Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Miriam Tödten

Einsatz eines maschinellen Lernverfahrens in einem Othello-Spielprogramm

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Miriam Tödten

Einsatz eines maschinellen Lernverfahrens in einem Othello-Spielprogramm

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter: Prof. Dr. rer. nat. Michael Neitzke

Abgegeben am 1. Juli 2008

Miriam Tödten

Thema der Bachelorarbeit
Einsatz eines maschinellen Lernverfahrens in einem Othello-Spielprogramm
Stichworte
Bewertungsfunktion, temporale Differenz, Multi Step Prediction Problem
Kurzzusammenfassung
Diese Arbeit untersucht den Einsatz eines maschinellen Lernverfahrens zum Lernen einer Bewertungsfunktion für Othello-Spielpositionen, wobei kein strategisches Wissen benutzt wird. Das Lernverfahren verwendet wie Samuels Dame-Programm eine temporale Differenz in einem Selfplay-Modus, um beim Spielen mehrmals eine Bewertungsfunktion P_z zu einer neuen Bewertungsfunktion P_{z+1} zu ändern, indem es die Gewichte anpasst.
In mehreren Versuchsgruppen werden gleichartige Versuche gestartet, die sich in ihren zufälligen Startgewichten unterscheiden. Für die Bewertung wird untersucht, ob das Lernverfahren in gleichartigen Versuchen ähnliche Bewertungsfunktionen produziert und ob die gelernten Bewertungsfunktionen als spielstark zu bewerten sind.

Miriam Tödten

Title of the paper
Machine learning algorithm for an Othello-game program
Keywords
evaluation function, temporal-difference method, multi-step prediction problem
Abstract
This Paper is concerned with a temporal-difference method to learn an evaluation function that can be used in a MinMax search in a game called Othello. Like in Samuels checker program a temporal difference is used to learn the evaluation function by changing its weights. The described approach does not use any strategic knowledge of Othello.
Main Part of the investigation concerns the question of similarity of learning results in different groups of similar experiments starting with random weights.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Ausgangspunkt für diese Arbeit war ein im Rahmen der Veranstaltung Software Engineering 2 entstandenes Othello[1]-Programm für zwei menschliche Spieler, das sich über ein TCP/IP-Netzwerk spielen ließ, und ein anderes Zweipersonen-Nullsummenspiel, das im Rahmen der Intelligente Systeme Veranstaltung entstanden ist, bei dem ein menschlicher Spieler gegen den Computer spielt, der ein Spielbaum-Suchverfahren benutzt, um die Züge zu bestimmen. Da Spielbaum-Suchverfahren, nur zum Teil vom konkreten Spiel abhängig sind, konnte der Programmcode der beiden Veranstaltungen mit wenig Aufwand kombiniert werden, so dass ein Othello-Programm entstanden ist, bei dem ein menschlicher Spieler gegen den Computer spielen kann. Othellos Spielbaum-Komplexität ist nach van den Herik u. a. (2002) mit 10^{58} zwar kleiner als beim Schach, das eine Spielbaumkomplexität von 10^{123} hat (vgl. van den Herik u. a. (2002); zitiert nach Jin (2007)), trotzdem ist sie groß und das Suchverfahren des Computerspielers hat den Spielbaum nicht bis zum Spielende aufgebaut. D.h. der Spielbaum wurde nur bis zu einer gewissen Tiefe aufgebaut und die Othello-Spielpositionen in den Blättern wurden bewertet, wobei im ersten Ansatz die Anzahl der Steine des Computerspielers als Bewertung benutzt wurde. Allerdings konnte der Computerspieler, der diese Bewertung benutzt, geschlagen werden. Dabei handelt es sich um das allgemeine Problem, dass bei Spielbaum-Suchverfahren, die die Bäume nur bis zu einer bestimmten Tiefe aufbauen, die Blätter dieses unvollständigen Suchbaums einen Suchhorizont bilden und das Ergebnis des Spiels hinter diesem Suchhorizont verborgen ist. Je schlechter die Bewertungen der Blätter das Spielergebnis voraussagen, umso schlechter spielt der Computerspieler.

Die nächste Idee war, dass das Othello-Programm selbständig beim Spielen lernen soll, wie es die Spielpositionen bewerten muss, so wie das Programm von Samuel (1959) durch Spielen lernte, wie Dame-Spielpositionen zu bewerten seien. Im Kern spielten in diesem Programm zwei Software-Spieler gegeneinander, wobei einer der Spieler die Bewertungsfunktion für Dame-Spielpositionen lernte. Die Bewertungsfunktion hatte die Form eines linearen Polynoms mit n Unbekannten $x_1, x_2, ..., x_n$, die mit n Gewichten $w_1, w_2, ..., w_n$ multipliziert wurden. Dabei wurden die x_i aus Aspekten der jeweiligen Spielposition, sogenannte Features,

[1] Trademarks of Anjar Co., ©1973, 2004 Anjar Co.

abgeleitet und das Programm versuchte durch Änderung der w_i eine Bewertungsfunktion zu lernen. Zum Ändern der Gewichte hat Samuel die Differenz der Bewertungen zweier Spielpositionen benutzt und sie als Fehler betrachtet, wobei er davon ausgegangen ist, dass die spätere Spielposition die bessere Voraussage auf das Spielergebnis macht. Daraus folgte, dass die Bewertungsfunktion so geändert werden musste, dass sie Bewertungen produziert, die der Bewertung der späteren Spielposition näher kommen.

Diese Arbeit untersucht den Einsatz eines Lernverfahrens für Othello-Spielpositionen, das von Samuels Programm inspiriert eine temporale Differenz nutzt. Als Features werden nur einfache Aspekte der Spielpositionen betrachtet, d.h. das Lernverfahren nutzt kein strategisches Wissen in Form von strategischen Features. Die Features, die betrachtet werden, kann sich jeder Anfänger aus den Spielregeln ableiten, ohne ein Othello-Spieler zu sein.

Der praktische Teil zu dieser Arbeit besteht aus der Implementierung eines Lernverfahrens zum Lernen einer Bewertungsfunktion in Form eines Polynoms mit mehreren Unbekannten[2], der Anwendung dieses Lernverfahrens in mehreren Versuchgruppen und einer grafischen Oberfläche, mit der ein menschlicher Spieler ein Othello-Spiel gegen einen Computerspieler spielen kann. Zum Bewerten der Spielpositionen benutzt der Computerspieler in seinem Spielbaum-Suchverfahren die in den Versuchen gelernten Bewertungsfunktionen. Die Implementierung des Lernverfahrens und der Oberfläche trägt den Namen Othilie.

1.2 Aufbau der Arbeit

In Kapitel 2 werden die Grundlagen dargestellt, die bei der Implementierung von Othilie eine Rolle spielen. Das sind die Othello-Spielregeln, die Anfänge der Spielbaum-Suchverfahren, soweit sie von Othilie benutzt werden, und Grundlagen zum maschinellen Lernen, die nötig sind, um die Lernaufgabe von Othilie zu spezifizieren und um darzustellen, welche Arbeiten dem Entwurf von Othilies Lernverfahren zugrunde liegen.
Kapitel 3 spezifiziert die Lernaufgabe und das Lernverfahren von Othilie und stellt dar, wie die Ergebnisse des Lernverfahrens getestet werden. Außerdem enthält das Kapitel eine Darstellung aller Features, die bei der Bewertung einer Spielposition betrachtet werden.
In Kapitel 4 werden sowohl die Architektur als auch Details der Realisierung vorgestellt.
Kapitel 5 beschreibt die Parameter, mit den fünf Versuchsgruppen gestartet wurden, und enthält die Darstellung der Ergebnisse.
In Kapitel 6 wird in einer weiteren Versuchsgruppe untersucht, wie sich die Modifikation der temporalen Differenz auswirkt.

[2] In dieser Arbeit wird sowohl der Begriff Bewertungsfunktion als auch der Begriff Polynom verwendet.

2 Grundlagen

2.1 Othello-Spielregeln

Die folgenden, dieser Arbeit zugrunde liegenden Spielregeln sind Rose (2005) entnommen.

Ein Othello-Spiel besteht aus einem 8×8-Spielfeld und 64 Steinen, mit einer hell- und einer dunkel-farbigen Seite; meist schwarz und weiß. Jedem der beiden Spieler wird eine Farbe zugeordnet und die Spieler werden entsprechend als Schwarz oder Weiß bezeichnet. Abwechselnd legen die beiden Spieler jeweils einen Stein, mit der ihnen zugeordneten Farbe nach oben auf das Spielfeld. Dabei gilt, dass ein Stein immer so auf das Spielfeld gelegt werden muss, dass mindestens ein gegnerischer Stein umgelegt wird und ein gegnerischer Stein kann umgelegt werden, wenn er sich auf einer Geraden zwischen dem aktuell gelegten Stein und einem anderen Stein des Spielers befindet. Befinden sich mehrere gegnerische Steine zwischen diesen beiden Steinen, dürfen diese alle umgedreht werden. Zum Umdrehen kommen alle acht Richtungen in Frage.

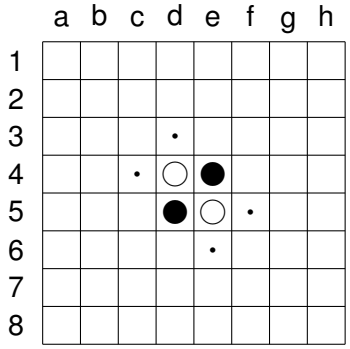


Abbildung 2.1: Startposition

Ein Spieler, der nach diesen Regeln keinen gültigen Zug hat, aussetzen. Das Spiel ist beendet, wenn beide Spieler nicht setzen können oder alle Felder belegt sind. Am Ende hat

der Spieler gewonnen, der die meisten Steine seiner Farbe auf dem Spielfeld hat. Haben beide Spieler die gleiche Anzahl an Steinen, wird das Spiel als unentschieden gewertet. In Wettkämpfen wird dagegen ein Spielstand mit der genauen Anzahl der Steine festgelegt.

Abbildung 2.1 zeigt die Startposition des Othello-Spiels. Schwarz ist am Zug und die Felder, auf die Schwarz setzen kann, sind durch einen kleinen Punkt markiert.

2.2 Spielbaum-Suchverfahren

Bei Othello gilt, dass der Gewinn des einen Spielers genau den Verlust des anderen bedeutet, dass jeder Spieler zu jedem Zeitpunkt das Spielbrett vollständig überblicken kann und dass es kein zufälliges Element wie beispielsweise einen Würfel gibt. Daher handelt es sich bei Othello um ein Zweipersonen-Nullsummenspiel mit vollständiger Information. Das bedeutet, dass mittels Spielbaum-Suchverfahren (Minimax-Suche, Minimax-Verfahren) der bestmögliche Zug für eine Spielposition bestimmt werden kann. Im Folgenden werden die wesentlichen Meilensteine der Entwicklung der Spielbaum-Suchverfahren bis zu dem in dieser Arbeit benutzen Alpha-Beta-Algorithmus dargestellt[3].

2.2.1 Anfänge der Spielbaum-Suchverfahren

John von Neumann (1928) hat die Frage untersucht, wie ein Spieler in einem n-Personen-Spiel spielen muss, um „dabei ein möglichst günstiges Resultat zu erzielen“, wobei „das Schicksal jeden Spielers außer von seinen eigenen Handlungen auch noch von denen seiner Mitspieler“ abhängt. Für den Fall eines Zweipersonen-Spiels, bei dem die beiden Spieler S_1 und S_2 jeweils nur Einfluss auf eine der beiden Variablen x_1 und x_2 haben, stellt von Neumann dar, dass - wenn sich ihre Gewinne wegen der entgegengesetzten Interessen als $g_1 = g$ und $g_2 = -g$ ergeben - der Gewinn, den S_1 erzielt, durch

$$max_{x_1}(min_{x_2}(g(x_1,x_2))) \tag{2.1}$$

gegeben ist, wenn S_2 optimal spielt. Entsprechend ist der Gewinn den S_2 erzielt, wenn S_1 optimal spielt, durch

$$min_{x_2}(max_{x_1}(g(x_1,x_2))) \tag{2.2}$$

gegeben. Nach Rieck (2006), der die Formel 2.1 als Maximin-Strategie und die Formel 2.2 als Minimax-Strategie bezeichnet, hat John von Neumann in seiner Arbeit für den Fall n=2

[3] Eine Darstellung der optimierten Varianten von Spielbaum-Suchverfahren ist z.B. bei Hartz (2005) zu finden, der ihre Verwendung im Hinblick auf die Benutzung auf ressourcenarmen Systemen betrachtet hat.

bewiesen, dass es für die zwei Spieler S_1 und S_2 in einem Nullsummenspiel eine Strategiekombination gibt, für die die Formel

$$max_{x_1}(min_{x_2}(g(x_1,x_2))) = min_{x_2}(max_{x_1}(g(x_1,x_2))) \tag{2.3}$$

gilt (Minimax-Theorem). Diese Formel beschreibt die Bedingung für ein Nash-Gleichgewicht, also das Gleichgewicht der Spieltheorie für Zweipersonen-Nullsummenspiele, bei dem es sich für einen der Spieler nicht lohnt, als einziger von der Strategiekombination abzuweichen, wenn sie sich beide unabhängig voneinander auf die Variablen x_1 und x_2 festlegen müssen.

Bei Zweipersonen-Nullsummenspielen mit vollständiger Information liegt eine leicht veränderte Situation vor, weil die beiden Spieler abwechselnd spielen und so alle vorherigen Züge und deren Auswirkungen kennen. Der Spieler, der am Zug ist, kann dann - wenn er zwei Züge im Voraus betrachtet und davon ausgeht, dass der Gegner optimal spielt - seinen Zug mittels der Formel 2.1 bestimmen.

Mit der Absicht zu zeigen, dass ein General Purpose Computer zum Schach spielen eingesetzt werden kann, hat Shannon (1950) eine Arbeit veröffentlicht, die für ein Schachprogramm einen Prozess abwechselnder Maxi- und Minimierungen nach der Formel 2.1 skizziert, um den Computer für eine gegebene Spielposition einen guten Spielzug bestimmen zu lassen. Dieser Prozess wird als Minimax-Verfahren oder Minimax-Suche bezeichnet und ist die grundlegende Idee der Spielbaum-Suchverfahren. Der Suchraum dieser Verfahren kann als Spielbaum aus Knoten, die Spielsituationen repräsentieren, dargestellt werden. Die Wurzel entspricht der aktuellen Spielposition und die Nachfolger eines Knotens werden durch die Menge der möglichen Züge bestimmt. Um den nächsten Zug zu bestimmen, könnte prinzipiell ein Spielbaum aufgebaut werden, der alle möglichen Folgezüge bis zum Spielende enthält. Die Blätter in diesem Spielbaum repräsentierten dann Endstellungen, die mit gewonnen, unentschieden oder verloren z.B. durch Werte aus {-1, 0, 1} bewertet werden könnten. Diese Bewertungen könnten dann nach dem Minimax-Verfahren von den Blättern zurück zur Wurzel übertragen werden, so dass der Computer ein perfektes Spiel spielen könnte. Da die Anzahl der Knoten dieses Spielbaums, mit steigender Baumtiefe exponentiell wächst, hat Shannon im Suchbaum nur vier[4] Züge betrachtet und eine grobe Bewertungsfunktion angegeben, um die Blätter in diesem unvollständigen Suchbaum mit dem erwarteten Gewinn zu bewerten. Shannon stellt dar, dass mit einer Bewertungsfunktion, die für eine Spielposition das tatsächliche Spielergebnis bestimmt, eine Maschine konstruiert werden kann, die perfekt spielt.

In dem Dame-Spiel von Samuel (1959), dessen Kern ein Minimax-Verfahren ist, ist die Bewertungsfunktion zum Schätzen der Blätter im unvollständigen Suchbaum ein lineares Poly-

[4] Da der Zug eines Spieler beim Schach als Halbzug bezeichnet wird, sind es im Schach-Jargon eigentlich zwei Züge. In dieser Arbeit ist aber ein Zug der Spielzug eines Spielers.

nom $P(x,w) = x^T w$, wobei die Elemente des x-Vektors durch gewisse Spielfeld-Features einer Spielposition bestimmt werden. Da die Gewichte des w-Vektors von Samuels Programm durch das Beobachten von Spielen gelernt wurden, ist diese Arbeit ausführlicher im Abschnitt über maschinelles Lernen beschrieben. Reinefeld (2006) beschreibt, dass Samuel nach Knuth und More in diesem Programm bereits als Spielbaum-Suchverfahren einen Alpha-Beta-Algorithmus benutzt hat, um eine Effizienzsteigerung zu erzielen, weil nur noch Teile des Suchbaums durchsucht werden müssen. Samuel hat das aber erst 1967 in seiner zweiten Veröffentlichung erwähnt. Nach Samuel ist der Erfinder dieses Verfahrens Mc Carthy, der seine Beobachtung allerdings nicht schriftlich festgehalten hat.

2.3 Maschinelles Lernen einer Bewertungsfunktion

Nach Simon (1980) findet Lernen statt, wenn das lernende System Änderungen vornimmt, so dass es ihm möglich wird dieselbe oder eine ähnliche Aufgabe beim nächsten Mal effektiver oder effizienter zu erledigen. Wrobel u. a. (2003) kritisieren diese Definition, indem sie anmerken, dass ein Messer besser schneiden kann, nachdem es geschliffen worden ist, dabei aber kaum ein Lernen stattgefunden hat. Um eine präzise Vorstellung vom Gebiet des maschinellen Lernens und Data Minings zu vermitteln, wählen sie einen Ansatz, der das Gebiet durch das Aufzählen der verschiedenen, maschinell lösbaren Lernaufgaben beschreibt. Die Beschreibung der Lernaufgabe von Othilie wird diesem Ansatz folgen.

2.3.1 Lernaufgabe

Eine Lernaufgabe wird bei Wrobel u. a. (2003) im Wesentlichen durch die Beschreibung der Eingaben und der vom lernenden System erwarteten Ausgaben beschrieben. Hinzu können Angaben zu Randbedingungen wie maximale Laufzeiten oder Speicherverbrauch kommen

Das Lernen einer Bewertungsfunktion, mit der in den Blättern eines unvollständigen Spielbaums das Spielergebnis geschätzt bzw. vorausgesagt werden kann, ist eine sogenannte prädikative Lernaufgabe, bei der es nach Wrobel darum geht, eine unbekannte Funktion $f : X \to Y$ durch eine gelernte Funktion $h : X \to Y$ zu approximieren, so dass später für alle möglichen Instanzen x des Instanzenraums X mit $h(x)$ der Wert für $f(x)$ möglichst gut vorhergesagt werden kann. Im Gegensatz dazu sind die Ergebnisse der deskriptiven Lernaufgaben auf Teilbereiche des Instanzenraums X beschränkt, d.h. es ist nicht ihr Ziel Aussagen über alle Instanzen von X zu machen. Die unbekannte Funktion f ist im Rahmen dieser Arbeit die ideale Bewertungsfunktion, die jeder in einem Othello-Spiel möglichen Spielposition $x \in X$, das Spielergebnis $y \in Y$ zuordnet, mit der nach Shannon (1950) „leicht eine Maschine

entwickelt werden könnte, die in der Lage wäre ein perfektes Spiel zu spielen“, ohne dass der Suchbaum vollständig aufgebaut werden muss. Die Funktion h ist die von Othilie gelernte Bewertungsfunktion für Othello-Spielpositionen, die im unvollständigen Spielbaum benutzt werden soll, um das Spielergebnis in den Blättern zu schätzen.

Auch wenn das Lernen einer Bewertungsfunktion für Othello dem von Wrobel definierten Funktionslernen aus Beispielen ähnelt, unterscheiden sich die beiden Lernaufgaben voneinander. Nach Wrobel wird bei der Lernaufgabe Funktionslernen aus Beispielen dem lernenden System eine Menge E von Beispielen zur Verfügung gestellt. Es ist

$$E = \{(x,y)|x \in X, y \in Y, y = f(x)\} \tag{2.4}$$

Vom lernenden System wird dann als Ausgabe die Funktion h (Hypothese) erwartet, für die gilt, dass der wahre Fehler von $h(x)$ und $f(x)$ möglichst gering ist, wenn die Instanzen beim Bestimmen dieses Fehlers und innerhalb der Beispielmenge E gemäß derselben Wahrscheinlichkeitsverteilung verteilt sind.

Für die Endpositionen eines Othello-Spiels, in denen das Spiel entschieden ist, lässt sich so ein y beispielsweise aus $\{gewonnen, verloren, unentschieden\}$ leicht festlegen, aber die Bewertungsfunktion von Othilie soll Aussagen über Spielpositionen machen, die keine Endpositionen sind. Solche y für Nicht-Endpositionen stehen aber nicht zur Verfügung.

2.3.2 Lernen mit temporaler Differenz

Samuel (1959) hat mit seinem Programm, das erfolgreich eine Bewertungsfunktion für das Dame-Spiel gelernt hat, ein Prinzip beschrieben, das ohne y-Werte für Nicht-Endpositionen auskommt: Es werden die Bewertungen zweier Spielpositionen eines Spiels verglichen, um darauf basierend die Bewertungsfunktion P_i zur neuen Bewertungsfunktion P_{i+1} zu verändern. Samuels Bewertungsfunktion ist ein lineares Polynom mit mehreren Unbekannten

$$P(x,w) = w_1x_1 + w_2x_2 + \ldots + w_nx_n = t_1 + t_2 + \ldots + t_n = w^T x \tag{2.5}$$

wobei die $x_i \in \mathbb{R}$ den Features einer Spielposition entsprechen. Anzahl der passiven Steine oder Anzahl der passiven Steine, an die mindestens ein leeres Feld grenzt, sind zwei Beispiele für Features, die Samuel für Dame-Spielpositionen definiert hat. Die $w_i \in \mathbb{R}$ sind Gewichte der jeweiligen Features, die vom Programm beim Lernen geändert werden; die Ausnahme bildet dabei das Gewicht des Features Materialvorteil (piece advantage), Samuel hat ihn auf einen festen Wert gesetzt. Die lernende Software-Komponente - die Alpha-Komponente - ermittelt bei jedem ihrer Züge mittels der aktuellen Bewertungsfunktion P_z per

unvollständiger Minimax-Suche den bestmöglichen Zug und verändert die Bewertungsfunktion zu P_{z+1} wie folgt: Wenn x_z die aktuelle Spielposition beschreibt und x_b die Spielposition, die als bestmögliches Blatt ermittelt wurde dann bestimmt Samuel

$$\delta = P_z(x_b) - P_z(x_{z-1}) \tag{2.6}$$

als Fehler. Hinter dieser Differenz steht Samuels Überlegung, dass das Ergebnis einer Bewertungsfunktion von besserer Qualität ist, wenn sie im späteren Spielverlauf eingesetzt wird. Für positive δ stellte Samuel fest, dass positiv beitragenden Termen t_i mehr Gewicht gegeben werden sollte und negativ beitragenden Termen weniger. Für negative δ gilt, dass positiv beitragenden Termen weniger Gewicht gegeben werden sollte und negativ beitragenden Termen mehr. Othilie basiert auf genau diesen Feststellungen und setzt sie direkt um. Samuel setzt sie allerdings nicht direkt um, sondern nimmt sie als Basis für weitere Berechnungen. Am Ende eines Spiels werden die Vorzeichen jeden Terms $t_{z,i}$ mit dem Vorzeichen von δ_z korreliert und die Gewichte darauf basierend so geändert, dass dem Gewicht des Terms mit dem größten Korrelationskoeffizienten ein Maximalwert zugewiesen wurde und den anderen Koeffizienten entsprechend kleinere Werte. Eine weitere Software-Komponente - die Beta-Komponente - wurde als Gegner für die Alpha-Komponente konstruiert. Sie bestimmt ihre Züge mittels unvollständiger Minimax-Suche und einer Bewertungsfunktion, die für die Dauer eines Spiels konstant ist. In einem Selfplay-Modus können die Alpha- und die Beta-Komponente gegeneinander mehrere Spiele spielen. Hat die Alpha-Komponente das Spiel gewonnen, bekommt Beta dessen Bewertungsfunktion. Nach einem Test und einer Modifikation des Programms war es zum Austausch der Bewertungsfunktion erforderlich, dass die Alpha-Komponente eine Mehrheit der Spiele gewonnen hat. Hat Alpha dreimal verloren, wird der Koeffizient in Alphas Bewertungsfunktion, der den größten Wert hat, auf null gesetzt. Die Notwendigkeit dieser Maßnahme hat Samuel damit begründet, dass sich das Programm aus einem lokalen Maximum befreien würde. Am Ende erreichte Samuel mit der gelernten Bewertungsfunktion ein Dame-Spiel, das er als besser als durchschnittlich bezeichnete und das, obwohl er die Menge der benutzen Features als redundant und unvollständig beschreibt.

