer bestimmten Position anzuordnen. Es existieren mehrere unterschiedliche Anordnungen von Spielsteinen, die das Spiel beenden und einen Sieg herbeiführen. Bei einem 4 x 4 Spielfeld existieren vier vertikale, vier horizontale und zwei diagonale Anordnungen der Spielfiguren, welche einen Sieg herbeiführen würden. Insgesamt zehn verschiedene Siegesanordnungen für beide Spieler. Sind alle Spielfelder besetzt und für keinen der Spieler ist eine Siegesformation aufgetreten, dann gewinnt beziehungsweise verliert keiner der beiden Spieler und es entsteht ein Unentschieden. Gewinnt ein Spieler mit einer Siegesanordnung seiner Spielsteine, dann verliert der andere Spieler dadurch automatisch in gleicher Höhe (Nullsummenspiel).

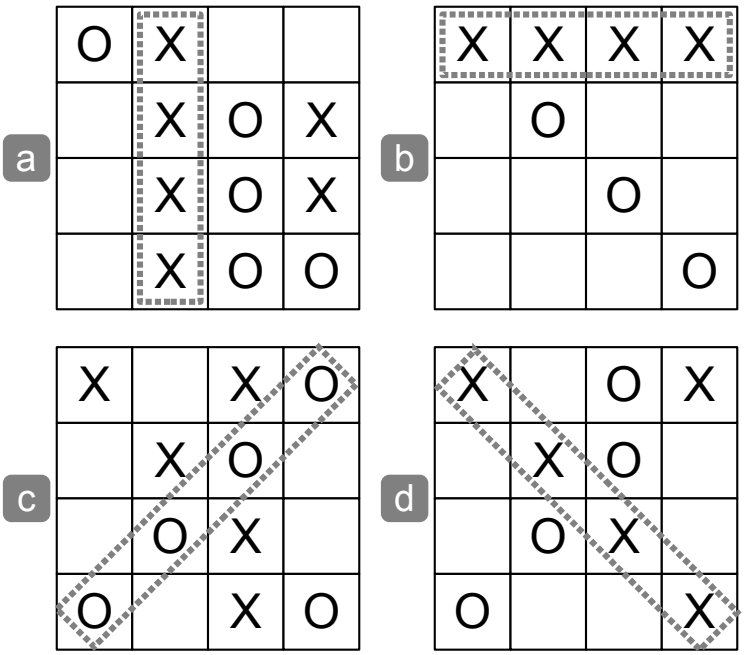

Abbildung 4.1 Veranschaulichung der vertikalen (a), horizontalen (b) und diagonalen (c, d) Siegesbedingung.

Vier mögliche Siegesformationen sind in Abbildung 4.1 dargestellt. (a) Der Kreuz-

spieler gewinnt knapp gegen seinen Kontrahenten mit einer ununterbrochenen vertikalen Anordnung seiner Spielsteine. Der Kreisspieler hätte fast eine diagonale Reihe aus Kreisen verbunden, diese wurde jedoch vom Kreuzspieler mit einem Spielstein unterbrochen. Zudem hätte der Kreisspieler auch fast eine vertikale Reihe ohne Unterbrechungen vervollständigt, aber der Sieg des Kreuzspielers hat die Partie vorher beendet. (b) Der Kreuzspieler erreicht eine horizontale Siegesanordnung, vier Kreuzsteine in einer horizontalen Zeile angeordnet. (c) Der Kreisspieler besiegt den Kreuzspieler mit einer diagonalen Siegesanordnung. (d) Der Kreuzspieler gewinnt ebenfalls durch eine diagonale Siegesformation seiner Spielsteine.