Nach Sutton (1988) ist Samuels Idee, die Differenz zweier Bewertungen als Fehler zu betrachten und die Bewertungsfunktion entsprechend zu ändern, "die früheste bekannte Anwendung einer TD-Methode [Temporal Difference Methode]". Sutton beschreibt TD-Methoden als passende Lernverfahren für Multi Step Prediction Probleme. Ein Multi Step Prediction Problem definiert er als ein Problem, bei dem die Eingabe in Form einer Folge von Beobachtungen

$$x_1, x_2, \ldots, x_m, e \tag{2.7}$$

vorliegt. Das beobachtete System geht dabei von einem Startzustand über eine Folge von Zuständen in einen Endzustand über. Dabei ist e das Ergebnis, das im Endzustand beob-

achtet wird, und die x_z mit $z = 1,\ldots,m$ sind Observation-Vektoren[5] (observation vectors) in den Nicht-Endzuständen, auf dessen Basis Voraussagen P_z über e getroffen werden. Typisch für ein Multi Step Prediction Problem ist, dass zwischen einer Beobachtung x_z mit $z = 1,\ldots,m-1$ und dem beobachteten Ergebnis e weitere Beobachtungen liegen.

Beim Ansatz des Supervised Learnings würde dem lernenden System die Eingabe E in Form von Tupeln aus Beobachtungen x_i und Ergebnis z übergeben:

$$E = \{(x_1,e),(x_2,e),\ldots,(x_m,e)\} \tag{2.8}$$

Beim Lernen werden dann die Gewichte der Funktion $P(x_z,w) = P_z$, die die Voraussagen macht, mittels

$$\Delta w_z = \alpha(e - P_z)\nabla_w P_z \tag{2.9}$$

geändert. Wobei für $P(x,w) = x^T w$ gilt, dass $\nabla_w P = x$. Durch Einsetzen in 2.9 erhält man die Widrow-Hoff-Regel

$$\Delta w_z = \alpha(e - w^T x_z)x_z \tag{2.10}$$

Als für Multi Step Prediction Probleme geeigneter beschreibt Sutton den Ansatz des Temporal Difference Learnings, bei dem die Differenz zweier aufeinanderfolgender Vorraussagen benutzt wird, um die Gewichte zu ändern. Die Gewichtsänderungen der TD(λ)-Methode für allgemeine λ werden nach

$$\Delta w_z = \alpha(P_{z+1} - P_z)\sum_{k=1}^{z}\lambda^{z-k}\nabla_w P_k \tag{2.11}$$

berechnet.

Für die TD(0)-Methode, die die Gewichte nach

$$\Delta w_z = \alpha(P_{z+1} - P_z)\nabla_w P_z \tag{2.12}$$

ändert, zeigt Sutton, dass sie für jede Initialisierung des Gewichtsvektors w_0 bei n-maliger Präsentation einer endlichen Trainingsmenge bei $n \to \infty$ gegen die optimalen Voraussagen, also gegen die optimalen Gewichte der Funktion P, konvergiert, wenn $\alpha > 0$ klein genug ist und wenn die Observation-Vektoren linear unabhängig sind, wobei Sutton die Änderungen, die im Verlauf einer Zustandsfolge bis zum Endzustand aufsummiert und am Ende der Folge die Gewichtsänderungen vornimmt. Die TD(0)-Methode ist damit im Vorteil gegenüber der Methode, die die Gewichte nach der Widrow-Hoff-Regel (Formel 2.10) ändert, denn diese konvergiert gegen Werte, die lediglich den Fehler auf der Trainingsmenge minimieren und nicht gegen ideale Werte. Das bedeutet, dass beide Verfahren bei unendlicher Erfahrung

[5] Die Elemente eines Observation-Vektors ergeben sich bei Othilie aus den einzelnen Features, die die betrachtete Spielposition hat.

dieselben Ergebnisse produzieren, die TD(0)-Methode bei limitierter Erfahrung die besseren Voraussagen machen kann.

Mit seinen Game Playing-Beispiel vermittelt Sutton eine intuitive Vorstellung davon, warum TD-Methoden effizienter arbeiten als Supervised Learning Methoden: Angenommen ein lernendes System hat für einen Zustand gelernt, dass er zu 90 Prozent zu einer Niederlage führt und zu 10 Prozent zum Sieg. Das System kommt dann in einen neuen Zustand, in dem es vorher noch nicht gewesen ist. Der Folgezustand des neuen ist der Zustand, der zu 90 Prozent zur Niederlage führt. Je nachdem, ob in diesem Fall eine Niederlage oder ein Sieg beobachtet würde, die Supervised Learning-Methode würde ein Tupel aus dem neuen Zustand und dem Ergebnis des Endzustandes bilden und das System lernt entweder, dass der neue Zustand zu 100 Prozent zur Niederlage führt, oder es lernt, dass der neue Zustand zu 100 Prozent zum Sieg führt. Eine TD-Methode würde dagegen ein Tupel aus dem neuen Zustand und dem 90%-Zustand bilden. Folglich kann es lernen, dass der neue Zustand zu 90 Prozent zur Niederlage führt. Dieses Beispiel veranschaulicht, dass - auch wenn beide

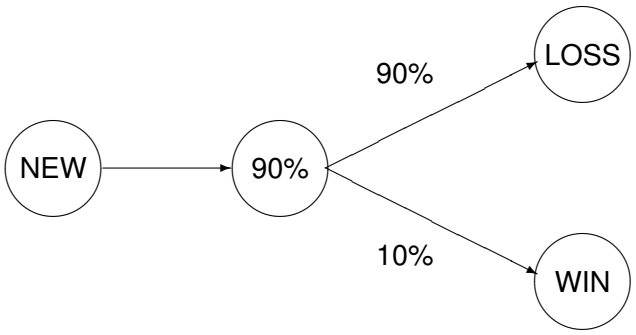


Abbildung 2.2: Game Playing-Beispiel Quelle: (Sutton, 1988)

Methoden bei einer unendlichen Anzahl von Spielen zum selben Ergebnis kommen - die TD-Methode besseren Gebrauch ihrer Erfahrung macht und so schneller konvergiert. Bei einer endlichen Trainingsmenge kann sie daher die besseren Voraussagen liefern.

Sutton merkt zu Samuels Dame-Programm an, dass sich die Behandlung der Endzustände von der Behandlung der Endzustände bei den TD-Methoden unterscheidet. Die Voraussagen der Endzustände werden bei Samuel nicht auf das Ergebnis e eines Spiels gezwungen, Endzustände wurden vielmehr wie Nicht-Endzustände behandelt. Nach Sutton führt das dazu, dass die gelernte Bewertungsfunktion sich mit der Zeit verschlechtern konnte und dass nutzlose Bewertungsfunktionen gelernt werden können. Die Tatsache, dass das Gewicht

des Features Materialvorteil nicht durch das Programm geändert wurde, bewertet Sutton als Maßnahme, die die Wahrscheinlichkeit, unnütze Bewertungsfunktionen zu lernen, zu verkleinern und bewertet Samuels Verfahrens, das betragsmäßig größte Gewicht auf null zu setzen, wenn die Alpha-Komponente dreimal verloren hat, als Notwendigkeit, nutzlose Bewertungsfunktionen, die sich trotzdem noch ergeben können, zu verwerfen.

Die von Sutton gezeigte Konvergenz von TD(0) basiert auf linear unabhängigen Observation-Vektoren und in einen Experiment hatten die Vektoren der fünf Nicht-Endzustände die Länge fünf. Die Observation-Vektoren von Othilie, die sich aus den Features der Spielpositionen ergeben, erfüllen diese Anforderung nicht. Nach van den Herik u. a. (2002) gibt es bei Othello ungefähr 10^{28} beobachtbare Zustände (vgl. van den Herik u. a. (2002); zitiert nach Jin (2007)), die Konstruktion von linear unabhängigen Observation-Vektoren führte zu Vektoren der Länge $\sim 10^{28}$ und zu $\sim 10^{28}$ Gewichten. Dayan (1992) zeigt aber in seiner Arbeit über die Konvergenz von TD(λ) für allgemeine λ, dass TD-Methoden auch dann konvergieren, wenn die Observation-Vektoren nicht linear unabhängig sind.

Temporal Differenz im Zusammenhang mit neuronalen Netzen

Tesauro (1995) kombiniert in seinem Programm TD-Gammon den Temporal Difference-Ansatz mit einem neuronalen Netz, um Backgammon spielen zu lernen. Das neuronale Netz fungiert dabei als Bewertungsfunktion und Lernen findet in Form von Änderungen der Gewichte an den Verbindungen der Neuronen statt. Das Programm beobachtet eine Folge von Spielpositionen vom Anfang bis zum Ende eines Spiels. Zu jedem Zug wird aus der Spielposition ein Observation-Vektor abgeleitet und die Eingabe-Neuronen übermittelt. Die Ausgabe des neuronalen Netzes wird als Voraussage für das Ergebnis des Spiels interpretiert und die Differenz zweier Voraussagen wird in die Formel 2.11 für TD(λ) eingesetzt. In dem Fall von den vier benutzten Ausgabe-Neuronen, wird diese Formel für jedes Ausgabe-Neuron angewendet und die Summe als Gewichtsänderung benutzt.

Temporal Differenz beim Reinforcement Learning

Bei der von Wrobel u. a. (2003) definierten Lernaufgabe des Verstärkungslernens, lernt ein Agent durch Agieren in seiner Umgebung optimales Verhalten, indem er Belohnungen nach $r : S \times A \rightarrow \mathbb{R}$ wahrnimmt und verarbeitet. Zentral für optimales Verhalten ist die Nutzenfunktion U für Zustände, denn beim Bestimmen der optimalen Aktion im Zustand s geht der Nutzen des Folgezustands s' mit dem γ-fachen mit ein. Es gibt verschiedene Ansätze U zu bestimmen. Russel und Norvig (2003) beschreiben ein Reinforcement Learning, das den Belohnungsmechanismus mit einem Ansatz der temporalen Differenz verbindet. Bewegt sich der Agent aus dem Zustand s in den Zustand s', erfährt er dafür eine Belohnung nach $r(s,a)$

und aktualisiert seine Nutzenschätzung für den Zustand s, unter Benutzung der Differenz der aktuellen Nutzenschätzungen für s' und s:

$$U(s) \leftarrow U(s) + \alpha(r(s,a) + \gamma(U(s') - U(s))) \tag{2.13}$$

Die Nutzenschätzung für s wird damit in Richtung der Nutzenschätzung des Folgezustands s' angepasst.

2.3.3 Qualität des Gelernten

Nach Poole u. a. (1998), die Lernen als Fähigkeit zur Verbesserung von Verhalten definieren, gehört zu jeder Lernaufgabe ein Maß für die Verbesserung, wobei die Verbesserung nicht auf den Trainingsdaten gemessen werden soll, sondern auf neuen, unbekannten Daten. So wie auch Wrobel u. a. (2003), die die Qualität des Lernergebnisses allerdings auf Basis eines Fehlers definieren, zwischen dem Trainingsfehler und dem wahren Fehler unterschieden. Für eine Bewertungsfunktion P des Othello-Spiels ist die Bestimmung des wahren Fehlers oder eine Schätzung des wahren Fehlers nicht sinnvoll, denn beim Lernen galt schon, dass f(x) bzw. y nur dann zur Verfügung stehen, wenn das Spiel zu Ende ist. Für alle Nicht-Endpositionen kann über die Qualität von P auf diese Weise keine Aussage getroffen werden. Zum Test einer Bewertungsfunktion, die eingesetzt werden soll, um Spielzüge zu bestimmen, bietet es sich dagegen an, einen Spieler zu programmieren, der diese Bewertungsfunktion benutzt, und ihn gegen einen anderen Spieler spielen zu lassen. Da sich beim Lösen eines Multi Step Prediction Problems im Laufe des Lernverfahrens mehrere Bewertungsfunktionen $P_0, P_1, ..., P_n$ ergeben, ist es leicht eine Verbesserung im Sinne von Poole u. a. (1998) zu messen. Eine Möglichkeit ist, die Bewertungsfunktionen $P_1, P_2, ..., P_n$, die sich jeweils am Ende eines Lernspiels ergeben, gegen P_0, mit der das Lernverfahren gestartet ist, in Testspielen antreten zu lassen. Eine andere Möglichkeit ist es, die erste und die letzte Bewertungsfunktion gegen denselben Gegner antreten zu lassen.

2.4 Fazit

Wenn Othilie lernen soll, wie Othello-Spielpositionen zu bewerten seien, könnten ihr Eingabesequenzen in der Form $(x_1, e), (x_2, e), ...$ zur Verfügung gestellt werden, wobei x_z mit z = 1, 2, ... aus den Spielpositionen eines Spiels abgeleitet werden und e das Spielergebnis ist. Da Sutton (1988) dargestellt hat, dass das Lernen mit temporaler Differenz diesem Ansatz des Supervised Learnings überlegen ist, wenn die Trainingsmenge endlich ist, scheint der Ansatz mit der temporalen Differenz, bei dem die Eingabesequenzen in der Form $(x_1, x_2), (x_2, x_3), ...$ verarbeitet werden, für Othilie sinnvoller zu sein.

Wenn Othilie selbständig lernen soll, scheint der von Samuel (1959) beschriebene Selfplay-Modus, bei dem einer der beiden Spieler das Spielbaum-Suchverfahren einsetzt, um seine Züge zu bestimmen und um die spätere Spielposition $x_b = x_{z+k}$ zu bestimmen, eine geeignete Methode zu sein, um Folgen von Spielposition-Tupeln zu generieren. Damit hätten die Eingaben die Form $(x_1, x_{2+k}), (x_2, x_{3+k}), ...$

Zum Testen der gelernten Bewertungsfunktionen eignen sich Testspiele, bei den Computerspieler diese Funtionen in ihrem Spielbaum-Suchverfahren einsetzen.

3 Aufgabenstellung

3.1 Lernaufgabe und Lernverfahren von Othilie

Es ist das Ziel von Othilie eine Bewertungsfunktion für das Spiel Othello zu lernen, die in einem Spielbaum-Suchverfahren eingesetzt werden kann, um den Computer Spielzüge ermitteln zu lassen. Basierend auf den in Kapitel 2 dargestelllten Arbeiten von Wrobel u. a. (2003), Samuel (1959), Sutton (1988) und Dayan (1992) kann die Lernaufgabe und das Lernverfahren von Othilie wie folgt festgelegt werden:

Die Lernaufgabe von Othilie ist das Lösen eines Multi Step Prediction Problems: Beobachtet werden mehrere Folgen von Othello-Spielpositionen, in denen einer der beiden Spieler am Zug ist. Aus der Menge N aller möglichen Spielpositionen, bei denen das Spiel nicht zu Ende ist, lassen sich aus den Features einer Spielposition Observation-Vektoren x_z ableiten. Für die Elemente der Menge T aller möglichen Spielpositionen, bei denen das Spiel entschieden ist, lässt sich das Ergebnis e des Spiels bestimmen. Gelernt werden sollen die Gewichte w_i einer linearen Bewertungsfunktion $h(x,w) = P(x,w) = w^T x$, die jeder in Othello-Spielen möglichen Spielposition eine Voraussage auf das Spielergebnis e zuordnet. h approximiert die ideale Bewertungsfunktion f(x, w) für Othello-Spielpositionen.

Als Lernverfahren wird eine Variante des von Samuel beschriebenen Lernverfahrens benutzt: In einem Selfplay-Modus spielt eine Alpha-Komponente gegen eine Beta-Komponente mehrere Spiele gegeneinander. Nachdem die Alpha-Komponente einen Zug bestimmt hat, ändert sie in ihrer Bewertungsfunktion die Gewichte w_i ähnlich der Formel 2.12 für Gewichtsänderungen bei TD(0), wobei statt $P_{z+1} - P_z$ die Formel 2.6 für die Bestimmung von δ benutzt wird. Hat die Alpha-Komponente zehn Spiele gewonnen, bekommt die Beta-Komponente für die nächsten Spiele die aktuelle Bewertungsfunktion von der Alpha-Komponente. Am Anfang starten beide mit denselben zufälligen Startgewichten w_i für ihre Bewertungsfunktion. Der Durchlauf eines solchen Lernverfahrens wird im folgenden Text als Lernphase und ihre Spiele als Lernspiele bezeichnet.

3.2 Test der Bewertungsfunktionen

Basierend auf dem Maß der Verbesserung nach Poole u. a. (1998) in Kapitel 2.3.3, wird jeder Versuch mittels zweier verschiedener Testverfahren bewertet:

- Die anfängliche Bewertungsfunktion P_0 mit zufälligen Gewichten für die Bewertungsfunktionen und die Bewertungsfunktion P_{Ende}, die die Alpha-Komponente nach dem letzten Lernspiel hat, spielen beide gegen denselben Zufallsspieler, der seine Züge mit einem Spielbaum-Suchverfahren bestimmt. Dieser Zufallsspieler setzt aber bei jedem Zug neue zufällige Gewichte für die Bewertungen der Blätter ein. Für P_0 und P_{Ende} werden n-mal zum Beispiel 1000 Spiele durchgeführt. Der Grund dafür, dass n-mal 1000 Spiele und nicht 1-mal $n \cdot 1000$ Spiele gespielt werden, ist, dass Tests im Vorfeld ergeben haben, dass die Standardabweichung bei 1000 Spielen hoch ist. Es können also schlecht Aussagen der Form „a% der Spiele wurden gewonnen“ gemacht werden. Werden jetzt n-mal 1000 Spiele gespielt und wird aus den drei Ergebnissen der Mittelwert berechnet, kann die Aussagekraft des Mittelwertes durch die Standardabweichung beschrieben werden. Im folgenden wird dieser Test als erste Testphase und ihre Spiele als Testspiele bezeichnet.
- Jede Bewertungsfunktion P_i, die die Alpha-Komponente am Ende der Lernspiele i = 1, 2, 3, ... gelernt hat, spielt ein Spiel gegen die anfängliche Bewertungsfunktion P_0 mit den zufälligen Startgewichten. Es braucht nur ein Spiel pro Bewertungsfunktion gespielt werden, da mehrere Spiele zu demselben Spiel führen, wenn sich beide Bewertungsfunktionen nicht ändern. Dieser Test wird im folgenden Text als zweite Testphase bezeichnet und ihre Spiele als Testspiele.

3.3 Versuch und Versuchsgruppe

Die Ausführung einer Lern- und der beiden Testphasen wird als Versuch bezeichnet. Um einen Eindruck zu vermitteln, ob mehrere gleiche Versuche auf ein ähnliches Ergebnis kommen, werden in einer Versuchsgruppe mehrere gleiche Versuche ausgeführt, die sich lediglich in den Gewichten unterscheiden, mit denen die jeweiligen Lernphasen beginnen.

3.4 Features der Spielpositionen

Die Elemente der Observation-Vektoren x_z, die von Othilie beobachtet und in der Bewertungsfunktion $P(x, w)$ benutzt werden, setzen sich aus Features zusammen, die die Ausprägungen bestimmter Eigenschaften der aktuellen Spielposition beschreiben. Samuels er-

folgreiches Dame-Programm, basierte auf einer von ihm als „redundant und unvollständig“ bezeichneten Menge von Features (Samuel (1959)). Die Features, die von Othilie benutzt werden, basieren auf sehr rudimentären Eigenschaften, die sich jeder Othello-Anfänger ausdenken kann, wenn er nur die Spielregeln kennt und keine strategischen Othello-Kenntnisse hat. Alle Features werden auf Werte aus [0, 1] normalisiert.

3.4.1 Position auf dem Spielfeld

Ein Faktor für die Qualität eines Felds ist dessen Position auf dem Spielfeld: Ecken können, wenn sie von einem Spieler einmal eingenommen sind, vom Gegner unter keinen Umständen zurückerobert werden, weil es wegen ihrer besonderen Position nicht möglich ist, sie mit zwei Steinen einzuklammern. Folglich müssen die direkt an eine Ecke grenzenden Felder von geringerer Qualität sein, denn der Stein eines Spielers auf einem dieser Felder erfüllt einen Teil der Voraussetzung dafür, dass der Gegner die Ecke einnehmen kann. Entweder der Gegner hat bereits den zweiten Stein zum Einklammern oder er könnte ihn im weiteren Spielverlauf erhalten. Für Felder am Rand, die keine Ecken sind, gilt, dass sie nicht so angreifbar sind, wie die Felder in der Mitte des Spielfelds, da sie nur aus zwei Richtungen angegriffen werden können. Je nachdem an welchem Rand sich das Feld befindet, von oben und unten oder von links und rechts. Die Felder in der Mitte sind dagegen aus acht Richtungen angreifbar.

	a	b	c	d	e	f	g	h
1		C	A	B	B	A	C	
2	C	X					X	C
3	A							A
4	B							B
5	B							B
6	A							A
7	C	X					X	C
8		C	A	B	B	A	C	

(a) strategische Feldbezeichnungen
Quelle: Rose (2005)

	a	b	c	d	e	f	g	h
1	A	B	C	D	D	C	B	A
2	E	F	G	H	H	G	F	E
3	I	J	K	L	L	K	J	I
4	M	N	O	P	P	O	N	M
5	M	N	O	P	P	O	N	M
6	I	J	K	L	L	K	J	I
7	E	F	G	H	H	G	F	E
8	A	B	C	D	D	C	B	A

(b) Feldbezeichnungen bei Othilie

Abbildung 3.1: Bezeichnungen der Felder

Diese Sachverhalte könnten durch Features beschrieben werden, die sich auf die Bezeichnungen der Felder nach Abbildung 3.1a stützen, indem die Anzahl der Steine auf den entsprechenden Feldern gezählt und durch die Anzahl dieser Felder geteilt würde. Diese stra-

tegischen Felder spielen bei Othello-Strategien immerhin wichtige Rollen. Trotzdem werden für Othilie Features nach den in Abbildung 3.1b dargestellten Feldern benutzt. Der Wert eines Features ergibt sich dabei aus der Anzahl der Steine auf den jeweiligen Feldern, geteilt durch die Anzahl der Felder. Features, die auf den strategischen Feldern basieren, führten dazu, dass ein Stein auf einem X-Feld stärker zum Wert des Features beiträgt, als ein Stein auf einem C-Feld zu dessen Feature, weil es acht C-Felder aber nur vier X-Felder gibt. Von den Feldern nach 3.1b gibt es dagegen immer vier. Diese Harmonisierung der Features für einzelne Felder ist wichtig, weil bei den Gewichtsänderungen nach Formel 2.12 bzw. im Programmcode des Listings 4.2 die Werte der Features als Faktor mit eingehen. Damit ergeben sich als Features für Othilie:

Nummer:	1 bis 16
Name:	relative Anzahl der Steine auf A-, ..., P-Feldern des Spielers
Bezeichnung:	Anzahl der Steine, die der Spieler auf den A-, ..., P-Feldern hat, geteilt durch 4.
Nummer:	17 bis 32
Name:	relative Anzahl der Steine auf A-, ..., P-Feldern des Gegners
Bezeichnung:	Anzahl der Steine, die der Gegner auf den A-, ..., P-Feldern hat, geteilt durch 4.

Aus der besonderen Bedeutung der Felder am Rand des Spielfelds, lassen sich weitere Features ableiten, die in Abbildung 3.2 skizziert werden. Wenn die Felder am Rand, d.h. im äußeren Ring, eine besondere Bedeutung haben, könnten die Felder in den Ringen daneben auch eine besondere Bedeutung haben, u.s.w..

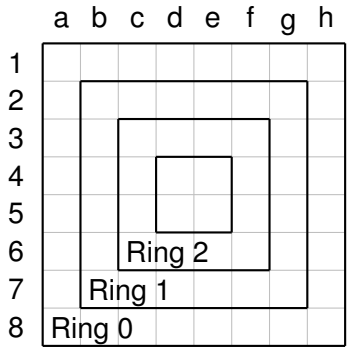


Abbildung 3.2: Bezeichnungen der Ringe

Für Othilie werden daher zusätzlich die folgenden Features abgeleitet:

Nummer:	33
Name:	relative Anzahl von Steinen des Spielers im Ring 0
Bezeichnung:	Anzahl der Steine, die der Spieler auf den Feldern des äußeren Rands hat, geteilt durch 28.
Nummer:	34
Name:	relative Anzahl von Steinen des Gegners im Ring 0
Bezeichnung:	Anzahl der Steine, die der Gegener auf den Feldern des äußeren Rands hat, geteilt durch 28.
Nummer:	35
Name:	relative Anzahl von Steinen des Spielers im Ring 1
Bezeichnung:	Anzahl der Steine, die der Spieler auf den Feldern im Ring neben dem Ring 0 hat, geteilt durch 20.
Nummer:	36
Name:	relative Anzahl von Steinen des Gegners im Ring 1
Bezeichnung:	Anzahl der Steine, die der Gegener auf den Feldern im Ring neben dem Ring 0 hat, geteilt durch 20.
Nummer:	37
Name:	relative Anzahl von Steinen des Spielers im Ring 2
Bezeichnung:	Anzahl der Steine, die der Spieler auf den Feldern im Ring innerhalb des Rings 1 hat, geteilt durch 12.
Nummer:	38
Name:	relative Anzahl von Steinen des Gegners im Ring 2
Bezeichnung:	Anzahl der Steine, die der Gegener auf den Feldern im Ring innerhalb des Rings 1 hat, geteilt durch 12.

3.4.2 Anzahl der Steine

Da sich das Ergebnis des Spiels auf die Anzahl der Steine stützt, die die beiden Spieler haben, ist es naheliegend, Features zu formulieren, die die Anzahl der Steine betrachten. Daraus werden die folgenden Features abgeleitet:

Nummer:	39
Name:	relative Anzahl der Steine des Spielers
Bezeichnung:	Anzahl der Steine, die der Spieler auf dem Spielfeld hat, geteilt durch 64.
Nummer:	40
Name:	relative Anzahl der Steine des Gegners
Bezeichnung:	Anzahl der Steine, die der Gegner auf dem Spielfeld hat, geteilt durch 64.
Nummer:	41
Name:	Spieler im Materialvorteil
Bezeichnung:	1, falls der Spieler mehr Steine als der Gegner hat - sonst 0
Nummer:	42
Name:	Gegner im Materialvorteil
Bezeichnung:	1, falls der Gegner mehr Steine als der Spieler hat - sonst 0

4 Design und Realisierung

Die Komponenten der Umsetzung der in Kapitel 3 festgelegten Aufgabenstellung mit dem an Samuel angelehnten Lernverfahren sind in Abbildung 4.1 dargestellt. Dabei sind die Bezeichnungen, die aus der Fachdomain stammen, in Klammern gesetzt; die anderen Bezeichnungen finden sich im Programmcode wieder. Othilie ist in Java implementiert.

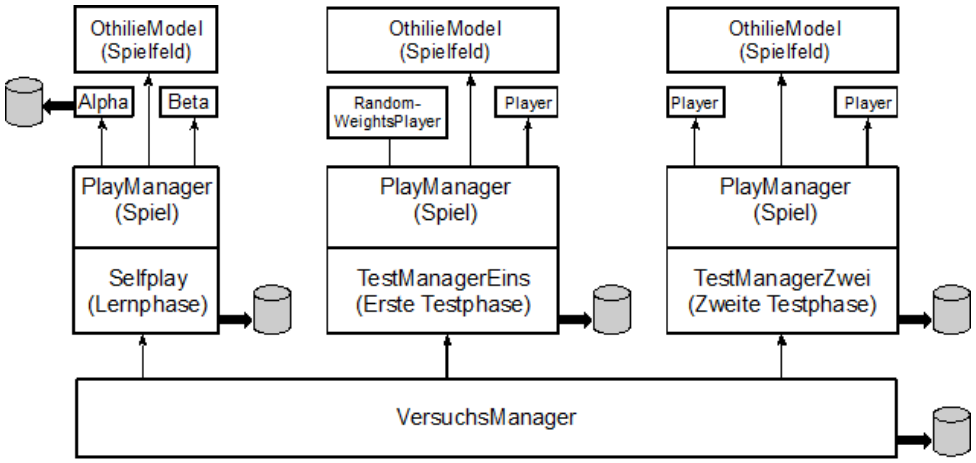


Abbildung 4.1: Komponenten des Lernverfahrens von Othilie

Der VersuchsManager hat zwei Aufgaben. Einmal ist er für die Ausführung eines Versuchs, der aus einer Lernphase und zwei Testphasen besteht, verantwortlich und außerdem für die Ausführung einer Versuchsgruppe. Wie in der Aufgabenstellung beschrieben, besteht eine Versuchsgruppe aus mehreren Versuchen, die sich lediglich in den unterschiedlichen, zufällig bestimmten Gewichten unterscheiden, mit denen die Alpha- und die Beta-Komponente die Lernphase beginnen. Da Versuche nicht häufig gestartet werden, gibt es für die Bedienung des VersuchsManagers kein grafisches User Interface, sondern nur eine main()-Methode, in die die Versuchsgruppen in Form von Methodenaufrufen eingetragen werden.

Die Komponente Selfplay realisiert die Lernphase und kontrolliert einen Teil des Lernverfahrens, indem sie die festgelegte Anzahl an Lernspielen durchführt und die Gewichte der Alpha-Komponente an die Beta-Komponente überträgt, wenn die Alpha-Komponente zehn

Spiele gewonnen hat. Die Alpha-Komponente realisiert den anderen Teil des Lernverfahrens, indem sie - wie von Samuel beschrieben - die Gewichte, die sie zur Bewertung der Spielpositionen benutzt, ändert. Die Art der Gewichtsänderung unterscheidet sich allerdings von Samuels Verfahren: Othilie benutzt Gewichtsänderungen, die eher an die von Sutton vorgestellte Formel für TD(0)-Learning angelehnt sind.

Die Komponente TestManagerEins führt die erste Testphase aus, bei der das Spielergebnis von Alphas ersten Polynom P_0 gegen einen Zufallsspieler mit dem Spielergebnis des letzten Polynoms P_{Ende} gegen denselben Zufallsspieler verglichen wird. Für den Zufallsspieler wird die Komponente RandomWeightsPlayer eingesetzt. Dieser Zufallsspieler benutzt ein Spielbaum-Suchverfahren, allerdings bei jedem Zug mit neuen, zufälligen Gewichten für die Bewertungsfunktion. Für die Spieler, die mit den Polynomen P_0 bzw. P_{Ende} bewerten, wird die Komponente Player benutzt. Bei dieser Player-Komponente handelt es sich einfach um einen Spieler, der am Anfang Gewichte bekommt und mit diesen Gewichten alle Spielpositionen bewertet.

Die Komponente TestManagerZwei setzt die zweite Testphase um, bei der die Spieler, die mit den Polynomen P_1 bis P_{Ende}, jeweils ein Spiel gegen das Polynom P_0 spielen. Alle Spieler, die in dieser zweiten Testphase benutzt werden, werden mit der Player-Komponente realisiert, die schon in der ersten Testphase eingesetzt wurde.

Für die Spiele aller drei Phasen, aus denen ein Versuch besteht, wird eine Komponente benötigt, die das aktuelle Spielfeld verwaltet. Dies wird von der OthilieModel-Komponente übernommen. Indem sie Spielzüge entgegennimmt, gegnerische Steine daraufhin umdreht und Funktionalitäten zur Verfügung stellt, mit den geprüft werden kann, welche Züge in einer Spielposition gültig sind, oder prüft, ob ein Spieler überhaupt einen gültigen Zug hat, setzt sie einen Teil der Othello-Spielregeln um. Für das Lernverfahren kann das OthilieModel aus der repräsentierten Spielposition ein Oberservation-Vektor x ableiten und die repräsentierte Spielposition bewerten.

Die Sub-Komponente Spiel ist Teil aller drei Phasen. Es handelt sich bei dieser Komponente um den Koordinator eines regelkonformen Spielablaufs: Er nutzt Methoden des OthilieModels, um zu entscheiden, welcher der beiden Spieler am Zug ist oder ob das Spiel beendet ist. Außerdem nutzt er die Methoden der entsprechendem Spieler, um deren Spielzüge zu erfragen und führt diese Spielzüge am OthilieModel aus. Damit setzen die Sub-Komponente Spiel und das OthilieModel zusammen alle Othello-Spielregeln um.

Die Aufgabe des Reportings ist, den Verlauf der Gewichte darzustellen, die von der Alpha-Komponente im Verlauf der Lernphase gelernt werden, und das Ergebnis der beiden Testphasen festzuhalten. Da Othilie mit dem Ziel programmiert wurde, das Ergebnis eines Lernverfahrens in einer schriftlichen Arbeit zu dokumentieren, erzeugt das Reporting einfache Ausgabedateien. Es gibt also keine interaktive Komponente, die zum Beispiel Log-Dateien ausliest, um das Ergebnis der Versuche darzustellen. An den Stellen, an denen ein Verlauf

dargestellt werden soll, wie beispielsweise beim Verlauf der Gewichte während der Lernphase, erzeugt das Reporting Daten- und Steuerdateien, die von gnuplot dargestellt werden können. Die Stellen, an denen Ausgaben für die Auswertung erzeugt werden, sind in Abbildung 4.1 durch die Datenbank-Symbole gekennzeichnet. Das Datenbank-Symbol an der Alpha-Komponente deutet an, dass einige Daten, die sich in dieser Komponente bei der Bestimmung der Gewichtsänderungen ergeben, ebenfalls erhoben werden. Dabei handelt es sich um die in Kapitel 4.4 beschriebenen Werte, wie zum Beispiel den Fehlerwert δ. Diese Fehlerwerte werden für ein Spiel akkumuliert und fließen als Summe in das Reporting der Lernphase ein. Als weiteres Reporting der Lernphase nimmt der VersuchsManager die Datendateien aus den Lernphasen, die zu einer Versuchsgruppe gehören, und stellt sie in der sogenannten Per Feature-Darstellung neu zusammen. Das Ziel der Per Feature-Darstellung ist, grfisch darzustellen, ob die Gewichte eines Features in mehreren Versuchen am Ende der Lernphase ähnliche Werte haben.

In der Abbildung 4.1 nicht dargestellt sind die Datenbankzugriffe. Im Grunde sind die Daten in der Datenbank zu den Daten des für eine Auswertung nutzbaren Reportings redundant und werden zum Teil nicht weiter genutzt[6]. An einigen Stellen sind sie dennoch nützlich: Für die beiden Testphasen werden alle Gewichte der Alpha-Komponente aus der Lernphase benötigt und es ist einfacher, diese Werte aus der Datenbank zu lesen, als sie aus den Datendateien herauszufiltern. Auf das automatische Erzeugen der Datendateien soll aber auch nicht verzichtet werden.

Auch wenn eine grafische Oberfläche, mit der ein Benutzer ein Othello-Spiel gegen den Computer spielen kann, nicht zum Lernverfahren oder dessen Tests beiträgt[7], ist für Othilie so eine grafische Oberfläche implementiert. Immerhin war die Überlegung, wie ein Computerspieler für Othello realisiert werden kann, die Ausgangsbasis für die Idee eines Programms, das die Bewertungsfunktion für ein Spielbaum-Suchverfahren lernt.

4.1 Spielfeld

Das Spielfeld bzw. die Spielposition eines Othello-Spiels wird durch die Klasse OthilieModel repräsentiert. Intern speichert es die aktuelle Spielposition, d.h. es wird für jedes Feld gespeichert, ob und welcher Spieler einen Stein auf dem Feld hat. Damit von außen auf diese Zustände Bezug genommen werden kann, definiert das OthilieModel die Konstanten LEER, WEISS und SCHWARZ. Die Konstanten WEISS und SCHWARZ werden außerdem benutzt,

[6] Ungenutzt sind beispielsweise die einzelnen Züge, die die beiden Spieler im Rahmen der Lernphase durchführen.

[7] Der Zufallsspieler ist gegenüber einer Reihe von menschlichen Spielern bevorzugt worden, weil er - da er beispielsweise nicht unter Ermüdungserscheinungen oder Motivationsproblemen leidet - als zuverlässiger bewertet worden ist und weil mit dem Zufallsspieler eine größere Anzahl von Testspielen leicht möglich ist.

um die beiden Spieler zu bezeichnen. Die Spielfelder der im Rahmen dieser Arbeit erzeugten OthilieModel-Objekte haben alle eine Seitenlänge von acht Feldern.

Die Details hinter der Frage, ob einer der Spieler einen Stein auf ein bestimmtes Feld setzen kann oder welche gegnerischen Steine bei einem Zug umgedreht werden, werden vom Model verborgen und über die Methoden isMovePossible(int, int, int) und setZustand(int, int, int) zur Verfügung gestellt:

isMovePossible(zeile:int, spalte:int, spieler:int):boolean
Mit dieser isMovePossible()-Methode kann geprüft werden, ob der durch spieler beschriebene Spieler nach den Spielregeln, auf das durch zeile und spalte beschriebene Feld einen Stein setzen kann, d.h. sie prüft, ob das Feld leer ist und ob ein Zug auf dieses Feld dazu führte, dass mindestens ein gegnerischer Stein umgedreht wird. Die Voraussetzung beim Aufruf dieser Methode ist, dass die Werte ziele und spalte ein Feld beschreiben, das auch existiert. Stammt der Wert für den Spieler nicht aus der Menge {WEISS, SCHWARZ}, liefert die Methode false.

setZustand(zeile:int, spalte:int, zustand:int):boolean
Mit der setZustand()-Methode kann ein Spielzug am OthilieModel ausgeführt werden, wobei zeile und spalte das Feld beschreiben und zustand die Farbe, die gesetzt wird. Das Model wird daraufhin den Stein setzen und die gegnerischen Steine nach den in Kapitel 2.1 beschriebenen Spielregeln umdrehen. Voraussetzung für den korrekten Ablauf dieser Methode ist, dass die Methode isMovePossible(int, int, int) vorher für dieses Feld und für diesen Spieler true ergeben hat.

Im Zustand eines Spielfelds sind Informationen enthalten, mit denen ein Spiel koordiniert und das Spielergebnis bestimmt werden kann. Um diese Informationen zur Verfügung zu stellen, implementiert das OthilieModel folgende Methoden:

isMovePossible(spieler:int):boolean
Diese isMovePossible()-Methode prüft, ob der durch spieler beschriebene Spieler irgendeinen gültigen Zug hat. Übergibt diese Methode false und ist das Spiel noch nicht beendet, muss der Spieler aussetzen. Gültige Werte für spieler sind WEISS und SCHWARZ.

isFertig()
Diese Methode übergibt true, wenn das Spiel beendet ist, d.h. keiner der beiden Spieler einen gültigen Zug hat.

getWinner():int
Mittels dieser Methode kann das Spielergebnis ermittelt werden. Zurückgegeben wird ein Wert aus der Menge {WEISS, SCHWARZ, UNENTSCHIEDEN}, je nachdem, welcher der

beiden Spieler mehr Steine auf dem Spielfeld hat, oder ob beide Spieler die gleiche Anzahl an Steinen haben.

Um eine Spielposition bewerten zu können, sind im OthilieModel die in Kapitel 3.4 beschriebnen Features implementiert, allerdings als private Methoden. Zum Umgang mit diesen Features und zum Bewerten von Spielpositionen stellt das OthilieModell zwei Methoden zur Verfügung:

getFeatures(spieler:int):double[]
Die getFeatures()-Methode übergibt für die aktuelle Spielposition ein Array mit den Werten der einzelnen Features. Dieses Array entspricht dem Observation-Vektor. Mit dem Parameter spieler wird angegeben, für welchen Spieler die Werte bestimmt werden sollen, wobei Werte aus {WEISS, SCHWARZ} gültig sind. Wird z.B. WEISS übergeben, ist WEISS der Spieler und SCHWARZ der Gegner, im Sinne der Features-Beschreibung aus Kapitel 3.4. Die Werte der einzelnen Features im Array liegen alle im Intervall [0, 1]. Benutzt wird diese Methode nicht nur lokal in der Methode score(), sondern auch bei der Bestimmung der Gewichtsänderung des LearningPlayers in Kapitel 4.2.1. Denn bei der Bestimmung der Gewichtsänderung w_i geht der Wert eines Features x_i als Faktor mit ein.

score(gewichte:double[], spieler:int):double
Die Methode score() berechnet die Bewertung der aktuellen Spielposition. Sie wird im Rahmen des Spielbaum-Suchverfahrens benutzt, um die Blätter des unvollständigen Suchbaums zu bewerten und bei der Bestimmung der Gewichtsänderungen des LearningPlayers, der in Kapitel 4.2.1 beschrieben wird.
Die Gewichte w_i werden von außen durch ein Array übergeben. Die Länge dieses Arrays muss mit der Anzahl der Features übereinstimmen. Es ist außerdem notwendig, dass sich die Reihenfolge der Gewichte w_i und die Reihenfolge der Features x_i entsprechen. Der Grund, dass die Gewichte von außen übergeben werden, ist, dass eine Spielposition durch zwei Spieler unterschiedlich bewertet werden kann. Daher sind die Gewichte Attribute der Spieler (vgl. Kapitel 4.2).
Die Angabe des Spielers ist wie bei der Methode getFeatures() wichtig, damit festgelegt ist, für welchen Spieler die Bewertung vorgenommen werden soll. Der Wert muss aus der Menge {WEISS, SCHWARZ} sein.
Die Methode score() gibt für den Endzustand „Spieler hat verloren“ den Wert -1 und für den Endzustand „Spieler hat gewonnen“ den Wert 1 zurück. Für den Endzustand „Das Spiel ist unentschieden“ wird der Wert 0 zurückgegeben, der genau zwischen den beiden anderen Werten liegt. Da alle Features Werte aus dem Intervall [0, 1] annehmen, ist sichergestellt, dass der Rückgabewert von score() genau dann Werte zwischen -1 und 1 annimmt, wenn alle übergebenen Gewichte im Intervall [-1, 1] liegen. Damit wäre dann sichergestellt, dass die Bewertung einer Spielposition, in der das Spiel nicht beendet ist, außerhalb des Intervalls [-1, 1] liegt.

Der Pseudocode in Listing 4.1 stellt die Arbeitsweise der Bewertungsfunktion dar, die im OthilieModel implementiert ist. Da die Festlegung auf der Werte für die Endzustände nicht zwingend so erfolgen muss, werden im Listing die Werte min für -1, max für 1 und zero für 0 benutzt.

```
1  score(gewichteVektor, spieler){
2          if(isSpielende(){
3                  case getWinner()==spieler:
4                          return max;
5                  case getWinner()==getOpponent(spieler):
6                          return min;
7                  case getWinner()==UNENTSCHIEDEN:
8                          return zero;
9          }else{
10                 featuresVektor = getFeatures(spieler);
11                 gSumme = gewichteteSumme(gewichteVektor, featuresVektor);
12                 result = Normalisierung von gSumme auf [min, max];
13                 return result;
14         }
15 }
```

Listing 4.1: Bewertungsfunktion des OthilieModels

4.2 Spieler

Ein Objekt der Klasse Player oder ein Objekt einer Unterklasse von Player repräsentiert in dieser Arbeit einen Spieler. Neben dem Speichern eines Arrays mit Gewichten, mit den ein Spieler Spielpositionen bewertet, gehört das Bestimmen eines Zugs unter Benutzung eines Spielbaum-Suchverfahrens zu den Kernfunktionalitäten eines Spielers. Ein Spieler-Objekt kann die Rolle beider Spieler einnehmen, d.h. er kann Schwarz sein und das Spiel beginnen oder er kann die Rolle von Weiß einnehmen. Die Rolle, die ein Player-Objekt hat, wird im Rahmen der Zugbestimmung beim Bewerten der Spielposition mit der Methode OthilieModel:score() wichtig: Wie oben beschrieben, muss dieser Methode die Rolle als Parameter übergeben werden.

4.2.1 Gewichte bei den verschiedenen Spielern

Für diese Arbeit wurden eine Reihe von Spieler-Klassen implementiert, denen gemeinsam ist, dass sie ein Array mit Gewichten für die Bewertung speichern. Die Klassen unterscheiden sich in der Art, wie ihre Gewichte geändert werden.

Player

Ein Objekt der Klasse Player erhält über seinen Konstruktor einen Satz von Gewichten und spielt bzw. bewertet mit diesen Gewichten, solange es existiert.

RandomWeightPlayer

Ein RandomWeightPlayer-Objekt generiert sich vor jedem Zug neue zufällige Gewichte, um sie im folgenden Spielbaum-Suchverfahren zur Bewertung einzusetzen. Dieser Spieler wird als Zufallsspieler in der ersten Testphase eingesetzt, bei dem die anfängliche Bewertungsfunktion P_0 und die letzte Bewertungsfunktion P_{Ende} gegen einen Zufallsspieler spielen.

ImprovingPlayer

Ein Objekt der Klasse ImprovingPlayer realisiert die von Samuel beschriebene Beta-Komponente, die im Verlauf der Lernphase zu gewissen Zeitpunkten neue Gewichte von der Alpha-Komponente erhält. Die Gewichte, mit denen die Beta-Komponente startet, werden dem Konstruktor übergeben und zum Ändern der Gewichte steht die Methode setGewichte(gewichte:double[]) zur Verfügung.

LearningPlayer

Ein Objekt der Klasse LearningPlayer realisiert die von Samuel beschriebene Alpha-Komponente, die während der Lernphase die Gewichte w_i ändert.
Für den LearningPlayer übernommen wurde Samuels Idee, eine Differenz $\delta = P(x_b) - P(x_{z-1})$ als Fehler der Bewertungsfunktion zu berechnen, um die Gewichte entsprechend zu ändern.
Wenn bei einem Othello-Spiel beim Zug z die beiden Spielpositionen verglichen werden, die sich mit x_b und x_{z-1}beschreiben lassen, können bei x_b ein oder mehrere Steine mehr auf dem Spielfeld liegen. Das unterscheidet Othello von Spielen, bei dem die Spielfiguren auf dem Spielfeld bewegt werden; bei Othello ist mit jedem Zug ein Stein mehr auf dem Spielfeld. Um zu verhindern, dass beim Bestimmen der Differenz zweier Spielpositionen, die Tendenz nur aus dem Grund dazu geht, dass δ positiv ist, weil mehr Steine auf dem Spielfeld liegen, bestimmt der LearningPlayer in Zeile 14 des Listings 4.2 nicht die Differenz zweier Bewertungen, sondern bildet sozusagen relative Bewertungen, indem er die beiden Bewertungen durch die Anzahl der Steine teilt, die jeweils auf dem Spielfeld liegen, und bestimmt dann die Differenz dieser relativen Bewertungen.