4.1.1 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 4.1.2 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

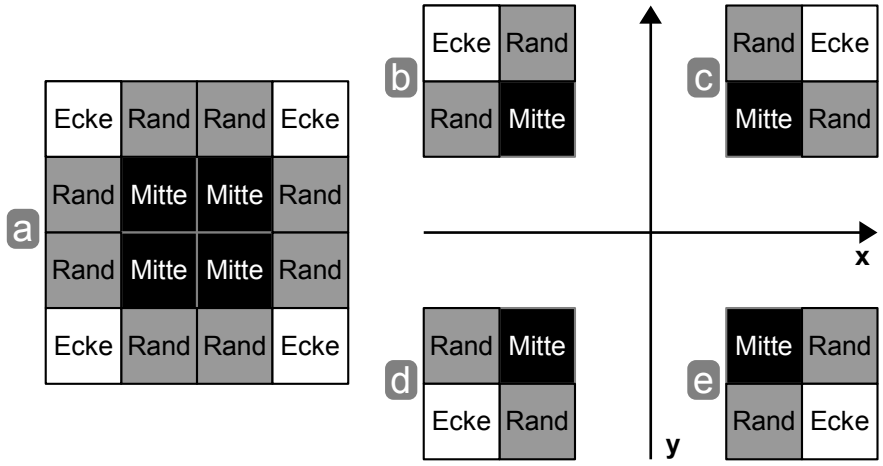

Abbildung 4.2 Symmetrie Eigenschaften des vier mal vier Tic Tac Toe Spielfelds.

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 4.2 a). Ziehen wir eine horizontale und eine vertikale Achse durch das Spielfeld, dann sind bestimmte Symmetrieeigenschaften zu erkennen, Abbildung 4.2 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 4.3 wird daher jedem der Spielfelder ein Index zugewiesen.

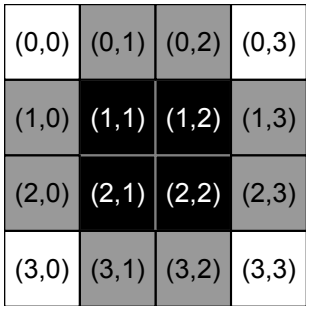

Abbildung 4.3 Die Indizes der einzelnen Spielfelder.

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. 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 (siehe Abbildung 4.4 1). 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.

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. In Abbildung 4.4 (2a) setzt Alice ihre Spielfigur auf das Feld (2,1) und Bob auf (1,2). Die Strategie der Mittelfeld Kontrolle offenbart unserer Agentin Alice zwei Aktionsmöglichkeiten mit positiver Belohnung. Setzt Agentin Alice ihre Spielfigur auf die Mittelfelder (1,1) oder (2,2), dann erhält sie einen numerische Belohnung von +0,5. Alice entscheidet durch Zufall welches der beiden Felder mit der gleichgroßen größtmöglichen Belohnung sie auswählt. Alice entscheidet sich durch Zufall für das Spielfeld (1,1) und Bob, der ebenfalls der Strategie der Mittelfeld Kontrolle folgt, entscheidet sich für das letzte nicht besetzte Mittelfeld (Abbildung 4.4 3a).

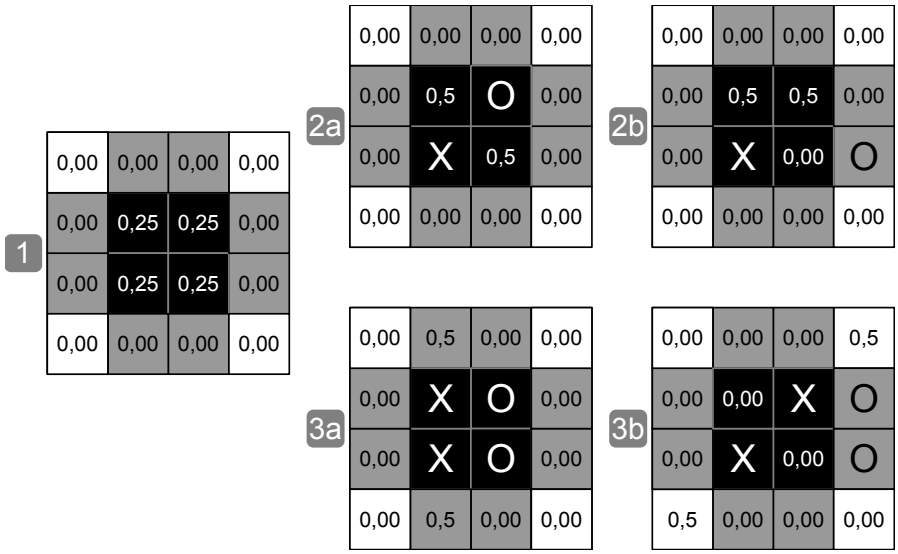

Abbildung 4.4 Strategie um die Mitte zu kontrollieren.

Sollte Agent Bob eine andere gelernte Strategie verfolgen und zum Beispiel ein Randfeld besetzen, dann werden Agentin Alice andere Aktionen von ihrer Strategie Vorgeschlagen. In einem alternativen Spielverlauf (Abbildung 4.4 2b) setzt Agent Bob seine Spielfigur nicht auf das Spielfeld (1,2), sondern auf das Spielfeld (2,3). Die Strategie der Kontrolle der Mittelfelder bevorzugt Mittelfelder, die in einer leeren vertikalen, horizontalen oder diagonalen Reihe sind, bezogen auf die bereits gesetzte Spielfigur. Das Mittelfeld (2,2) wird uninteressant für die Strategie von Alice, weil eine mögliche horizontale Reihe aus vier gleichen Spielsteinen bereits nicht mehr möglich ist. Dahingegen schlägt die Strategie die Mittelfelder (1,1) und (1,2) mit einer numerischen Belohnung von +0,5 vor. Setzt Alice ihre Spielfigur auf das Mittelfeld (1,1), dann ist eine vertikale Verbindung von vier gleichen Spielfiguren möglich und setzt Alice ihre Spielfigur auf das Mittelfeld (1,2), dann ist eine diagonale Verbindung von vier gleichen Spielfiguren möglich.

Agentin Alice entschiedet durch Zufall ihre Spielfigur auf das Mittelfeld (1,2) zu setzen, denn der Agent soll durch Zufall entscheiden, wenn für mehrere Aktionen in einem Zustand die gleich Hohe größtmögliche Belohnung vergeben wird. Agent Bob setzt daraufhin seine Spielfigur auf das Randfeld (1,3). Ein neue Spielsituation (Zustand) entsteht (Abbildung 4.4 3b). Alice erhält von der Strategie der Mittelfeld Kontrolle die Aktionsoptionen (0,3) und (3,0). Beide Aktionsoptionen werden mit +0,5 belohnt. Diese Strategie berücksichtigt nicht, dass Agent Bob bereits zwei Spielsteine in einer ungestörten vertikalen Verbindung positioniert hat. Daher kann Alice durch Zufall entscheiden welche Aktion sie ausführt. Eine bessere Strategie würde Alice in diesem Zustand diese Option nicht lassen, denn das Eckfeld (0,3)

ist in dieser Spielsituation attraktiver als das Eckfeld (3,0). Eckfeld (0,3) erweitert die diagonale ungestörte Verbindung von Agentin Alice auf eine Länge von drei und gleichzeitig würde die vertikale ungestörte Verbindung von Agent Bob gestört werden. Eine optimale Strategie würde Alice mit einer Belohnung von +0,75 das Eckfeld (0,3) empfehlen.

Bevor wir Alice und Bob die Möglichkeit geben die Strategie selber zu verbessern beziehungsweise eigenständig zu entwickeln und solche Auffälligkeiten und Muster in die Strategie zu integrieren, werden wir die Ausgangsstrategie noch um eine Teilstrategie erweitern.