Da Samuels Aussagen über die Gewichtsänderungen, für die Konstruktion des LearningPlayers grundlegend sind, werden sie in Tabelle 4.1 noch einmal wiederholt.

Fall	**Schlussfolgerung**
$\delta > 0$	mehr Gewicht für positiv beitragende Terme t_i
	weniger Gewicht für negativ beitragende Terme t_i
$\delta < 0$	weniger Gewicht für positiv beitragende Terme t_i
	mehr Gewicht für negativ beitragende Terme t_i

Tabelle 4.1: Samuels Aussagen über die Gewichtsänderungen

In der Zeile 26 ist der Pseudocode dargestellt, der beim LearningPlayer die Gewichtsänderungen bestimmt. Anders als bei Sutton (1988) werden die Gewichtsänderungen nicht gesammelt und bei Spielende durchgeführt, sondern sofort, wenn die Gewichtsänderungen vom LearningPlayer bestimmt werden. Das Vorzeichen der Gewichtsänderung wird durch das Vorzeichen von δ bestimmt, wenn features[i], lernrate und differenzierungsfaktor positiv sind. Samuels in Tabelle 4.1 dargestellten Aussagen über die Art der Gewichtsänderungen sind damit umgesetzt, denn im ersten Fall werden die Gewichte vergrößert und im zweiten Fall werden sie verkleinert. Da auch der Betrag von δ in die Gewichtsänderung eingeht, bestimmt δ auch das Ausmaß der Gewichtsänderung mit. Je größer der Fehler desto größer die Gewichtsänderung und bei einem Fehler von null gibt es keine Gewichtsänderung.
Die Variable lernfaktor entspricht dem α beim Lernen mit temporaler Differenz, der bestimmt, zu welchem Bruchteil der Fehler δ in die Gewichtsänderungen eingeht. Der Wert ist für alle Gewichtsänderungen, die der LearningPlayer berechnet, konstant und wird über den Konstruktor des Spielers festgelegt.
Beim Lernen mit temporaler Differenz mit einer Bewertungsfunktion der Form $P(x, w) = w^T x$ geht der Wert x_i in die Gewichtsänderung von w_i als Faktor mit ein. Beim LearningPlayer entspricht dieser Wert dem Wert features[i].
Ein weiterer Faktor, der in die Gewichtsänderung mit eingeht, ist der Differenzierungsfaktor, der dem LearningPlayer im Konstruktor übergeben wird. Die Idee hinter diesem Faktor ist, dass Gewichtsänderungen in Richtung der x-Achse - also betragsmäßigen Verkleinerungen - mehr Gewicht gegeben werden kann, als betragsmäßigen Vergrößerungen. Gewichte, die mal vergrößert und genauso häufig verkleinert werden, sollen durch diesen Faktor dazu tendieren, einen Wert nahe Null anzunehmen und nur die Gewichte, die deutlich oder häufig vergrößert bzw. verkleinert werden, erhalten große Beträge. Auf diese Weise soll erreicht werden, dass Features, die keine deutliche Rolle bei der Bewertung einer Spielposition spielen, kleine Gewichte erhalten. Diesen Differenzierungsfaktor hat es weder bei Samuels Dame-Programm noch bei theoretischen Ausführungen zum Lernen mit temporaler Differenz gegeben.

In Listing 4.2 ist die Arbeitsweise des LearningPlayers als Pseudocode dargestellt. In Zeile 11 wird, wie von Samuel beschrieben, die gespeicherte Spielposition des letzten Zugs des LearningPlayers mit den inzwischen veränderten Gewichten neu bewertet. Mit dem Aufruf der Methode alphaBetaSearch() wird das Spielbaum-Suchverfahren gestartet.

```
1  getNextMove(ply, othilieModel){
2
3          ergebnis = alphaBetaSearch(ply, OthilieModel, player);
4          zug = ergebnis.getSpielzug();
5          score = ergebnis.getScore();
6          anzZuege = ergebnis.getAnzahlZuege();
7
8          if(pastBoardIsStored){
9
10                 oldModel = getStoredBoard();
11                 oldScore = oldModel.score(gewichte, player);
12                 oldAnzZuege = oldModel.getAnzahlZuege();
13
14                 delta = ((score/anzZuege)-(oldScore/oldAnzZuege));
15
16                 featuresVektor = getFeatures(player);
17                 for i in {1, 2, ..., gewichte.length}{
18
19                         diffFak = 1.0;
20                         if((gewichte[i] >= 0.0 and delta < 0.0)
21                                 || (gewichte[i] < 0.0 and delta >= 0.0)){
22                                 diffFak = differnenzierungsfaktor;
23                         }
24
25                         gewichte[i] += lernrate * delta * featuresVektor[i] * diffFak;
26                 }
27
28                 Normalisierung aller Gewichte auf [-1.0, 1.0];
29
30         }
31
32
33         kopie = OthilieModel.clone();
34         kopie.setZustand(zug.getZeile(), zug.getSpalte(), player);
35         saveBoard(kopie);
36
37         return zug;
38 }
```

Listing 4.2: Bestimmung eines Zugs beim LearningPlayer

In Zeile 31 normalisiert der LearningPlayer alle Gewichte auf einen Bereich von -1 bis 1. Damit wird erreicht, dass alle Bewertungen von Spielpositionen aus [-1, 1] stammen, denn - wie in Kapitel 4.1 beschrieben - sind Gewichte zwischen -1 und 1 die Vorraussetzung für Bewertungen im Intervall [-1, 1].

Durch diese Realisierung bekommt das Lernverfahren von Othilie Eingaben der Form

$$E = \{((stored(x_a), x_{a+k}), (stored(x_b), x_{b+k}), ...) | a, b, ... \; Zug \; der \; Alpha - Komponente\} \tag{4.1}$$

wobei der Ausdruck stored(x) die Spielposition, die bei der letzten Zugbestimmung der Alpha-Komponente gespeichert wurde, beschreibt und k die Tiefe des Suchbaums ist. Ab dem Zeitpunkt, zu dem der 60. Stein auf das Spielfeld gelegt wird, werden die Bewertungen der späteren Spielpositionen - wie von Sutton gefordert - auf das Spielergebnis e gezwungen. Im Unterschied zu den Eingabesequenzen in Kapitel 2.4 werden Spielpositionen ab dem dritten Zug verarbeitet, d.h. eine Eingabesequenz hat die Form $(x_3, x_{4+k}), (x_4, x_{5+k}), ...$ - vorausgesetzt, die Alpha-Komponente setzt nicht aus. Setzt die Alpha-Komponente aus, hat das zur Folge, dass nicht alle aufeinanderfolgenden Spielpositionen eines Spiels von dem Lernverfahren betrachtet werden und dass nach dem Aussetzen, bei der nächsten Zugbestimmung der Alpha-Komponente nicht die Spielposition vor der aktuellen, sondern eine weiter zurückliegende Spielposition für die temporale Differenz benutzt wird.

4.2.2 Spielbaum-Suchverfahren aller Spieler

Die Klasse Player ist die Superklasse aller Spieler-Implementierungen, so dass die von ihr implementierte Methode getNextMove() allen Spielern zur Verfügung steht, um in einer gegebenen Spielposition mittels eines Spielbaum-Suchverfahrens einen Zug zu bestimmen.

getNextMove(int:ply, OthilieModel:model, int:player):OthilieSpielzug
Die getNextMove()-Methode der Klase Player startet das Spielbaum-Suchverfahren, wobei eine Alpha-Beta-Suche benutzt wird. Der Methode wird durch den Parameter ply übergeben, bis zu welcher Tiefe der unvollständige Spielbaum aufgebaut werden soll, wenn das Spielende nicht vorher erreicht ist. Die aktuelle Spielsituation, für die der Zug bestimmt werden soll, wird durch das OthilieModel übergeben. Der Rückgabewert dieser Methode ist vom Typ OthilieSpielzug, der im Wesentlichen die Zeile und die Spalte des Spielzugs kapselt.

Klassen und Methoden der Spielbaum-Suche

Zu einem Spielbaum-Suchverfahren gehören nach Reinfeld drei Komponenten:

- Die Erzeugung der Nachfolgerknoten (Zugerzeugung),
- die Bewertung der Blätter des Suchbaums und
- die Baumsuche

wobei die ersten beiden Aspekte vom jeweiligen Spiel abhängig sind, während die Baumsuche vom Spiel unanhängig ist (vgl. Reinefeld (2006)). Bei der Implementierung für Othilie ist die vom konkreten Spiel unabhängige Baumsuche in der abstrakten Klasse AbstractSuchknoten untergebracht. Die von der Baumsuche benutzten, vom konkreten Spiel abhängigen Komponenten Zugerzeugung und Bewertung der Blätter sind in dieser Klasse als abstrakte Methoden deklariert, die erst in der konkreten Implementierung OthilieSuchknoten implementiert werden.

Die Klasse AbstractSuchknoten implementiert die öffentliche Methode alphaBetaSearch(). Wird sie am Wurzelknoten aufgerufen, führt sie die Spielbaum-Suche durch. Wie bei allen Suchverfahren für Zweipersonen-Nullsummenspiele, rufen sich daraufhin die privaten Methoden maxAlphaBeta() und minAlphaBeta() wechselseitig rekursiv auf, bis der Spielbaum die gewünschte Tiefe erreicht hat und übergeben dann das Blatt bis zur Wurzel zurück, das unter der Annahme, dass beide Spieler optimal spielen, das beste ist, dass aus der aktuellen Spielposition zu erreichen ist.
Für den Code der Alpha-Beta-Suche gibt es zahlreiche Quellen und er soll an dieser Stelle nicht wiederholt werden. Die Quelle für die Implementierung von Othilie ist von Luck (2004). Konkret umgesetzt ist nicht das Verfahren, bei dem die Bewertung des bestmöglichen Blatts zurückgegeben wird, sondern das Verfahren, bei dem das bestmögliche Blatt zurückgegeben

wird. Jedesmal, wenn ein Blatt nach oben gegeben wird, wird in das Blatt der Spielzug eingetragen, der zu dem Unterbaum führt, aus dem das Blatt stammt. Damit wird erreicht, dass das Ergebnis-Blatt, das am Ende der Alpha-Beta-Suche vom Wurzelknoten zurückgegeben wird, den bestmöglichen Spielzug als Ergebnis der Suche enthält.

Die konkrete Klasse OthilieSuchknoten erbt das Suchverfahren von der Klasse Abstract-Suchknoten und stellt einen Knoten im Suchbaum eines Othello-Spiels dar. Jeder Knoten enthält ein OthilieModel, um die Spielposition zu repräsentieren, für die der Knoten steht. Die implementierten Methoden, der vom konkreten Spiel abhängigen Komponenten zur Zugerzeugung und zur Bewertung, sind getNachfolger() und score(). Die spielEnde()-Methode ist auch vom konkreten Spiel abhängig.

getNachfolger():List
Die Methode getNachfolger() liefert eine Liste von Nachfolgerknoten, die durch gültige Züge entstehen, die im aktuellen Knoten möglich sind. Damit in der letzten Ebene des unvollständigen Suchbaums keine weiteren Nachfolgerknoten expandiert werden, ist es erforderlich, dass der restPly eines Knotens, der die Anzahl der Ebenen enthält, die einem Knoten noch folgen, für einen Nachfolgerknoten um den Wert 1 dekrementiert wird.
Der Fall, dass einer der beiden Spieler aussetzen muss, muss bei der Implementierung der Methode getNachfolger() besonders beachtet werden. Wichtig ist, dass in diesem Fall, die Methode nicht einfach eine leere Liste erzeugt, sonst wäre die Suche in diesem Knoten beendet, obwohl das Spiel weiter geht. Stattdessen wird ein Suchknoten als Nachfolger erzeugt, der eine Kopie des OthilieModels im aktuellen Knoten enthält. Der Grund für das Einfügen dieser Kopie ist, dass die wechselseitigen Aufrufe der Funktionen minAlphaBeta() und maxAlphaBeta() andernfalls falsche Ergebnisse lieferten. In Abbildung 4.2 ist die Notwendigkeit dieser Kopie exemplarisch dargestellt. Enthalten sind drei Pfade von der Wurzel A bis zum bestmöglichen Blatt E. Die beiden Spieler, die jeweils am Zug sind, werden in der Abbildung nicht als Weiß oder Schwarz bezeichnet, sondern als MIN und MAX. Dies soll verdeutlichen, welche der beiden Methoden aufgerufen werden sollte.

Der obere Pfad zeigt die Aufrufe der beiden Methoden für den Fall, dass keiner der beiden Spieler aussetzt. Im Knoten D, in dem der MIN-Spieler am Zug ist, wird die Methode minAlphaBeta() aufgerufen. Im mittleren Pfad sind die Aufrufe für den Fall dargestellt, dass der Spieler MIN aussetzen muss und kein Knoten als Kopie eingefügt wird. Durch das Aussetzen und weil die beiden Methoden abwechselnd aufgerufen werden, wird im Knoten D die Methode maxAlphaBeta() aufgerufen, obwohl der MIN-Spieler am Zug ist und das ist falsch. Der untere Pfad stellt den implementierten Fall dar. Der MAX-Spieler setzt aus, aber ein Knoten B' wird als Kopie von B eingefügt. Das erwünschte Ergebnis ist, dass im Knoten D, in dem der MIN-Spieler am Zug ist, auch die Methode minAlphaBeta() aufgerufen wird. Anzumerken ist dazu noch, dass die Tatsache, dass in der Kopie B' für den Zug (der zu diesem Knoten bzw. diesem Unterbaum führt) der Wert null gespeichert wird, keine Auswirkungen

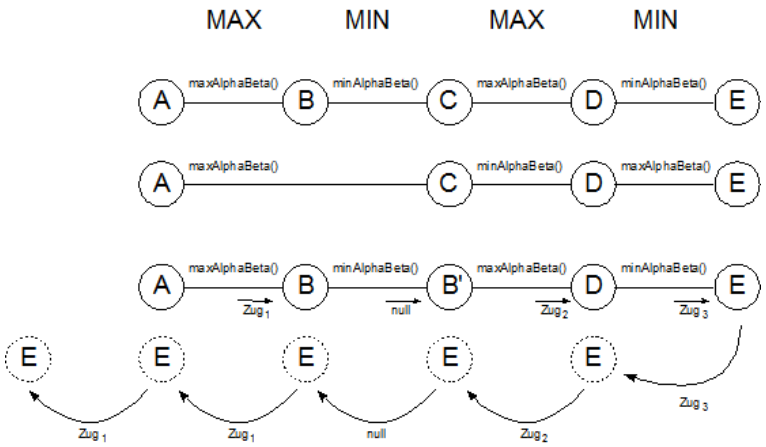


Abbildung 4.2: Der Fall Aussetzen bei der Zugerzeugung

auf das Ergebnis der Alpha-Beta-Suche hat. Zwar wird, wenn das bestmögliche Blatt zur Wurzel übertragen wird, in jedem Knoten der Spielzug, der zu diesem Knoten geführt hat in das Blatt übertragen; also auch der Spielzug null. Allerdings ist sichergestellt, dass der Spielzug null in der Ebene darüber durch einen gültigen Zug überschrieben wird, weil immer so eine Ebene mit gültigem Zug existiert. Existierte sie nicht, würde es daran liegen, dass der MAX-Spieler in der Wurzel A keinen gültigen Zug hat. Da die Alpha-Beta-Suche nur dann gestartet wird, wenn der MAX-Spieler am Anfang auch einen gültigen Zug hat, ist es ausgeschlossen, dass die Spielbaum-Suche den ungültigen Zug null liefert.

In Listing 4.3 ist die Methode getNachfolger() als Pseudocode dargestellt. Dem Konstruktor für den Nachfolgerknoten werden u.a. zwei Spieler übergeben. Der Wert des ersten Spielers ist im ganzen Suchbaum unverändert. Er gibt an, aus der Sicht welchen Spielers die Blätter bewertet werden sollen. Der zweite Spieler wechselt von Ebene zu Ebene. Es handelt sich dabei um den Spieler, der im jeweiligen Knoten am Zug ist.

score():double
Die Bewertung eines Knotens wird an das OthilieModel, das die Spielposition speichert, delegiert. Listing 4.3 stellt den Programmcode der Bewertungsmethode dar. In der Variablen gewichte sind die Gewichte gespeichert, mit den bewertet werden soll. In der Variablen bewertungsSpieler ist der Spieler gespeichert, aus dessen Sicht die Bewertung vorgenommen werden soll.

spielEnde():boolean
Die Frage, ob ein Spiel beendet ist, wird an die Methode OthilieModel:isFertig() delegiert.

```
1  getNachfolger(){
2
3          result = {};
4
5          for zeile in {1, 2, ..., model.groesse(){
6                  for spalte in {1, 2, ..., model.groesse(){
7
8                          if(model.isMovePossible(zeile, spalte, spieler)){
9
10                                 clonedModel = model.clone();
11                                 clonedModel.setZustand(zeile, spalte, spieler);
12                                 nachfolger = new OthilieSuchknoten(bewertungsSpieler,
13                                             getOpponent(spieler),
14                                             clonedModel,
15                                             new Spielzug(zeile, spalte),
16                                             restPly - 1);
17                                 result.add(nachfolger);
18                         }
19                 }
20         }
21
22         if(result == {}){
23                 ClonedModel = model.clone();
24                 nachfolger = new OthilieSuchknoten(bewertungsSpieler,
25                                         getOpponent(spieler),
26                                         clonedModel,
27                                         null,
28                                         restPly - 1);
29                 result.add(nachfolger)
30         }
31 }
32
33
34 score(){
35         model.score(gewichteVektor, bewertungsSpieler);
36 }
37
38 spielEnde(){
39         return model.isFertig();
40 }
```

Listing 4.3: Methoden der konkreten Klasse OthilieSuchknoten

JUnit-Test des Spielbaum-Suchverfahrens

Zum Testen der Implementierung der Alpha-Beta-Suche wurde die konkrete Klasse StreichholzSuchknoten programmiert. Beim Streichholz-Spiel liegt eine bestimmte Anzahl n Streichhölzer auf dem Tisch und zwei Spieler nehmen abwechselnd ein oder zwei Streichhölzer weg. Derjenige, der das letzte oder die letzten beiden Streichhölzer wegnimmt, hat verloren. Dieses Spiel eignet sich deshalb, weil bei n = 4, 7, 10, 13, 16, ... der Spieler, der anfängt unabhängig von seinen Spielzügen verliert, wenn der Gegner optimal spielt und weil der Spielbaum bei n=16 vollständig aufgebaut werden kann. Für jeden möglichen Spielverlauf wurde für den Spieler, der anfängt, eine Liste mit Spielzügen generiert. Der zweite Spieler

benutzt die Alpha-Beta-Suche mit dem vollständigen Spielbaum. Am Ende jeden Spiels wurde durch eine assertTrue()-Anweisung geprüft, ob der zweite Spieler auch der Gewinner war. Der Test ist für n = 4, 7, 10, 13, 16 in allen Fällen erfolgreich durchgelaufen. Auch wenn dass kein Beweis für die Korrektheit der Spielbaum-Suche ist, wäre es umgekehrt im Fall, dass der zweite Spieler einmal nicht der Gewinner gewesen wäre, ein deutlicher Hinweis darauf gewesen, dass die Spielbaum-Suche falsch implementiert ist.

In Abbildung 4.3 sind die wesentlichen Komponenten der Spielbaum-Suchverfahrenen von Othilie in einem UML-Diagramm dargestellt.

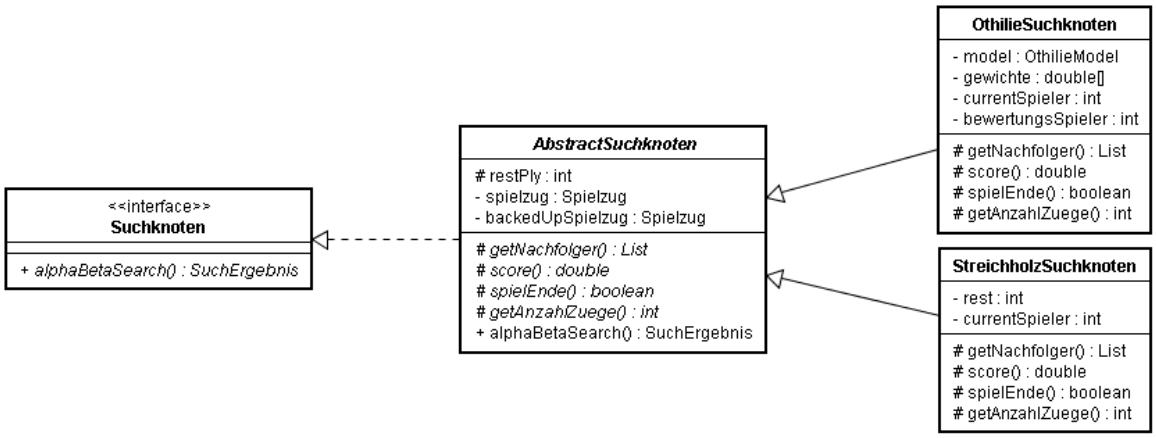


Abbildung 4.3: UML-Diagramm zu den Spielbaum-Suchverfahren

4.3 Versuch

Ein Versuch besteht wie in der Aufgabenstellung in Kapitel 3 und in Abbildung 4.1 dargestellt aus einer Lernphase, der zwei Testphasen folgen, wobei allen Phasen gemeinsam ist, dass Othello-Spiele durchgeführt werden. Damit die Funktionalität Othello-Spiel in allen drei Phasen zur Verfügung steht, ist sie in der Klasse PlayManager implementiert, die die Superklasse der drei Klassen ist, die die Versuchsphasen implementieren.

4.3.1 Durchführung eines Spiels

Zum Spielen eines Othello-Spiels bekommt ein PlayManager-Objekt mit dem Konstruktor zwei Objekte von Typ Player übergeben, so dass alle in Kapitel 4.2 beschriebenen Player-Klassen mittels des PlayManagers gegeneinander spielen können. Um das Spiel dann zu starten, wird die Methode playOneGame() aufgerufen:

playOneGame()
Die playOneGame()-Methode implementiert den regelkonformen Ablauf eines Spiels:
Am Anfang wird ein neues OthilieModel erzeugt, das die anfängliche Spielposition repräsentiert. Im Verlauf des Spiels werden die Züge der beiden Spieler mittels der Methode setZustand() am Model durchgeführt, so dass es immer die aktuelle Spielposition enthält.
Der PlayManger benutzt die Methode getNextMove() der beiden Player-Objekte, um ihre Spielzüge zu erfragen, wobei er sich darauf verlässt, dass die Player-Objekte gültige Züge erzeugen.
Um das Spiel zu steuern, d.h. um den richtigen Spieler aufzurufen, solange das Spiel nicht beendet ist, werden die Methoden isFertig() und isMovePossible(spieler:int) des OthilieModels benutzt.
Außerdem wird in der Methode playOneGame() festgelegt, bis zu welcher Tiefe der Suchbaum maximal aufgebaut wird, da den Player-Objekten beim Aufruf der Methode getNextMove() übergeben wird, mit welcher Suchbaum-Tiefe der Zug bestimmt werden soll. Für diese Arbeit ist die Tiefe für beide Spieler auf fünf festgelegt.

In Abbildung 4.4 ist in einem UML-Diagramm dargestellt, wie die Klassen zusammenarbeiten, um ein Othello-Spiel mit zwei „Computer-Spielern“ zu realisieren.