Verteidigung ist der beste Angriff

Diese Strategie konzentriert sich darauf, gegnerische Stellungen zu erkennen und dahingehend Gengenmaßnahmen einzuleiten. Abbildung 4.5 verdeutlicht gefährliche Spielsituationen, in denen die Strategie Gegenmaßnahmen vorschlagen sollte. Horizontale Verbindugsmöglichkeiten (1a - 1d)

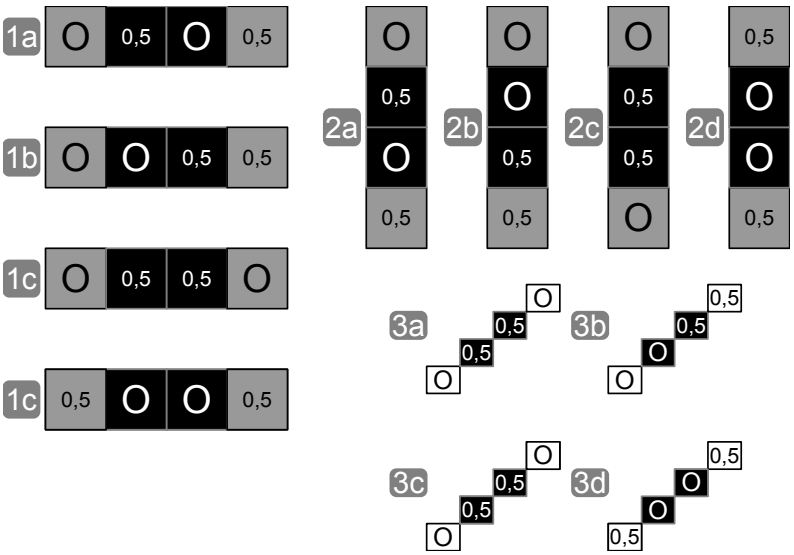

Abbildung 4.5 TicTacToe Angriffsstrategien (aus der Sicht von Bob) und Verteidigungsstrategien (aus der Sicht von Alice).

4.1.2 Lernen ohne Strategie

4.2 Reversi

Das Spiel Reversi oder auch Othello genant, wird auf einem 8x8 Spielbrett gespielt. Es ist ein Spiel für zwei Personen die gegeneinander antreten. Eine Person setzt weiße runde Spielsteine und die andere Person schwarze runde Spielsteine. Jede neue Partie Reversie beginnt im selben Ausgangszustand (siehe Abbildung 4.6). Die Spieler setzen nacheinander in ihren Spielzügen genau einen Spielstein. Wie beim klassischen Tic Tac Toe aus Abschnitt 4.1 behalten die Spieler während des gesamten Spiels ihre Spielsteinfarbe und einmal gesetzte Spielsteine können ihre Position nicht mehr verändern.

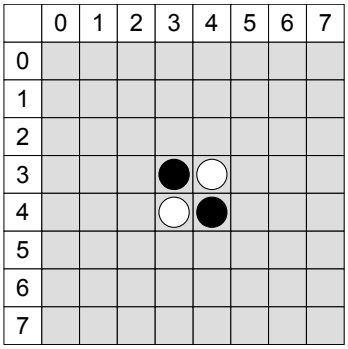

Abbildung 4.6 Ausgangssituation einer beginnenden Partie Reversi. Die äußeren weiß hinterlegten Reihen in denen sich Zahlen befinden, dienen dazu die Positionen der einzelnen Spielfelder genau zu definieren. In der Ausgangsspielsituation befinden sich bereits 2 weiße Spielsteine an den Positionen (3,4) und (4,3) und zwei schwarze Spielsteine an den Positionen (3,3) und (4,4).

Eine Besonderheit von Reversi ist, dass gesetzte Spielsteine ihre Farbe ändern können. Werden z.B. zwei weiße Spielsteine von zwei schwarzen in einer horizontalen Linie eingeschlossen, dann werden die weißen Spielsteine in schwarze umgewandelt beziehungsweise umgedreht. Das erobern der gegnerischen Spielsteine ist vom aktuell gesetzten Spielstein abhängig. Die Siegesbedingung von Reversi ist es, am Ende des Spiels, mehr Spielsteine seiner eigenen Farbe zu haben als der Gegner Spielsteine in seiner Farbe hat. Das Spiel endet, wenn keiner der beiden Spieler mehr einen Spielstein, nach den Regeln des Spiels, auf das Spielbrett setzen kann.