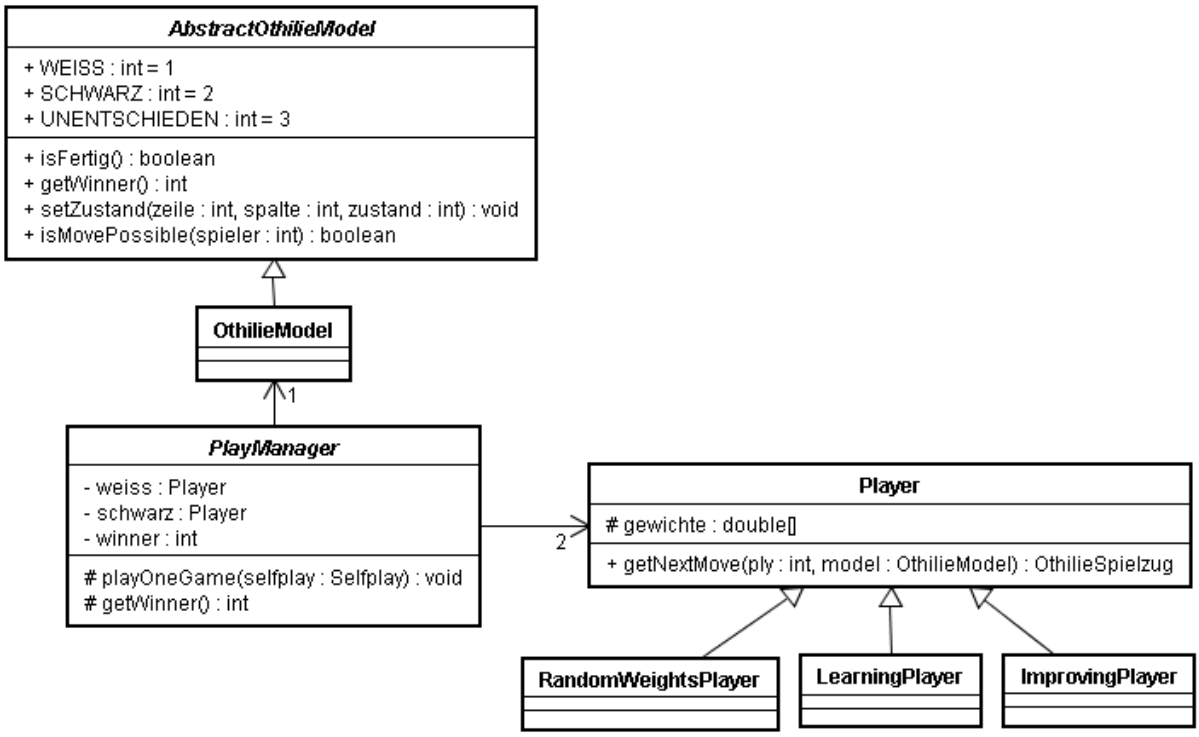


Abbildung 4.4: UML-Diagramm zum PlayManager, der ein regelkonformes Spiel realisiert.

getWinner():int
Das Ergebnis eines Spiels wird durch die Methode getWinner() vom PlayManager zur Verfügung gestellt. Ihr Ergebnis ist der Ergebniswert der Methode getWinner() des OthilieModels und stammt aus der Menge {WEISS, SCHWARZ, UNENTSCHIEDEN}. Die Voraussetzung für den Aufruf dieser Methode ist, dass das Spiel beendet ist, d.h. die Methode playOneGame() durchgelaufen ist.

4.3.2 Lernphase

Der von Samuel beschriebene Selfplay-Modus einer Alpha-Komponente gegen eine Beta-Komponente stellt die Lernphase dar. Sie erzeugt zwei Player-Objekte. Einmal ein LearningPlayer-Objekt, das die von Samuel beschriebene Alpha-Komponente darstellt, die versucht, Gewichte für eine Bewertungsfunktion zu lernen, und außerdem ein ImprovingPlayer-Objekt, das die Rolle von Samuels Beta-Komponente übernimmt und immer dann die aktuellen Gewichte von Alpha bekommt, wenn Alpha zehnmal gewonnen hat.
Es ist im SelfplayManager festgelegt, dass die lernende Alpha-Komponente die Rolle des Spielers Weiß bekommt und dass die Beta-Komponente die Rolle von Schwarz übernimmt und daher die Spiele beginnt. Für diese Festlegung waren folgende Überlegungen entscheidend: Bei einer grafischen Oberfläche, mit der ein menschlicher Spieler gegen einen Computerspieler spielen kann, ist es naheliegend, dass der menschliche Spieler beginnt und der Computerspieler in der Rolle von Weiß spielt. Es scheint sinnvoll, den lernenden Spieler in der Rolle spielen zu lassen, in der die gelernte Bewertungsfunktion eingesetzt werden soll.

Zum Steuern der Lernphase werden folgende Parameter benutzt:

- Mit dem Parameter anzahlSpiele wird angegeben, aus wie viel Lernspielen die Lernphase besteht.
- Durch den Parameter sameStartPolynom wird festgelegt, ob die Alpha- und die Beta-Komponente die Lernphase mit denselben zufälligen Gewichten starten. Wird hier false übergeben, werden für die beiden Spieler eigene zufällige Gewichte bestimmt.
- Der Wert differenzierungsfaktor wird der lernenden Alpha-Komponente als Parameter übergeben. Es handelt sich dabei um den in Abschnitt 4.2.1 beschriebenen Parameter, der an den LearningPlayer weitergegeben wird, um die Art der Gewichtsänderungen des lernenden Spielers zu beeinflussen.
- Der Wert lernfaktor ist ebenfalls ein Wert, der die Gewichtsänderungen des LearningPlayers beeinflusst und wird diesem als Parameter übergeben (vgl. Kapitel 4.2.1).

playManyGamesSamuelLearning(anzahlSpiele:int,
sameStartPolynom:boolean,
differenzierungsfaktor:double,
lernfaktor:double)

Das Listing 4.4 zeigt die essentiellen Aspekte der Lernphase in Form von Pseudocode.
Am Anfang der Lernphase müssen die beiden Player-Objekte erzeugt werden. Zum Erzeugen eines Arrays mit zufälligen Werten aus dem Intervall $[-1, 1]$ wird die Methode generateRandomWeights() benutzt. Es ist damit sichergestellt, dass die Länge des Arrays mit den Gewichten mit der Länge des Feature-Arrays, das in der Bewertungsfunktion score() benutzt

wird, übereinstimmt. Wenn die beiden Player-Objekte erzeugt sind, wird innerhalb der Schleife ein Lernspiel durchgeführt und ggf. die aktuellen Gewichte der Alpha-Komponente an die Beta-Komponente übertragen.

```
1  playManyGamesSamuelLearnng(anzahlLernspiele, sameStartPolynom, lernfaktor, differenzierungsfaktor){
2
3          alphaPolynom = new Polynom(generateRandomWeigthts());
4
5          if(sameStartPolynom){
6                  betaPolynom = alphaPolynom;
7          }else{
8                  betaPolynom = new Polynom(generateRandomWeigthts());
9          }
10
11         learningPlayer = new LearningPlayer(WEISS, alphaPolynom, lernfaktor, differenzierungsfaktor);
12         improvingPlayer = new ImprovingPlayer(SCHWARZ, betaPolynom);
13
14         counter = 0;
15         for i in {1, 2, ..., anzahlLernspiele}{
16
17                 ein Lernspiel durchführen mit learningPlayer und improvingPlayer;
18
19                 if(getWinner() == WEISS){
20                         counter++;
21                 }
22
23                 if(counter == 10){
24                         counter = 0;
25
26                         improvingPlayer.setGewichte(learningPlayer.getGewichte());
27                 }
28         }
29 }
```

Listing 4.4: Programmcode der Lernphase

4.3.3 Erste Testphase

Das Ziel der ersten Testphase ist, die relativen Gewinnhäufigkeiten der beiden Polynome P_0 und P_{Ende} gegen den Zufallsspieler zu vergleichen. Wie in der Aufgabenstellung in Kapitel 3 beschrieben, werden die Testspiele in n Duchrchläufen mit jeweils 1000 Testspielen durchgeführt, damit die Standardabweichung als Maß für die Aussagekraft der Gewinnhäufigkeit berechnet werden kann. Für diese Testphase stellt die Klasse TestManagerEins zwei Methoden zur Verfügung. Die Methode playOneDurchlauf() wird benutzt, um die 1000 Spiele eines Durchlaufs durchzuführen und mit der Methode playNDurchläufe() werden die n Durchläufe gestartet. Der Pseudocode beider Methoden ist in Listing 4.5 dargestellt. Dass die Spieler, die die beiden gelernten Polynome einsetzten, in der Rolle von Weiß spielen liegt daran, dass ein Computerspieler in einer grafischen Oberfläche in der Rolle Weiß spielen sollte und dass die Alpha-Komponente die Rolle Weiß hat.

```
1  playOneDurchlauf(anzahlTestspiele, versuchsBez, lfdNummer){
2
3          alphaPolynom = getAlphaPolynomFromDatabase(versuchsBez, lfdNummer);
4          spielerWeiss = new Player(WEISS, alphaPolynom);
5          spielerSchwarz = new RandomWeightsPlayer(SCHWARZ);
6
7          counter = 0;
8          for i in {1, 2, ..., anzahlTestspiele}{
9
10                 ein Testspiel durchführen mit spielerWeiss und spielerSchwarz;
11
12                 if(getWinner() == WEISS){
13                         counter++;
14                 }
15         }
16
17         return counter/anzahlTestSpiele;
18 }
19
20 playNDurchlaeufe(anzahlDurchlaeufe, anzahlTestspiele, versuchsBez, lfdNummer){
21
22         for i in {1, 2, ..., anzahlDurchlaeufe}{
23
24                 gewinne[i] = playOneDurchlauf(anzahlTestspiele, versuchsBez, lfdNummer);
25         }
26
27         mittelwert = mittelwert(gewinne);
28         standardabweichung = standardabweichung(gewinne);
29 }
```

Listing 4.5: Methoden zum Realisieren der ersten Testphase

4.3.4 Zweite Testphase

Für die zweite Testphase, in der die Polynome P_1 bis P_{Ende} aus der Lernphase jeweils ein Spiel gegen das Polynom P_0 spielen, implementiert die Klasse TestManagerZwei die Methode playSequenceFrom1ToEnd(). Dass die Spieler, die die gelernten Polynome einsetzten, in der Rolle von Weiß spielen liegt wie bei der ersten Testphase daran, dass ein Computerspieler in einer grafischen Oberfläche in der Rolle Weiß spielen sollte und dass die Alpha-Komponente die Rolle Weiß hat.

playSequenceFrom1ToEnd(String versuchsBez, int letztesSpiel)
Die Methode erzeugt anfangs für den Spieler Schwarz ein Objekt vom Typ Player, das mit den Gewichten des Polynoms P_0 initialisiert wird. Dieser Spieler wird in der gesamten zweiten Testphase benutzt. Innerhalb einer Schleife werden für den Spieler Weiß Player-Objekte erzeugt, die mit den Gewichten der Polynome P_1 bis P_{Ende} initialisiert werden, um jeweils ein Spiel gegen den Spieler Schwarz zu spielen.

In Listing 4.6 stellt den Pseudocode der Methode playSequenceFrom1ToEnd() dar.

```
playSequenceFrom1ToEnd(versuchsBez, anzahlLernsiele){

        polySchwarz = getAlphaPolynomFromDatabase(versuchsBez, 0);
        playerSchwarz = new Player(SCHWARZ, polySchwarz);

        counterWeis = 0;
        counterSchwarz = 0;
        counterUnentschieden = 0;
        for i in {1, 2, ..., anzahlLernsiele}{

                polyWeiss = getAlphaPolynomFromDatabase(versuchsBez, i);
                playerWeiss = new Player(WEISS, polyWeiss);

                ein Testspiel durchführen mit spielerWeiss und spielerSchwarz;

                if(getWinner() == WEISS){
                        counterWeiss++;
                }else if(getWinner() == SCHWARZ){
                        counterSchwarz++;
                }else{
                        counterUnentschieden++;
                }
        }
}
```

Listing 4.6: Methoden zum Realisieren der zweiten Testphase

4.3.5 Starten von Versuchen und Versuchsgruppen

Wie am Anfang des Kapitels 4 beschrieben, ist das Starten eines Versuchs eine der Aufgaben des VersuchsManagers. Dabei geht es darum, die Methoden der einzelnen Phasen mit den richtigen Parametern aufzurufen. Beispielsweise muss die zweite Testphase aus m Testspielen bestehen, wenn die Lernphase aus m Lernspielen besteht, oder es müssen die Ausgaben des Reportings, das - wie in Abbildung 4.1 dargestellt - über die Architektur verteilt ist, in einer sinnvollen Verzeichnisstruktur angeordnet werden.

Die zweite Aufgabe des VersuchsManagers ist, eine Versuchsgruppe zu starten. Eine Versuchsgruppe ist - wie in Kapitel 3 beschrieben - eine Gruppe von Versuchen mit denselben Parametern, die sich lediglich in den unterschiedlichen, zufällig bestimmten Gewichten unterscheiden, mit denen die Alpha-Komponente die Lernphase beginnt. Der VersuchsManager startet daher hintereinander mehrere Versuche mit denselben Parametern.

4.4 Reporting

Am Anfang dieses Kapitels wurde beschrieben, dass das Reporting in erster Linie Ausgabedateien erzeugt. Abbildung 4.1 hat verdeutlicht, dass es keine zentrale Reporting-

Komponente gibt, vielmehr erzeugt jede Phase eines Versuchs eigene Ausgabedateien im Dateisystem.

An den Stellen, an denen Datendateien für Gnuplot erzeugt werden, werden gleichzeitig die nötigen Steuerdateien erzeugt. Diese Steuerdateien dienen eher als Kopiervorlage, die nicht ohne Überarbeitung benutzt werden können. Es macht beispielsweise - wegen der unterschiedlichen Bildbereiche - keinen Sinn relative und absolute Gewinnhäufigkeit in einer Grafik darzustellen, trotzdem werden Steuerdateien erzeugt, die Einträge für beide Verläufe enthalten. Durch einfaches Überarbeiten, können dann daraus Steuerdateien erzeugt werden, die für die Auswertung in Kapitel 5 nützlich sind.

4.4.1 Reporting der Lernphase

Während der Lernphase wird eine Datendatei erzeugt, in die Daten einfließen, die beim LearningPlayer und beim SelfplayManager anfallen. Sie enthält für jedes Lernspiel

- die Gewichte des LerningPlayers am Ende des Spiels,
- einige statistische Informationen wie Gewinnhäufigkeiten und
- Werte, die Gewichtsänderungen innerhalb eines Lernspiels beschreiben.

In Detail handelt es sich bei den Werten zu den Gewichtsänderungen um

- die Summe der Fehlerwerte delta,
- den größten Fehlerwert delta,
- die Summe der Beträge aller Gewichtsänderungen und
- den größten Betrag aller Gewichtsänderungen.

Der Versuchsmanager, der für die Ausführung einer Versuchsgruppe verantwortlich ist, nimmt die m Datendateien der m Lernphasen und stellt die Werte in der Per Feature-Darstellung neu zusammen: Für jedes Feature erzeugt er eine Datendatei mit den Verläufen aus den m Versuchen. Das Ziel dabei ist, einen Eindruck zu vermitteln, ob die Gewichte eines Features in verschiedenen Versuchen gegen denselben Wert streben.

4.4.2 Reporting der ersten Testphase

Das Ziel des Reportings der ersten Testphase ist, die relativen Gewinnhäufigkeiten der beiden Spieler, die mit den Bewertungsfunktionen P_0 bzw. P_{Ende} bewerten, festzuhalten. Für jeden dieser beiden Polynom-Spieler wird eine einfache Textdatei erstellt, die für jeden der

drei Testdurchläufe die relative Gewinnhäufigkeiten und am Ende deren Mittelwert und ihre Standardabweichung enthält.

4.4.3 Reporting der zweiten Testphase

Für die zweite Testphase soll dargestellt werden, wie sich die Gewinnhäufigkeit entwickelt, wenn die Spieler, die mit den Bewertungsfunktionen P_1 bis P_{Ende} bewerten, jeweils ein Testspiel gegen den Spieler spielen, der mit der Bewertungsfunktion P_0 bewertet. Dazu erzeugt das Reporting eine Datendatei mit einer Zeile für jedes der Testspiele, die die folgenden Werte enthält:

- absolute Gewinnhäufigkeit der Spieler mit P_1 bis P_{Ende},
- relative Gewinnhäufigkeit der Spieler mit P_1 bis P_{Ende},
- absolute Gewinnhäufigkeit des Spielers mit P_0,
- relative Gewinnhäufigkeit des Spielers mit P_0,
- absolute Anzahl der Testspiele, die unentschieden endeten und
- relative Anzahl der Testspiele, die unentschieden endeten.

4.5 Grafische Oberfläche zum Spielen eines Spiels

Beim Spielen eines Othello-Spiels mit der grafischen Oberfläche spielt der Benutzer in der Rolle von Schwarz, d.h. der Benutzer beginnt das Spiel, und der Computerspieler spielt in der Rolle von Weiß. Die Spielsteine der Oberfläche haben aber nicht diese Farben; Weiß spielt mit gelben Steinen und Schwarz spielt mit blauen Steinen.

Der Computerspieler wird von einem Player-Objekt realisiert. Die Bewertungsfunktion, die dieser Spieler für die Spielbaum-Suche benutzt, wird von dem Benutzer vor dem eigentlichen Spiel in einer Combobox ausgewählt. Abbildung 4.5 zeigt das Fenster mit dieser Combobox. Die Bewertungsfunktionen in dieser Combobox werden beim Programmstart aus Dateien im Filesystem gelesen. Es gibt ein kleines Programm, das diese Dateien im entsprechenden Format und an entsprechender Stelle erzeugt, indem es die letzte Bewertungsfunktion P_{Ende} der Lernphasen aller in Kapitel 5 und Kapitel 6 beschriebenen Versuche aus der Datenbank ausliest.

Zur Realisierung der grafischen Oberfläche wurde das OthilieModel um die Funktionalität erweitert, Observer-Objekte zu informieren, wenn sich der Zustand im Model ändert. Dies

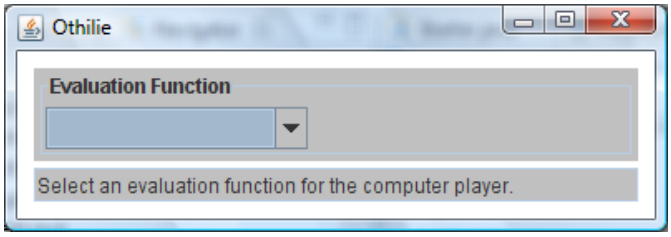


Abbildung 4.5: Auswahl einer Bewertungsfunktion

ermöglicht, das OthilieModel im Rahmen des von Cooper (1998) beschriebenen Observer-Patterns als Subject für eine Spielfeld-Repräsentation zu benutzen, bei dem Observer die grafische Darstellung aktualisieren, wenn sie durch das Subject benachrichtigt werden. Dabei wird zum Benachrichtigen nur das Observer-Interface SpielfeldListener der Observer-Objekte benutzt; die Observer-Objekte kennt das Subject nicht.

Abbildung 4.6 stellt die grafische Oberfläche dar. Sie enthält zwei Observer-Objekte: Die grafische Darstellung des Spielfelds und die Spielstand-Anzeige. Beide sind bei dem Othilie-Model, an dem die Spielzüge ausgeführt werden, als Observer angemeldet.

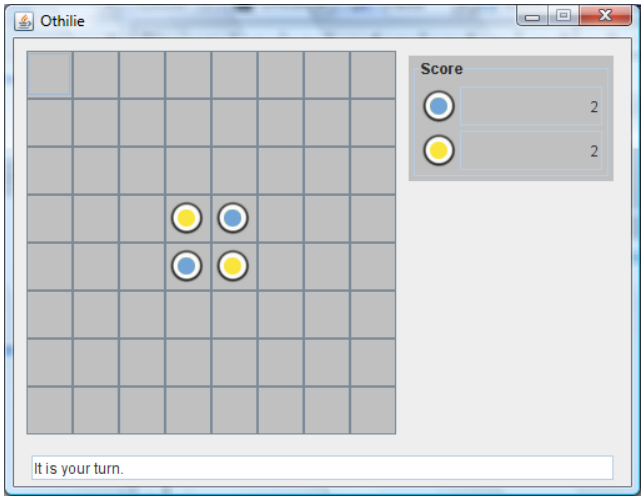


Abbildung 4.6: Grafische Oberfläche zum Spielen

Die grafische Darstellung des Spielfelds und die zum Spielen eines Spiels nötige Controler-Funktionalität sind in dem Observer SpielfeldPanel platziert: Die grafische Darstellung des Spielfelds ist durch 64 Buttons realisiert, die wie leere Felder oder wie Felder mit gelben oder blauen Stein aussehen können. Der Benutzer kann - vorausgesetzt er ist am Zug und es gibt Felder, auf die ein gültiger Zug möglich ist - auf eines dieser Felder setzen, indem er mit der Maus darauf klickt. Um dies zu ermöglichen, ist der Observer zusätzlich als ActionListener

implementiert und bei allen Buttons als ActionListener registriert. Durch das Setzen eines Steins durch den Benutzer, wird das OthilieModel durch den Observer geändert und mittels des Player-Objekts ein Zug des Computers generiert, der ebenfalls am OthilieModel ausgeführt wird. Das OthilieModel informiert daraufhin den Observer, dass sich der Zustand geändert hat und dass die grafische Darstellung bestimmter Buttons aktualisiert werden muss.

Die Spielstand-Anzeige auf der rechten Seite wird durch ein SpielstandPanel realisiert. Es implementiert ebenfalls das Observer-Interface SpielfeldListener und ist bei dem OthilieModel, an dem die Spielzüge ausgeführt werden, als Observer registriert. Wird nun das SpielstandPanel über Änderungen im OthilieModel informiert, erfragt es beim OthilieModel die Anzahl der Steine, die die beiden Spieler haben, und stellt die Antwort als Spielstand dar.

Um den Benutzer beim Spielen zu unterstützen, hat der Mauszeiger über den Feldern, auf die er setzen kann, die Form eines Spielsteins und über Feldern, auf die der Benutzer nicht setzen kann, die Form eines Kreuzes. Dazu implementiert das SpielfeldPanel zusätzlich das MouseListener-Interface und ist bei den Buttons, die die Felder darstellen, als MouseListener registriert. Kommt der Mauszeiger nun in den Bereich eines bestimmten Felds, wird vom OthilieModel mit der isMovePossible()-Methode erfragt, ob der Spieler einen Stein auf dieses Feld setzen kann und der Mauszeiger wird entsprechend gesetzt.

Anhang A enthält ein Klassendiagramm, in dem dargestellt ist, wie das Observer-Pattern in Othilies grafischer Oberfläche umgesetzt ist.

4.6 Zusammenfassung ausgewählter Designentscheidungen

Dieser Abschnitt fasst die wesentlichen Design-Entscheidungen zusammen, die wichtig sind, weil sie entscheidenen Einfluss auf das Lernverfahren und auf die beiden Testphasen haben.

Obwohl theoretisch größere Spielfelder möglich sind, beträgt die Seitenlänge des Spielfelds acht Felder. Dieser Wert entspricht der klassischen Spielfeldgröße bei Othello-Spielen und soll auch für Othilie gelten.

Spielpositionen, in denen das Spiel zu Ende ist, werden mit Werten aus {-1, 0, 1} bewertet. Für die anderen Spielpositionen bildet die Bewertungsfunktion des OthilieModels aus x und w die auf [-1, 1] normierte gewichtete Summe, d.h. sie bildet $w_1x_1 + w_2x_2 + ... + w_nx_n$ und teilt diesen Wert durch n. Da die Werte der Features im Intervall [0, 1] liegen, die Gewichte Werte aus dem Intervall [-1, 1] an nehmen und weil nicht alle Features gleichzeitig den Feature-Maximalwert 1 annehmen können ist sichergestellt, dass Bewertungen für Spielpositionen,

in denen das Spiel nicht zu Ende ist, nicht größer als der Bewertungs-Maximalwert 1 und nicht kleiner als der Bewertungs-Minimalwert -1 sind.