Spielzüge sind bei Reversi nicht beliebig, sie unterliegen bestimmten Regellungen. Eine Regel für das Setzen eines Spielsteins ist, nur wenn mindestens ein geg-

nerischer Spielstein erobert wird, darf ein Spielstein an diese Stelle gesetzt werden. Weiterhin darf ein Spielstein nur dann gesetzt werden wenn, ein anderer Spielstein(Anker), mit der gleichen Farbe, in einer diagonalen, vertikalen oder horizontalen Linie, existiert. Es dürfen auch keine freien Felder zwischen dem zu setzendem Stein und dem Anker liegen. Ein Anker ist ein Spielstein mit der selben Farbe wie der zu setzende Spielstein. Ein zu setzender Spielstein kann mehrere Anker haben, aber er muss mindestens einen und kann maximal acht Anker haben.

Der Anker in Abbildung 4.7 ist der schwarze Spielstein an der Stelle (3,3). (a) In diesem Beispiel soll schwarz am Zug sein und einen Spielstein platzieren. Die Positionen (3,1) und (3,6) ermöglichen eine horizontale, (5,3) ermöglicht eine vertikale und (5,1), (0,6) und (7,7) ermöglichen eine diagonale Verbindung mit dem Anker auf Position (3,3). Die meisten gegnerischen Spielsteine könnte schwarz erobern, indem er seinen Spielstein auf das Spielfeld (7,7) setzt. (b) Setzt der Spieler seinen schwarzen Spielstein an die Position (3,1), dann hat dieser 3 Anker. Einen vertikalen Anker (0,1), einen horizontalen Anker (3,3) und einen diagonalen Anker (0,4). Insgesamt würden 5 weiße Spielsteine erobert werden, also 2 mehr als in (a) maximal möglich wären.

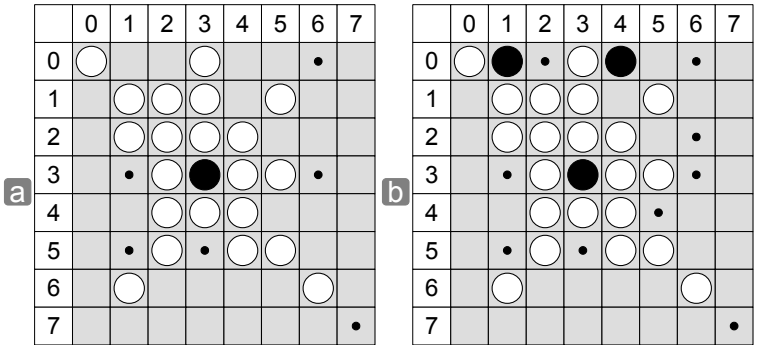

Abbildung 4.7 Zwei möglicherweise nicht in der Praxis auftretende Spielsituationen, die einzig verdeutlichen sollen welche Zugmöglichkeiten der Spieler mit den schwarzen Spielsteinen hat und warum nur diese Züge möglich sind. Die kleinen schwarzen Punkte zeigen die Positionen an denen ein schwarzer Spielstein gesetzt werden darf. (a) Eine Spielsituation mit maximal einem möglichen Anker. (b) Eine Spielsituation mit maximal 3 möglichen Ankern für die Position (3,1).

4.2.1 Lernen mit Strategie

4.2.2 Lernen ohne Strategie

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.

[RN12] Stuart Russel und Peter Norvig. *Künstliche Intelligenz: Ein moderner Ansatz*. 3. Aufl. Pearson, 2012.

[Zob70] Albert L. Zobrist. „A new hashing method with application for game playing“. In: *Technical Report* 88 (1970), S. 1–12.

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!