Für die Bestimmung des Fehlerwertes δ wird nicht die Bewertung des OthilieModels benutzt, sondern relative Bewertungen, d.h. beide Bewertungen werden durch die Anzahl der Steine geteilt, die jeweils auf dem Spielfeld liegen.

In die Gewichtsänderungen geht ein Differenzierungsfaktor mit ein, durch die Gewichtsänderungen in Richtung der x-Achse verstärkt werden können.

Durch die Realisierung des Teil des Lernverfahrens, der - an Samuels Beschreibungen angelehnt - im LearningPlayer untergebracht ist werden von Othilie Eingaben der Form $(x_3, x_{4+k}), (x_4, x_{5+k}), ...$ verarbeitet, wenn die Alpha-Komponente nicht aussetzt. Die Fälle, in denen die Alpha-Komponente aussetzt, werden von Othilie nicht besonders behandelt. So wird in diesen Fällen vernachlässigt, dass einige Spielpositionen nicht betrachtet werden und dass sich der Abstand zwischen den beiden Spielpositionen für die temporale Differenz vergrößert.

Im PlayManager, der für beide Spieler eines Spiels das Spielbaum-Suchverfahren startet, ist festgelegt, dass die Tiefe des Suchbaums fünf beträgt. Da der PlayManager die Super-Klasse aller Klassen ist, die Lern- und Testspiele realisieren, ist damit die Suchbaum-Tiefe für alle Spiele, die von Othilie durchgeführt werden, auf fünf festgelegt.

Bei einer grafischen Oberfläche zu Spielen eines Othello-Spiels ist es naheliegend, dass der menschliche Spieler das Spiel beginnt, also in der Rolle von Schwarz spielt, und dass der Computerspieler in der Rolle von Weiß spielt. Daher ist festgelegt worden, dass die Alpha-Komponente in der Rolle von Weiß spielt, weil dann die gelernte Bewertungsfunktion in der Rolle gelernt wird, in der sie eingesetzt werden soll. Aus ähnlichen Gesichtspunkten wurde festgelegt, dass bei den beiden Arten von Testspielen die Spieler, die die gelernten Bewertungsfunktionen einsetzen in der Rolle von Weiß spielen.

Der Zufallsspieler der ersten Testphase wählt nicht aus der Menge der gültigen Züge zufällig einen Zug aus, sondern benutzt ein Spielbaum-Suchverfahren, allerdings mit neuen zufälligen Gewichten für jeden Zug.

Die Parameter mit denen eine Versuchsgruppe gesteuert werden kann, sind:

- Anzahl der Versuche, aus denen die Versuchsgruppe besteht
- Anzahl der Lernspiele in der Lernphase eines Versuchs
- Anzahl der Durchläufe und Anzahl der Testspiele pro Durchlauf in der ersten Testphase
- Wert des Lernfaktors α, der die Gewichtsänderungen bei der Alpha-Komponente beeinflusst

- Wert des Differenzierungsfaktors γ, der ebenfalls die Gewichtsänderungen der Alpha-Komponente beeinflusst
- Ein boolscher Wert, mit dem bestimmt werden kann, ob die Alpha- und die Beta-Komponenten die Lernphase mit denselben zufälligen Gewichten beginnen, oder ob für beide Spieler eigene zufällige Werte bestimmt werden.

5 Versuchsergebnisse

Bei der Bewertung eines Lernverfahrens, das eine Bewertungsfunktion lernt, die im Rahmen eines Spielbaum-Suchverfahrens eingesetzt werden kann, interessieren zwei Aspekte:

- Wie spielstark ist das Suchverfahren, das diese Bewertungsfunktion benutzt?
- Kommt das Lernverfahren bei gleichen Paramentern auf dasselbe oder ein ähnliches Ergebnis, wenn es mehrmals gestartet wird? Dabei handelt es sich um die Frage, ob die Gewichte der einzelnen Features in den drei Versuchen gegen dieselben Werte konvergiert.

Für die Fragestellung nach der Spielstärke werden die Ergebnisse der ersten und zweiten Testphase herangezogen. Einen visuellen Eindruck zur zweiten Fragestellung vermitteln die Graphen, die der VersuchsManager aus den Datendateien der drei Lernphasen generiert; der Per Feature-Darstellungen. Zusätzlich dazu werden in diesem Kapitel zwei Kennzahlen betrachtet, die die Ähnlichkeit der Ergebnisse messen sollen:

Wenn sich die Polynome von drei Versuche einer Versuchsgruppe am Ende eines Lernspiels spiel durch $P_{1,spiel}$, $P_{2,spiel}$ und $P_{3,spiel}$ mit

$$P_{versuch,spiel}(x,w) = \sum_{i=0}^{41} w_{i_{versuch,spiel}} x_i \tag{5.1}$$

darstellen lassen und $\sigma_{i_{spiel}}$ die Standardabweichnung der drei Gewichte $w_{i_{1,spiel}}$, $w_{i_{2,spiel}}$ und $w_{i_{3,spiel}}$ ist, dann kann die Summe dieser Standardabweichungen über alle Gewichte mit

$$Q_{spiel} = \sum_{i=0}^{41} \sigma_{i_{spiel}} \tag{5.2}$$

als Maß für die Ähnlichkeit der drei Polynome nach dem Spiel spiel benutzt werden.

Als Maß für die Konvergenz des Lernverfahrens wird als weitere Kennzahl Q^*_{spiel} betrachtet. Dabei werden die Q_{spiel} der letzten Spiele der Lernphasen aufsummiert, d.h. es wird

$$Q^*_{spiel} = \sum_{s=spiel}^{6500} Q_{spiel} \tag{5.3}$$

berechnet, um einen Eindruck zu bekommen, ob sich die Polynome ab dem Lernspiel spiel bis zum Ende der Lernphasen ähnlich sind.

5.1 Überblick über die Parameter der Versuchsgruppen

In diesem Kapitel werden die Ergebnisse von fünf Versuchsgruppen beschrieben, die jeweils aus drei Versuchen bestehen. Wie in Kapitel 3.3 dargestellt, besteht eine Versuchsgruppe aus mehreren Versuchen, um festzustellen, ob das Lernverfahren mehrmals bei unterschiedlichen Startgewichten auf ein ähnliches Ergebnis kommt. Der Wert drei wurde bestimmt durch folgende Überlegungen: Es muss mehrere Versuche geben und es sollen wenig Versuche sein, um die Laufzeit zu begrenzen. Der Wert drei ist ein kleiner Wert, der ausreichend erscheint.

Die Lernphase, mit der jeder Versuch beginnt, besteht aus 6500 Lernspielen, in denen die Alpha-Komponente durch Gewichtsänderungen auf Basis der Differenz zweier Spielpositionen versucht, eine spielstarke Bewertungsfunktion für Othello-Spielpositionen zu lernen. Der Wert 6500 ist motiviert durch die Beobachtung bei der Implementierung, dass 6500 Lernspiele ausreichen können, um Gewichte zu erhalten die ihren Wert nur noch wenig ändern. Die Alpha- und die Beta-Komponente beginnen die Lernphase mit denselben, zufällig bestimmten Startgewichten.

Es werden in der ersten Testphase zum Testen der ersten und der letzten Bewertungsfunktion aus der Lernphase drei Durchläufe mit jeweils 1000 Testspielen benutzt, wobei der Zufallsspieler, gegen den die beiden Bewertungsfunktionen spielen, ein Spielbaum-Suchverfahren benutzt; allerdings mit neuen zufälligen Gewichten für jeden Zug. Die erste Bewertungsfunktion aus der Lernphase wird als Startpolynom P_0 und die letzte als Ergebnispolynom P_{6500} bezeichnet. In jedem Durchlauf wird die relative Gewinnhäufigkeit des Spielers gemessen, der die Polynome P_0 bzw. P_{6500} in ihrem Spielbaum-Suchverfahren einsetzten. Für diese drei Werte werden Mittelwert und Standardabweichung bestimmt. Dieser Mittelwert wird als Spielstärke oder als mittlere Spielstärke bezeichnet. Dem Wert 1000 liegt die Beobachtung zugrunde, dass 1000 Testspiele gegen den Zufallsspieler ausreichen um Spielstärken auf fünf Prozent genau zu ermitteln, wobei auch hier ein möglichst kleiner Wert gewählt wurde, um die Laufzeit eines Versuchs zu begrenzen. Die erste Testphase besteht damit aus $2 \cdot 3 \cdot 1000$ = 6000 Testpielen.

Da die Lernphase aus 6500 Lernspielen besteht, spielen in der zweiten Testphase die Polynome P_1 und P_{6500} aus der Lernphase je ein Testspiel gegen das Polynom P_0. Daraus ergibt sich, dass die zweite Testphase aus 6500 Testspielen besteht.

Die Versuchsgruppen A, B und C beschreiben den Versuch, die Ähnlichkeit der Ergebnisse durch den Lernfaktor α = 1,0, 0,1, 0,05 zu erreichen. Der Differenzierungsfaktor ist bei diesen Versuchsgruppen γ = 1,0, so dass die Gewichtsänderungen denen von Sutton in Formel 2.12 möglichst nahe kommen.

Die Versuchsgruppe D wurde mit dem Lernfaktor α = 0,01 und dem Differenzierungsfaktor γ = 1,0 gestartet. Sie gehört ebenfalls zu dem Versuch, die Ähnlichkeit der Ergebnisse durch den Lernfaktor zu erreichen. Die Ergebnisse dieser Versuchsgruppe sind sich in keiner Weise ähnlich geworden und der Eindruck ist, dass der Lernfaktor für 6500 Lernspiele zu klein ist.

Die Versuchsgruppe E ist die einzige Versuchsgruppe mit einem Differenzierungsfaktor ungleich 1,0; er beträgt γ = 1,5 und der Lernfaktor ist α = 0,1. In Bezug auf die Ähnlichkeit der Ergebnisse liefert diese Versuchsgruppe die besten Ergebnisse. Der Wert 1,5 basiert auf der Erfahrung bei der Implementierung, dass er ähnliche Ergebnisse erzeugt. Der Wert $\alpha = 0,1$ wurde gewählt, weil der Wert 1,0 nach den Ergebnissen der Versuchsgruppe A zu groß erschien und weil die Werte 0,05 und 0,01 nach den Ergebnissen der Versuchsgruppen C und D als zu klein erschien.

Die Tabelle 5.1 enthält die Parameter der fünf vorgestellten Versuchsgruppen.

	Versuchsgruppe A	Versuchsgruppe B	Versuchsgruppe C
Versuche	3	3	3
Lernspiele	6500	6500	6500
Lernfaktor α	1,0	0,1	0.05
Differenzierungsfaktor γ	1,0	1,0	1,0
Testdurchläufe	3	3	3
Testspiele pro Durchlauf	1000	1000	1000
	Versuchsgruppe D	Versuchsgruppe E	
Versuche	3	3	
Lernspiele	6500	6500	
Lernfaktor α	0,01	0,1	
Differenzierungsfaktor γ	1,0	1,5	
Testdurchläufe	3	3	
Testspiele pro Durchlauf	1000	1000	

Tabelle 5.1: Übersicht über die Paramenter der Versuchsgruppen

Ein Versuch mit einer Lern- und zwei Testphasen besteht damit aus 6500 + 6000 + 6500 = 19000 Spielen und eine Versuchsgruppe aus 57000 Spielen, wobei bei der Versuchsgruppe E die zweite Testphase weggelassen wurde, um Zeit zu sparen. Damit wurden für die Versuchsguppen in diesem Kapitel 228000 + 37500 = 265500 Spiele durchgeführt, wobei die Laufzeit für diese Spiele ungefähr 18 Tage betrug.

5.2 Ergebnisse der Versuchsgruppe A

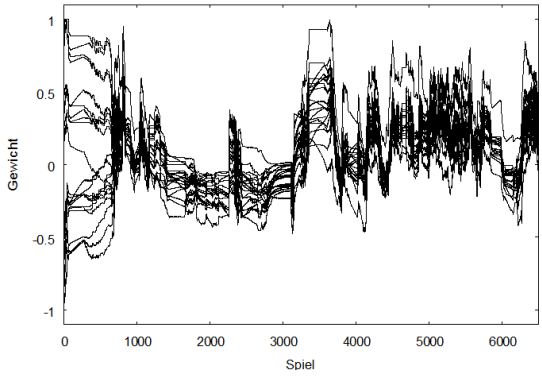
5.2.1 Lernphasen

Die Versuchsgruppe A wurde mit dem Lernfaktor $\alpha = 1,0$ und dem Differenzierungsfaktor $\gamma = 1,0$ gestartet. Die Werte der einzelnen Gewichte, die die drei Ergebnispolynome dieser Versuchsgruppe haben, sind in Anhang B dargestellt.

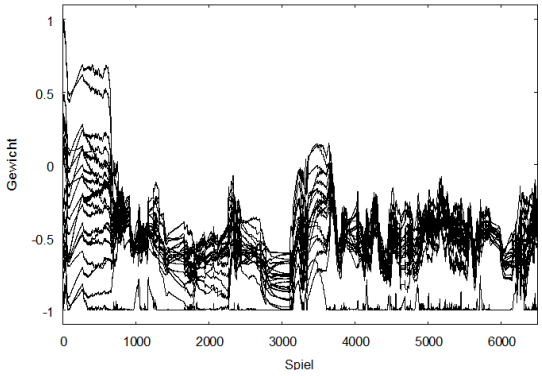
Abbildung 5.1 stellt den Verlauf der Gewichte in den drei Lernphasen der Versuchsgruppe A grafisch dar. Auf der linken Seite befinden sich die Gewichte, die zu Features gehören, die für eine Spielposition Aspekte betrachten, die die Steine des Spielers betreffen, wie relative Anzahl der Steine des Spielers auf bestimmten Feldern oder Spieler im Materialvorteil. Im Folgenden werden diese Gewichte kurz als Spielergewichte bezeichnet. Entsprechend werden als Gegnergewichte die Gewichte bezeichnet, die zu Features gehören, die für eine Spielposition die Aspekte betrachten, die die Steine des Gegners betrefffen. Die Gegnergewichte sind auf der rechten Seite dargestellt.

In allen sechs Darstellungen sind die zufällig bestimmten Gewichte bei Spiel Null über das Intervall [-1, 1] verteilt. Nach etwa $1/4$ der Lernspiele liegen die Spielergewichte vorwiegend im oberen Teil der Koordinatensysteme und die Gegnergewichte vorwiegend im unteren Teil. Dies entspricht der intuitiven Erwartung, dass ein Spieler die eigenen Steine als vorteilhaft bewerten und die gegnerischen Steine als negativ bewerten würde.

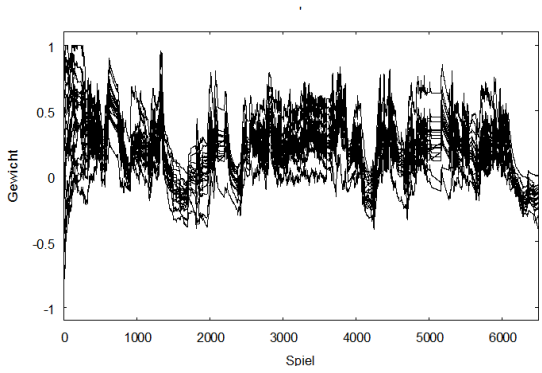
Der visuelle Eindruck bei allen sechs Darstellungen ist, dass die Gewichte über einen größeren Bereich schwanken - auch am Ende der Lernphasen. Das Problem bei diesen schwankenden Gewichten ist, dass die Lernphase nicht abhängig von definierten Größen beendet wird, sondern nach dem 6500. Spiel, wobei der Wert 6500 willkürlich bestimmt wurde. Das Ergebnispolynom eines Versuchs ist damit das Polynom, das die Alpha-Komponente am Ende des 6500. Lernspiels hat. Hätte die Lernphase einige Spiele vorher geendet, beispielsweise nach dem 5000. Spiel, hätten sich andere Gewichte für die Ergebnispolynome ergeben. Im zweiten und im dritten Versuch fallen die Spielergewichte am Ende der Lernphasen, wie vorher schon mehrmals, deutlich auf Werte um oder kleiner Null. Sie liegen zwar noch oberhalb der Gegnergewichte, trotzdem widersprechen negative Spielergewichte der intuitiven Erwartung. Beim Vergleich mit den Gewinnhäufigkeiten innerhalb der Lernphasen in Abbildung 5.2 fällt auf, dass in den Intervallen am Ende der Lernphasen des zweiten und des dritten Versuchs, in den die Gewichte insgesamt fallen die Gewinnhäufigkeit der Alpha-Komponente nicht steigt. Im Fall des zweiten Versuchs gewinnt die Alpha-Komponente drei Spiele der ungefähr letzten 400 Lernspiele und im Fall des dritten Versuchs gewinnt die Alpha-Komponente in den ungefähr letzten 1000 Lernspielen nicht ein Spiel. Das bedeutet auch, dass es der Alpha-Komponenten in 400 bzw. 1000 Lernspielen nicht gelingt, die Gewichte so zu verändern, dass sie sich gegenüber der Beta-Komponente verbessert.



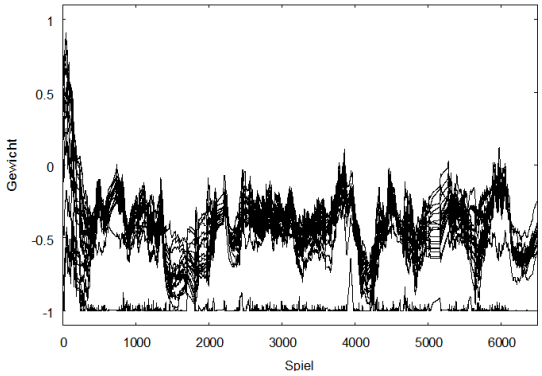
(a) Alle Spielergewichte in Versuch 1



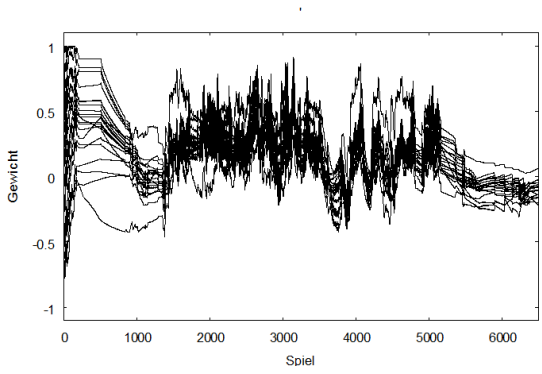
(b) Alle Gegnergewichte in Versuch 1



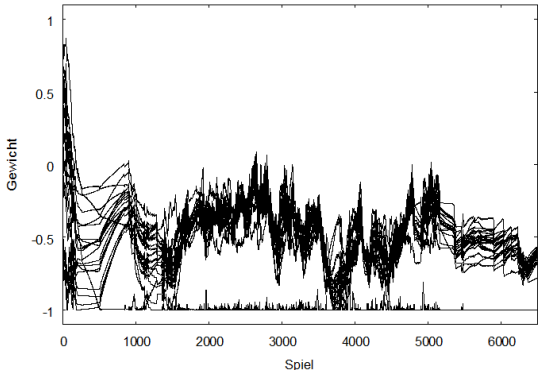
(c) Alle Spielergewichte in Versuch 2



(d) Alle Gegnergewichte in Versuch 2

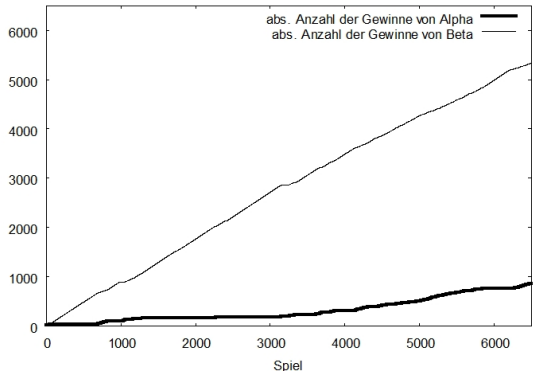


(e) Alle Spielergewichte in Versuch 3

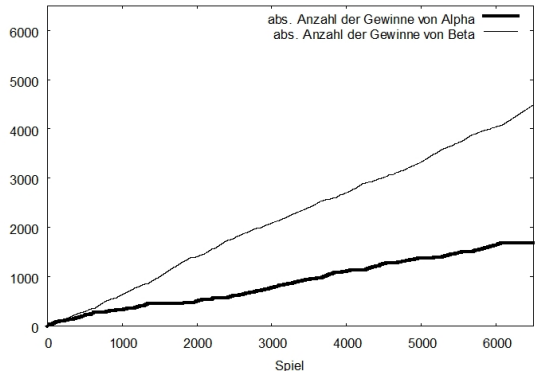


(f) Alle Gegnergewichte in Versuch 3

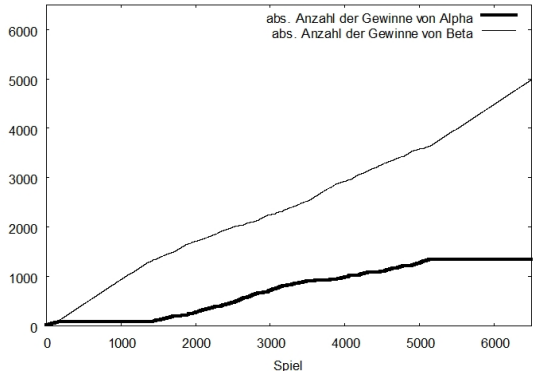
Abbildung 5.1: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe A



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung 5.2: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe A

Einige Kennzahlen Q_{spiel} und Q^*_{spiel} sind in Tabelle 5.2 dargestellt. Bis jetzt sind diese Zahlen noch nicht besonders nützlich, es kann lediglich festgestellt werden, dass sich die Ergebnispolynome ähnlicher sind, als die Startpolynome, da Q_{6500} kleiner als Q_0 ist. Die Werte können aber später mit den Kennzahlen der folgenden Versuchsgruppen verglichen werden.

Gruppe	Q_0	Q_{6500}	Q^*_{5000}	Q^*_{5500}	Q^*_{6000}
Versuchsgruppe A	18,24	6,052	7019,0	5170,0	2677,0

Tabelle 5.2: Kennzahlen mit den Standardabweichungen in der Versuchsgruppe A

Da die Gewichte diesen starken Schwankungen unterliegen und da schon aus der grafischen Darstellung der Gewichte in den Lernphasen hervorgeht, dass sich die drei Ergebnispolynome deutlich unterscheiden, werden die Graphen der Per Feature-Darstellung nicht abgebildet.

5.2.2 Erste Testphasen

In Tabelle 5.3 sind die Ergebnisse der ersten Testphasen aus Versuchsgruppe A dargestellt[8]. Nur im ersten Versuch hat sich das Ergebnispolynom gegenüber dem Startpolynom beim Spielen gegen den Zufallsspieler leicht verbessert: Im Mittel erreicht das Startpolynom eine Gewinnhäufigkeit von 42,8 Prozent und das Ergebnispolynom gewinnt 52,2 Prozent der Testspiele. Damit ist die mittlere Spielstärke des Ergebnispolynoms, trotz Steigerung, nicht hoch, denn es hat ungefähr die Hälfte der Testspiele verloren.
Im zweiten Versuch hat sich das Ergebnispolynom im Mittel leicht gegenüber dem Startpolynom verschlechtert. Während das Startpolynom im Mittel 45,8 Prozent der Testspiele gegen den Zufallsspieler gewinnt, gewinnt das Ergebnispolynom im Mittel nur 44,4 Prozent. Das Problem bei diesen nah beieinander liegenden Werten ist, dass sich nicht sagen lässt, ob sich das Ergebnispolynom tatsächlich verschlechtert hat, da der Unterschied der Spielstärken nicht signifikant ist. Unter der Annahme, die Spielstärke eines Polynoms, die in einem Testdurchlauf mit 1000 Testspielen gegen den Zufallsspieler ermittelt wird, ist normalverteilt und unter der Annahme, die für die mit nur drei Durchläufen für das Startpolynom ermittelte Standardabweichung von $\sigma_x = 1,84\%$ und der mit nur drei Durchläufen ermittelte Mittelwert von $\bar{x} = 45,8\%$ sind repräsentativ für die zugrunde liegende Grundgesamtheit, ist der für die Spielstärke des Ergebnispolynoms berechnete Mittelwert von $\bar{y} = 44,4\%$ nicht signifikant kleiner. Signifikant kleiner wären Werte, die kleiner als $45,8\% - 2 \cdot 1,84\% = 42,1\%$ sind. Die Spielstärke des Ergebnispolynoms ist schlecht, da es über die Hälfte der Testspiele verloren hat.
Auch beim dritten Versuch hat sich die Spielstärke des Ergebnispolynoms gegenüber dem Startpolynom im Mittel verschlechtert. Während das Startpolynom gemessen am Zufallsspieler eine mittlere Spielstärke von 50,4 Prozent erreicht, erreicht das Ergebnispolynom nur eine mittlere Spielstärke von 47,2 Prozent. Wie beim zweiten Versuch unterscheiden sich die beiden Mittelwerte nicht signifikant, so dass die Wahrscheinlichkeit auch hier hoch ist, dass der Unterschied zufällig zustande gekommen ist. Die mittlere Spielstärke des Ergebnispolynoms ist wieder schlecht.
Dass sich die mittleren Spielstärken der Ergebnispolynome in den drei Versuchen unterscheiden ist Ausdruck dafür, dass in den drei Versuchen drei Ergebnispolynome entstanden sind, die sich nicht ähnlich sind. Dabei ist das spielstärkste Ergebnispolynom in dem Versuch entstanden, in dem die Gewichte am Ende der Lernphase nicht deutlich gefallen sind und bei dem es kein größeres Intervall gab, in dem die Alpha-Komponente kein oder nur sehr wenig Spiele gewonnen hat.

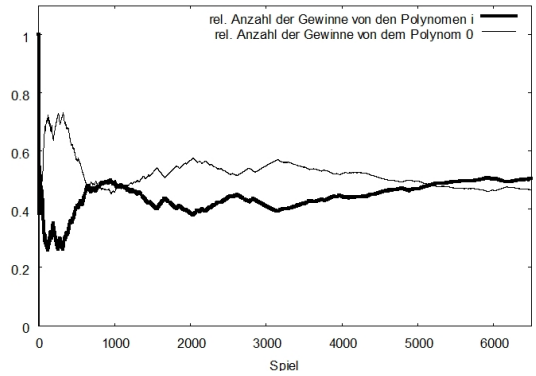
[8] Dargestellt sind die relativen Gewinnhäufigkeiten der Polynome. D.h. nicht dargestellt ist das Verhältnis von unentschieden und verloren, da nur interessiert, wie häufig die Gewinne sind. Die Häufigkeit von unentschieden liegt in allen Versuchsgruppen dieses Kapitels unter 5 Prozent.

Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	41,8 %	43,3 %	43,3 %	42,8 %	0,71 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	52,8 %	52,0 %	51,8 %	52,2 %	0,43 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	47,2 %	43,2 %	47,0 %	45,8 %	1,84 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	43,7 %	44,3 %	45,3 %	44,4 %	0,66 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	51,1 %	52,4 %	47,6 %	50,4 %	2,03 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	46,7 %	49,1 %	45,7 %	47,2 %	1,43 %

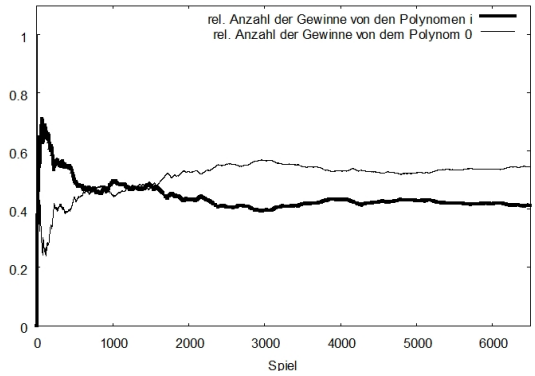
Tabelle 5.3: Ergebnisse der ersten Testphasen in der Versuchsgruppe A

5.2.3 Zweite Testphasen

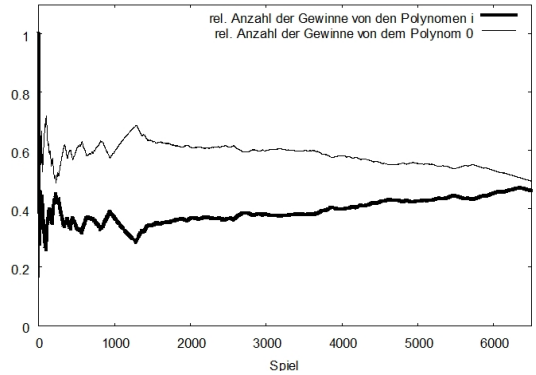
In Abbildung 5.3 sind die Verläufe der Gewinnhäufigkeiten in den zweiten Testphasen dargestellt.
Nur im ersten Versuch liegt die relative Gewinnhäufigkeit, die die gelernten Polynome P_1 bis P_{6500} gemeinsam erzielen, knapp über der des Startpolynoms P_0 mit den zufälligen Gewichten. In den anderen beiden Versuchen liegt die relative Gewinnhäufigkeit der gelernten Polynome unter der der jeweiligen Startpolynome. Während im zweiten Versuch keine Steigerung der relativen Gewinnhäufigkeiten der gelernten Polynome und des Startpolynoms zu erkennen ist, steigen die Gewinnhäufigkeiten der gelerten Polynome im ersten Versuch ungefähr ab Polynom P_{3000} und im zweiten Versuch ungefähr ab dem Polynom P_{1200} bis zum Ende. Die Intervalle, in denen die Alpha-Komponente nicht oder fast nicht gewonnen hat, sind in den Graphen der zweiten Testphasen nicht auszumachen.



(a) Relative Gewinnhäufigkeiten in Versuch 1



(b) Relative Gewinnhäufigkeiten in Versuch 2



(c) Relative Gewinnhäufigkeiten in Versuch 3

Abbildung 5.3: Gewinnhäufigkeiten in den zweiten Testphasen der Versuchsgruppe A

5.3 Ergebnisse der Versuchsgruppe B

Der Lernfaktor der Versuchsgruppe B ist mit $\alpha = 0,1$ im Vergleich zur Versuchsgruppe A zehnmal kleiner. Das Ziel eines kleineren Lernfaktors ist, die in der Versuchsgruppe A beobachteten Schwankungen der Gewichte zu reduzieren. Der Differenzierungsfaktor dieser Gruppe ist $\gamma = 1,0$.

5.3.1 Lernphasen

Die Werte der einzelnen Gewichte, die die Ergebnispolynome dieser Versuchsgruppe haben, sind wieder in Anhang B dargestellt.

In Abbildung 5.4 ist der Verlauf der Spieler- und der Gegnergewichte in den Lernphasen

dargestellt. Wie in Versuchsgruppe A haben die Spielergewichte nach mehreren Lernspielen Werte, die mehrheitlich oberhalb der x-Achse liegen und die Gegnergewichte haben Werte, die unterhalb der x-Achse liegen. Die Phase, in der die Gewichte die x-Achse überschreiten und mehrheitlich Werte ober- bzw. unterhalb der x-Achse einnehmen ist bei Versuchsgruppe B größer als bei Versuchsgruppe A. In jedem der drei Versuche sind die Spielergewichte am Ende der Lernphasen größer, als die Gegnergewichte. Die Schwankungen der Gewichte sind kleiner als in Versuchsgruppe A.

Im ersten Versuch liegen die Spielergewichte am Ende der Lernphase fast alle oberhalb der x-Achse und die Gegnergewichte liegen alle darunter. Damit entsprechen die Werte der Gewichte im ersten Versuch den intuitiven Erwartungen, dass eigene Steine als positiv zu bewerten sind und die gegnerischen als negativ. Innerhalb der Lernphase dieses Versuchs gibt es nach Abbildung 5.5 keine größeren Intervalle, in denen die Alpha-Komponente nicht gewinnt. Die relative Gewinhäufigkeit der Alpha-Komponente innerhalb der gesamten Lernphase liegt bei 34,0 Prozent.

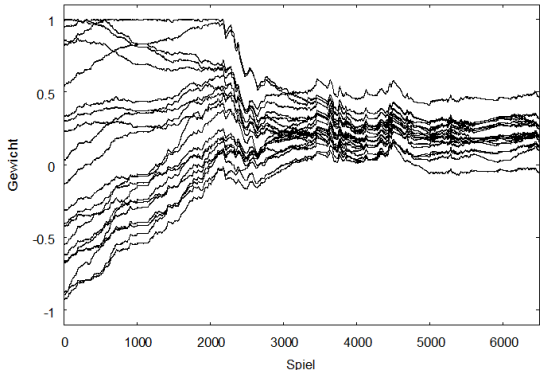
Obwohl die Spielergewichte auch im zweiten Versuch am Ende der Lernphase mehrheitlich oberhalb der x-Achse liegen, haben nur vier Spielergewichte einen Wert, der oberhalb von 0,25 liegt. Im Vergleich dazu haben im ersten Versuch acht Spielergewichte einen Wert oberhalb von 0,25. Die Summe über alle Spielergewichte beträgt im zweiten Versuch nur 2,9; im ersten Versuch dagegen 4,6. Bei den Gewinnhäufigkeiten innerhalb der Lernphase, die in Abbildung 5.5 dargestellt sind, fällt auf, dass es große Intervalle gibt, in denen die Alpha-Komponente nicht gewinnt. Die relative Gewinnhäufigkeit der Alpha-Komponente innerhalb der gesamten Lernphase liegt nur bei 13,5 Prozent.

In der Lernphase des dritten Versuchs fallen die Spieler- und die Gegnergewichte ungefähr ab dem 6000. Lernspiel deutlich; die Spielergewichte des Ergebnispolynoms nehmen sogar mehrheitlich negative Werte an. Beim Vergleich mit den Gewinnhäufigkeiten innerhalb der Lernphase fällt auf, dass die Alpha-Komponente insgesamt mit 32,3 Prozent eine relative Gewinnhäufigkeit erreicht, die mit der des ersten Versuchs vergleichbar ist, dass aber am Ende der Lernphase die Gewinnhäufigkeit der Alpha-Komponente kaum steigt; tatsächlich gewinnt sie von den letzten 1000 Lernspielen nur ungefähr 100 Spiele.

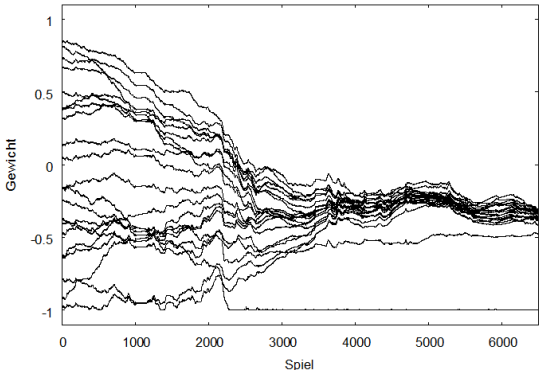
In Abbildung 5.4 sind die Kennzahlen der Versuchsgruppe B dargestellt, die sich auf die Standardabweichungen der Gewichte stützen. Der Vergleich mit den Kennzahlen aus Versuchsgruppe A ergibt, dass sich die Ergebnispolynome und die Polynome am Ende der Lernphasen in Versuchsgruppe B etwas ähnlicher sind als in Versuchsgruppe A, denn die Kennzahlen Q_{6500}, Q^*_{5000}, Q^*_{5500} und Q^*_{6000} sind in Versuchsgruppe B alle etwas kleiner - allerdings nicht wesentlich.

Gruppe	Q_0	Q_{6500}	Q^*_{5000}	Q^*_{5500}	Q^*_{6000}
Versuchsgruppe B	20,02	5,841	6075,0	4077,0	2599,0

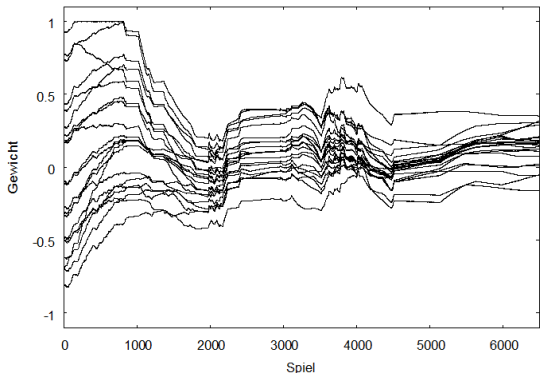
Tabelle 5.4: Kennzahlen mit den Standardabweichungen in Versuchsgruppe B



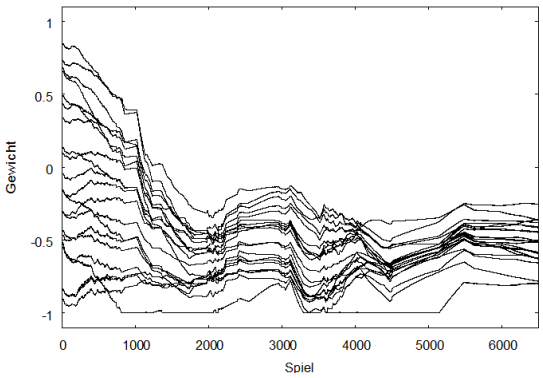
(a) Alle Spielergewichte in Versuch 1



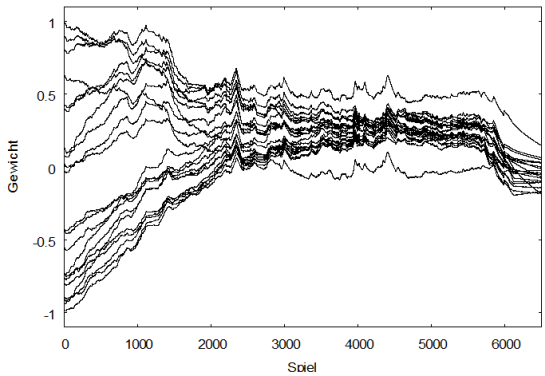
(b) Alle Gegnergewichte in Versuch 1



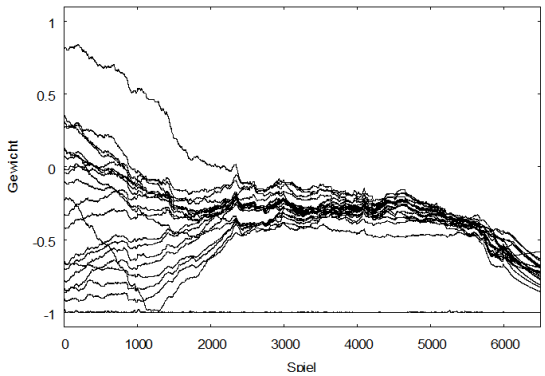
(c) Alle Spielergewichte in Versuch 2



(d) Alle Gegnergewichte in Versuch 2

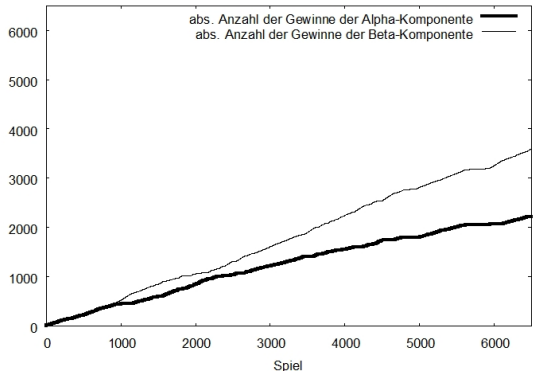


(e) Alle Spielergewichte in Versuch 3

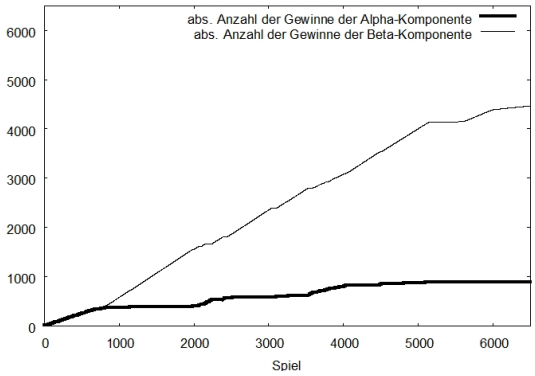


(f) Alle Gegnergewichte in Versuch 3

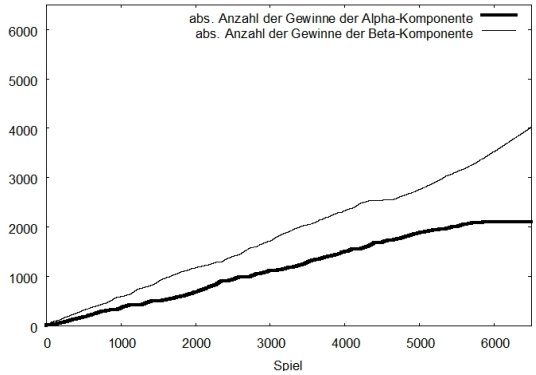
Abbildung 5.4: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe B



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung 5.5: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe B

Wie bei Versuchsgruppe A ist schon aus der Darstellung des Verlaufs der Gewichte innerhalb der Lernphasen in Abbildung 5.4 ersichtlich, dass sich die Ergebnispolynome der drei Versuche unterscheiden, so dass auf die Per Feature-Darstellung, die visualisieren soll, ob die einzelnen Gewichte in den drei Versuchen auf ein ähnliches Ergebnis kommen, verzichtet werden kann.

5.3.2 Erste Testphasen

In Tabelle 5.5 sind die Gewinnhäufigkeiten der Start- und der Ergebnispolynome der drei Versuche dargestellt, die sich in der ersten Testphase gegen den Zufallsspieler ergeben, sowie deren Mittelwert und Standardabweichung.
Beim ersten Versuch tritt wieder der Fall ein, dass sich die Mittelwerte von Start- und Endpolynom nicht signifikant unterscheiden. Der Mittelwert der Spielstärke des Startpolynoms

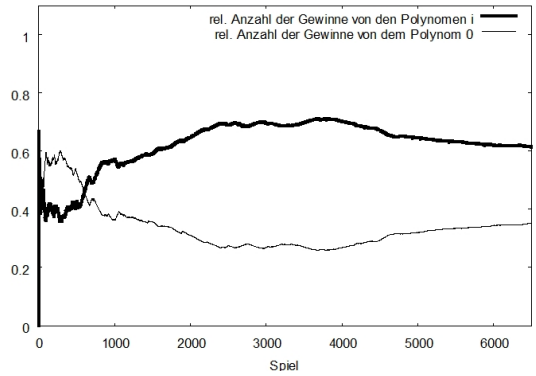
beträgt $\bar{x} = 50,9\%$ und die Standardabweichung $\sigma_x = 1,52\%$, d.h. signifikant größere Werte sind größer als 53,9%. Die drei gemessenen Spielstärken des Ergebnispolynoms sowie ihr Mittelwert sind aber kleiner. Wenn sich der Wert $\bar{x}$ nicht signifikant von $\bar{x}$ unterscheidet, kann der Unterschied auch zufällig entstanden sein und es kann nicht entschieden werden, ob sich das Ergebnispolynom gegenüber dem Startpolynom verbessert hat. Die erste Testphase ist eher geeignet, Veränderungen in größerer Größenordnung abzuschätzen. Das Ergebnispolynom ist mit einer mittleren Spielstärke von 52,9 Prozent nicht spielstark.
Signifikant ist dagegen im zweiten Versuch der Unterschied der mittleren Spielstärke des Startpolynoms und der mittleren Spielstärke des Ergebnispolynoms. Während das Startpolynom 64,3 Prozent der Testspiele gegen den Zufallsspieler gewinnt, gewinnt das Ergebnispolynom davon nur 42,5 Prozent. Das bedeutet, dass sich durch die Gewichtsänderungen in der Lernphase die Spielstärke der Alpha-Komponente verschlechtert hat.
Ebenfalls signifikant verschieden sind die Werte der beiden mittleren Spielstärken im dritten Versuch. Aus dem Startpolynom mit einer Spielstärke von 29,3 Prozent hat sich im Laufe der Lernphase ein Ergebnispolynom mit einer Spielstärke von 45,3 Prozent entwickelt. Die Spielstärke des Ergebnispolynoms ist damit gegenüber der des Startpolynoms verbessert, aber ein spielstarkes Ergebnispolynom ist nicht entstanden, denn immerhin hat das Ergebnispolynom über die Hälfte der Testspiele gegen den Zufallsspieler verloren.

Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	53,0 %	50,0 %	49,6 %	50,9 %	1,52 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	51,8 %	53,3 %	53,5 %	52,9 %	0,76 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	62,9 %	67,0 %	63,1 %	64,3 %	1,89 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	43,0 %	43,4 %	41,1 %	42,5 %	1,0 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	27,9 %	29,4 %	30,5 %	29,3 %	1,07 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	43,6 %	45,8 %	46,4 %	45,3 %	1,2 %

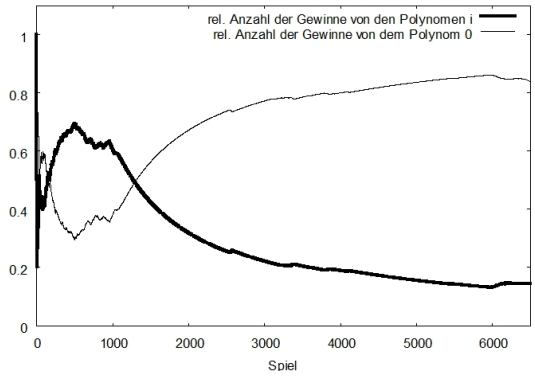
Tabelle 5.5: Ergebnisse der ersten Testphasen in der Versuchsgruppe B

5.3.3 Zweite Testphasen

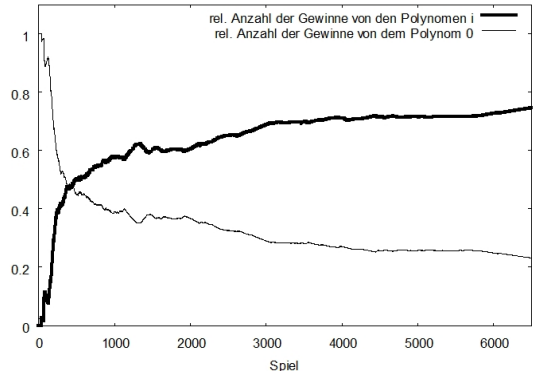
In Abbildung 5.6 sind die Ergebnisse der zweiten Testphase dargestellt. Aus dem Vergleich der Graphen des zweiten und des dritten Versuchs ist deutlich der Einfluss der Startpolynome P_0 auf die Ergebnisse dieses Tests abzulesen. In beiden Versuchen haben die Ergebnispolynome eine Spielstärke von ungefähr 45 Prozent gegen den Zufallsspieler im ersten Test erreicht. In diesem zweiten Test zeigt sich, dass die relative Gewinnhäufigkeit der gelernten Polynome P_1 bis P_{6500} im zweiten Versuch bis zum Ende des Tests auf unter 20 Prozent sinkt. Die relative Gewinnhäufigkeit der Polynome P_1 bis P_{6500} des dritten Versuchs steigt dagegen auf fast 75 Prozent. Der Grund für dieses Verhalten - bei einer vergleichbaren Spielstärke der Ergebnispolynome - liegt in der Spielstärke des jeweiligen Startpolynoms. Das Startpolynom im zweiten Versuch ist spielstark und es gelingt dem Lernverfahren in 6500 Lernspielen nicht Polynome in größerer Anzahl zu erzeugen, die dem Startpolynom überlegen sind. Im dritten Versuch handelt es sich um ein spielschwaches Startpolynom und das Lernverfahren erzeugt Polynome, die diesem Startpolynom mehrheitlich überlegen sind, obwohl nach dem 6500. Lernspiel kein spielstarkes Ergebnispolynom entstanden ist.
Auch beim Vergleich des ersten mit dem dritten Versuch spielt die Spielstärke des Startpolynoms eine wichtige Rolle. Die relative Gewinnhäufigkeit der Polynome P_1 bis P_{6500} aus dem ersten Versuch erreicht insgesamt einen Wert von ungefähr 60 Prozent. Dieser Wert liegt unter dem Wert von fast 75 Prozent aus dem dritten Versuch, obwohl im ersten Versuch das spielstärkere Ergebnispolynom entstanden ist. Der Grund liegt wieder in der Spielstärke der Startpolynome: Das Startpolynom des ersten Versuchs ist spielstärker, als das aus dem dritten Versuch.



(a) Relative Gewinnhäufigkeiten in Versuch 1



(b) Relative Gewinnhäufigkeiten in Versuch 2



(c) Relative Gewinnhäufigkeiten in Versuch 3

Abbildung 5.6: Gewinnhäufigkeiten in den zweiten Testphasen der Versuchsgruppe B

5.4 Ergebnisse der Versuchsgruppe C

Der Lernfaktor der Versuchsgruppe C ist mit $\alpha = 0{,}05$ halb so groß wie der Lernfaktor der Versuchsgruppe B. Das Ziel dieses kleineren Lernfaktors ist, die in der Versuchsgruppe B noch beobachteten Schwankungen der Gewichte weiter zu reduzieren. Der Differenzierungsfaktor dieser Versuchsgruppe beträgt $\gamma = 1{,}0$.

5.4.1 Lernphasen

Aus Abbildung 5.7, in der der Verlauf der Gewichte in den drei Lernphasen der Versuchsgruppe C dargestellt ist, wird deutlich, dass die Gewichte deutlich weniger schwanken, als in den anderen beiden Versuchsgruppen, es aber wieder in 6500 Lernspielen nicht gelungen ist, dass sich die Gewichte eines Features in den drei Ergebnispolynomen ähnlich sind:

Während sich die Spielergewichte des Ergebnispolynoms im ersten Versuch alle oberhalb der x-Achse befinden, befinden sich die Spielergewichte im zweiten Versuch mehrheitlich oberhalb der x-Achse und die des dritten Versuchs sind weit über das Intervall [-1, 1] verteilt und entsprechen so in keiner Weise der Erwartung positiver Spielergewichte. Die Gegenergewichte des Ergebnispolynoms entsprechen im ersten und im zweiten Versuch den Erwartungen negativer Gegnergewichte, sind aber im dritten Versuch über das Intervall [-1, 1] verteilt.

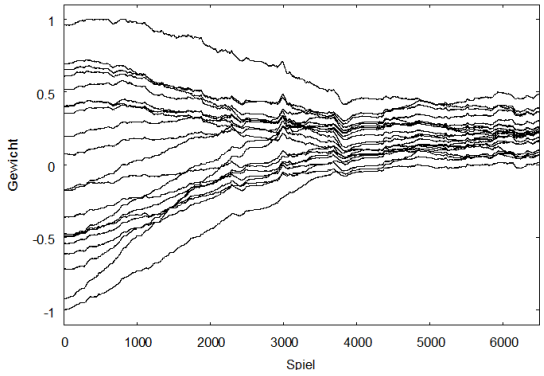
Beim zweiten und beim dritten Versuch fallen große Intervalle auf, in denen sich die Gewichte kaum ändern. Beim Vergleich mit den Gewinnhäufigkeiten während der Lernphasen, die in Abbildung 5.8 dargestellt werden, fällt auf, dass die Gewinnhäufigkeit der Alpha-Komponente in diesen Intervallen nicht bzw. kaum steigt. Tatsächlich gewinnt die Alpha-Komponente im zweiten Versuch ungefähr ab dem 3400. Lernspiel nur dreimal. Beim dritten Versuch gewinnt die Alpha-Komponente ungefähr vom 1900. bis zum 5600. Lernspiel nur sechsmal.

Die Tabelle B.3 im Anhang, in der die Gewichte der drei Ergebnispolynome zusammen mit dem jeweiligen Mittelwert und der jeweiligen Standardabweichung dargestellt sind, ergibt, dass die Gewichte von vierzehn Features eine Standardabweichung haben, die größer als 0,3 ist. Entsprechend fallen die Kennzahlen Q aus, die in Tabelle 5.6 dargestellt sind: Sie ergeben zwar, dass durch das Lernverfahren erreicht wurde, dass sich die drei Ergebnispolynome ähnlicher sind als die drei Startpolynome, trotzdem sind sich die Ergebnispolynome der Versuchsgruppe C weniger ähnlich, als die der Versuchsgruppen A und B: Q_{6500} hat in der Versuchsgruppe C einen Wert von 9,762 und in den Versuchgruppen A und B wurden für Q_{6500} Werte von 6,052 bzw. 5,841 erreicht. Ebenso verhält es sich mit den Werten Q*: Durch den im Vergleich zur Versuchsgruppe A kleineren Lernfaktor konnte in der Versuchsgruppe B erreicht werden, dass sich die die Werte von Q^*_{5000}, Q^*_{5500} und Q^*_{6000} im Vergleich zu denen der Versuchsgruppe A verkleinerten. In Versuchsgruppe C mit dem noch kleineren Lernfaktor dagegen liegen diese Werte deutlich über denen der Versuchsgruppe A.

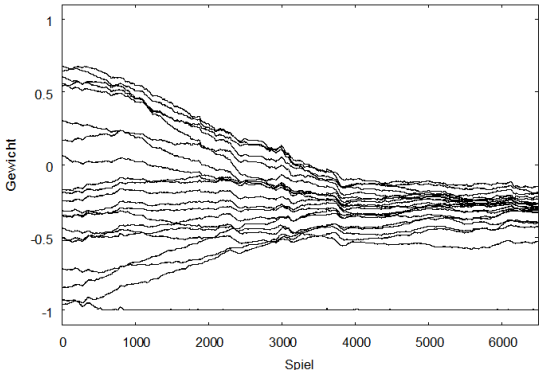
Gruppe	Q_0	Q_{6500}	Q^*_{5000}	Q^*_{5500}	Q^*_{6000}
Versuchsgruppe 3	17,26	9,762	15340,0	10050,0	4914,0

Tabelle 5.6: Kennzahlen mit den Standardabweichungen in Versuchsgruppe C

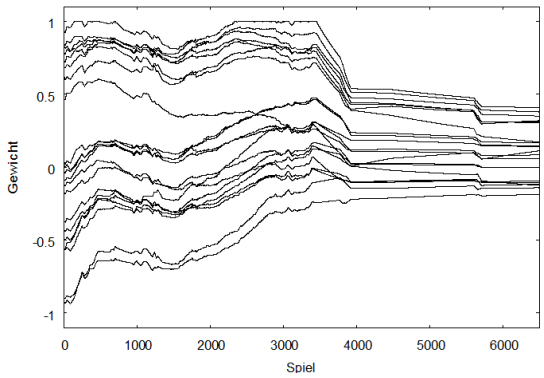
Da schon aus der Darstellung des Verlaufs der Gewichte in der Lernphase in Abbildung 5.7 ersichtlich ist, dass die drei Ergebnispolynome sich nicht ähnlich sind, wird auf die Per Feature-Darstellung verzichtet.



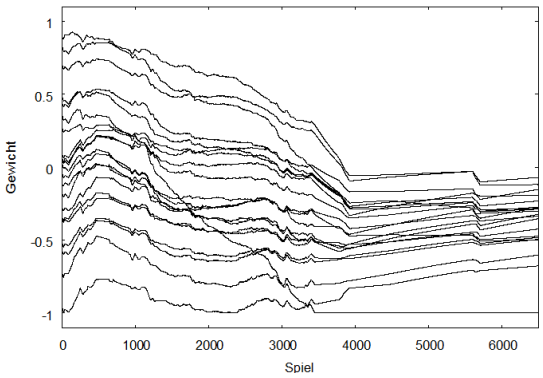
(a) Alle Spielergewichte in Versuch 1



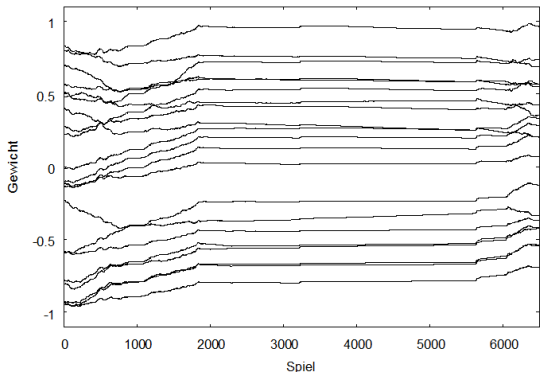
(b) Alle Gegnergewichte in Versuch 1



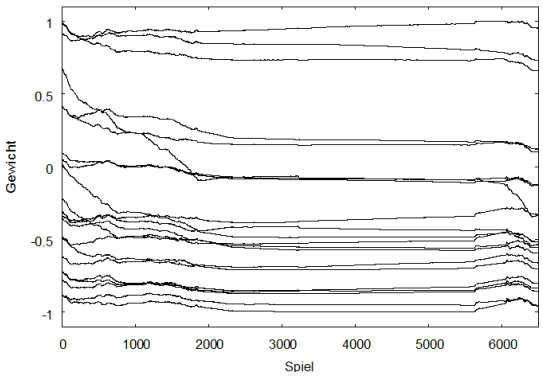
(c) Alle Spielergewichte in Versuch 2



(d) Alle Gegnergewichte in Versuch 2

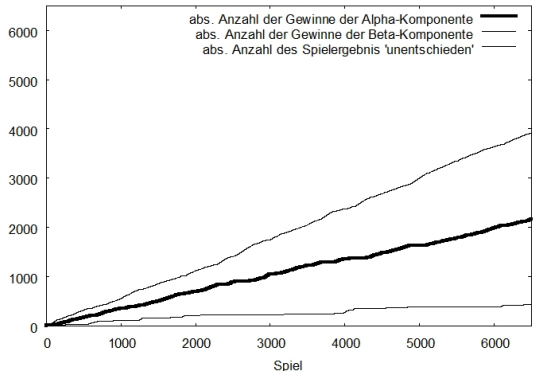


(e) Alle Spielergewichte in Versuch 3

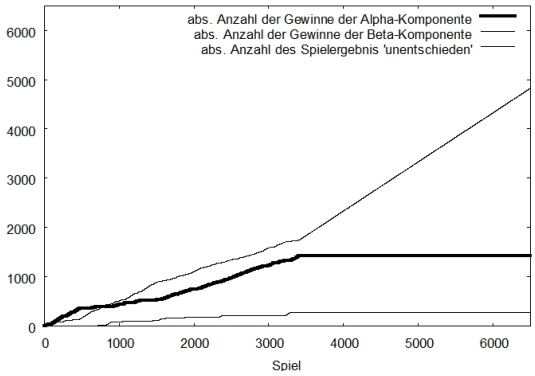


(f) Alle Gegnergewichte in Versuch 3

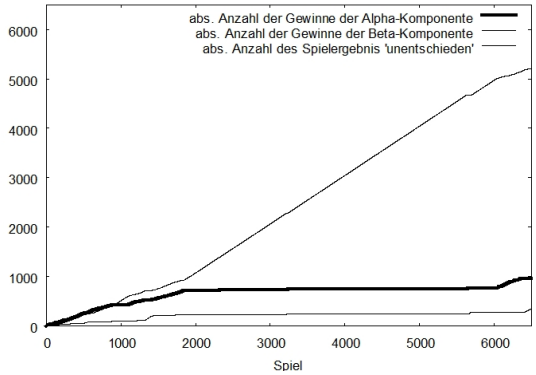
Abbildung 5.7: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe C



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung 5.8: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe C

5.4.2 Erste Testphasen

Nach Tabelle 5.7, in der die Ergebnisse der ersten Testphasen für die Versuchsgruppe C dargestellt sind, hat das Ergebnispolynom des ersten Versuchs, bei dem es kein großes Intervall gab, in dem sich die Gewichte nicht oder kaum geändert haben, beim Spielen gegen den Zufallsspieler eine mittlere Gewinnhäufigkeit von 58,6 Prozent erreicht. Es hat sich damit gegenüber dem Startpolynom, das eine mittlere Gewinnhäufigkeit von 24,9 Prozent erreicht, deutlich verbessert. Das Ergebnispolynom hat in diesem Test nur wenig mehr als die Hälfte der Spiele gewonnen und ist daher nicht besonders spielstark, dennoch ist es das beste Ergebnis, das in den Versuchsgruppen A, B und C erzielt worden ist.
Im zweiten Versuch ist es, wegen der ähnlichen Gewinnhäufigkeiten des Start- und des Ergebnispolynoms und der dazu vergleichsweise großen Standardabweichung, wieder schwierig, eine Aussage zu machen, ob sich das Ergebnispolynom gegenüber dem Startpolynom verschlechtert hat oder ob es sich um zwei Polynome mit vergleichbarer Spielstärke handelt:

Das Startpolynom gewinnt im Mittel 46,6 Prozent der Testspiele und das Ergebnispolynom 44,6 Prozent. Eine von 46,6 Prozent signifikant verschienene Spielstärke beginnt bei 44,56, die mittlere Spielstärke des Ergebnispolynoms liegt nur knapp darüber. Auf jeden Fall ist auch in diesem Versuch mit einem großen Intervall, in dem sich die Gewichte kaum ändern, ein Ergebnispolynom von geringer Spielstärke entstanden, denn es gewinnt weniger als die Hälfte der Testspiele gegen den Zufallsspieler.
Das Ergebnispolynom des dritten Versuchs, bei dem es auch ein großes Intervall gibt, in dem sich die Gewichte kaum geändert haben, gewinnt im Mittel 52,0 Prozent der Testspiele gegen den Zufallsspieler. Das bedeutet zwar eine Verbesserung gegenüber dem Startpolynom, das im Mittel 39,8 Prozent der Testspiele gewinnt, die Spielstärke des Ergebnispolynoms kann aber nicht als stark bezeichnet werden, weil es ungefähr die Hälfte der Spiele gegen den Zufallsspieler verloren hat.

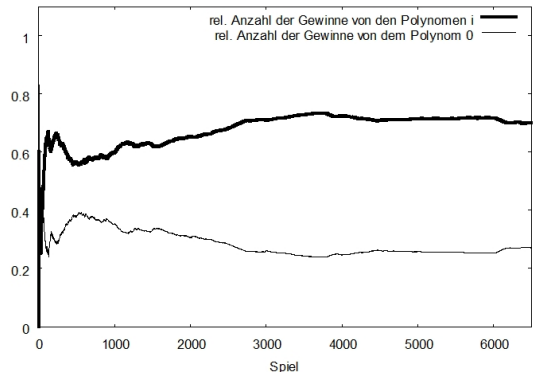
Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	24,8 %	24,3 %	25,6 %	24,9 %	0,54 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	57,6 %	60,5 %	57,8 %	58,6 %	1,32 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	47,6 %	45,2 %	47,0 %	46,6 %	1,02 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	44,5 %	43,5 %	45,7 %	44,6 %	0,9 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	41,5 %	37,1 %	40,9 %	39,8 %	1,95 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	53,1 %	50,0 %	52,9 %	52,0 %	1,42 %

Tabelle 5.7: Ergebnisse der ersten Testphasen in der Versuchsgruppe C

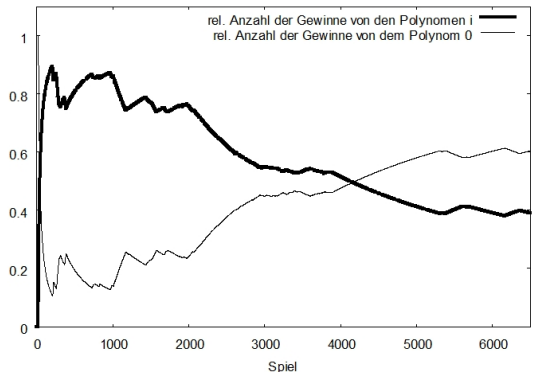
5.4.3 Zweite Testphasen

In Abbildung 5.9 sind die Verläufe der relativen Gewinnhäufigkeiten in den drei zweiten Testphasen dargestellt. Gemessen am jeweiligen Startpolynom ist die Spielstärke der gelernten Polynome P_1 bis P_{6500} im ersten Versuch größer, als beim zweiten und dritten Versuch. Am

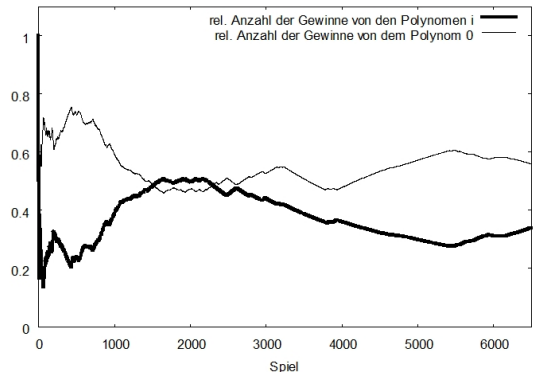
Ende der jeweiligen Tests haben im ersten Versuch die gelernten Polynome ungefähr 70 Prozent der Testspiele gegen das Startpolynom P_0 gewonnen und im zweiten und dritten Versuch nur ungefähr 40 bzw. 35 Prozent. Das deckt sich mit den Ergebnissen der ersten Testphase, nach denen im ersten Versuch das spielstärkste Ergebnispolynom aus einem spielschwachen Startpolynom entstanden ist.



(a) Relative Gewinnhäufigkeiten in Versuch 1



(b) Relative Gewinnhäufigkeiten in Versuch 2



(c) Relative Gewinnhäufigkeiten in Versuch 3

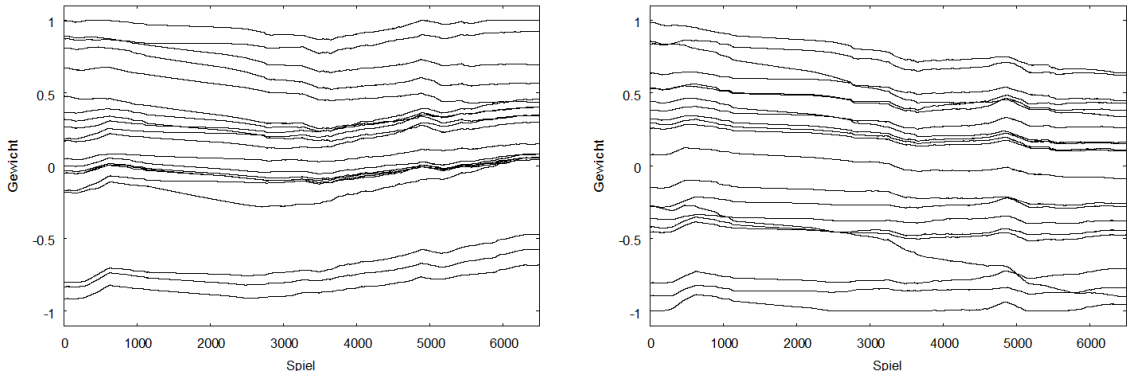
Abbildung 5.9: Gewinnhäufigkeiten in den zweiten Testphasen der Versuchsgruppe C

Beim ersten Versuch ändern sich ungefähr ab dem 4000. Lernspiel die relative Gewinnhäufigkeite der gelernten Polynome und des Startpolynoms kaum, das bedeutet, dass sich die Gewinne der gelernten Polynome und die Gewinne des Startpolynoms die Waage halten.
Beim zweiten Versuch haben die gelernten Polynome anfangs eine hohe Spielstärke, die aber bis zum Ende der zweiten Testphase bis auf die 40 Prozent sinkt und damit unter der des Startpolynoms P_0 liegt. Es gibt zwei Intervalle, in denen die Gewinnstärke der gelernten Polynome fällt: Ungefähr von Polynom P_{2000} bis zum Polynom P_{3000} und von Polynom P_{4000} bis Polynom P_{5000}. Beim zweiten Intervall handelt es sich um das Intervall, in dem sich die Gewichte in der Lernphase kaum ändern. Offensichtlich war das Polynom am Anfang dieses

Intervalls dem Startpolynom unterlegen und die folgenden Polynome bleiben unterlegen, weil sich ihre Gewichte kaum ändern. Beim ersten Intervall dagegen ändern sich die Gewichte der Polynome P_{2000} und P_{3000}, das bedeutet, dass trotz der Gewichtsänderungen Polynome entstehen, die dem Startpolynom mehrheitlich unterlegen sind.
Beim dritten Versuch steigt die Gewinnhäufigkeit der gelernten Polynome anfangs und fällt dann in dem Intervall, in denen sich die Gewichte kaum ändern. Nach diesem Intervall, wenn sich die Gewichte wieder ändern, steigt die Gewinnhäufigkeit der gelernten Polynome wieder, erreicht aber - weil die Lernphase kurz danach endet - nur eine relative Gewinnhäufigkeit von 35 Prozent.

5.5 Ergebnisse zur Versuchsgruppe D

In der Versuchsgruppe D mit dem Lernfaktor $\alpha = 0{,}01$ und dem Differenzierungsfaktor $\gamma = 1{,}0$ gehört auch zu dem Versuch die Ähnlichkeit der Ergebnisse durch die Veränderung des Lernfaktors zu erreichen. Da sich die Gewichte im Verlauf der Lernphase sehr wenig verändern und da der erste Eindruck ist, der Lernfaktor ist zu klein, wurde sie zuerst verworfen. Trotzdem sind die Ergebnisse der Versuchgruppe interessant. In Abbildung 5.10 ist der Verlauf der Gewichte in der Lernphase des ersten Versuchs dargestellt und in Abbildung 5.11 die zugehörigen relativen Gewinnhäufigkeiten[9].



(a) Alle Spielergewichte in Versuch 1

(b) Alle Gegnergewichte in Versuch 1

Abbildung 5.10: Verlauf der Gewichte in der Lernphase in der Versuchsgruppe D

Im Unterschied zu den anderen Versuchsgruppen ist nicht beobachtbar, dass große Intervalle, in den die Alpha-Komponente nicht oder kaum gewinnt, gleichzeitig mit Ergebnispolynomen auftreten, die nicht so spielstark sind, wie die Ergebnispolynome anderer Versuche aus

[9] Die Abbildungen und Tabellen für alle Versuche befinden sich in Anhang C.

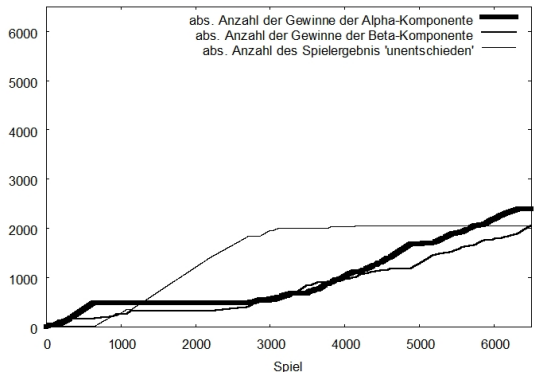
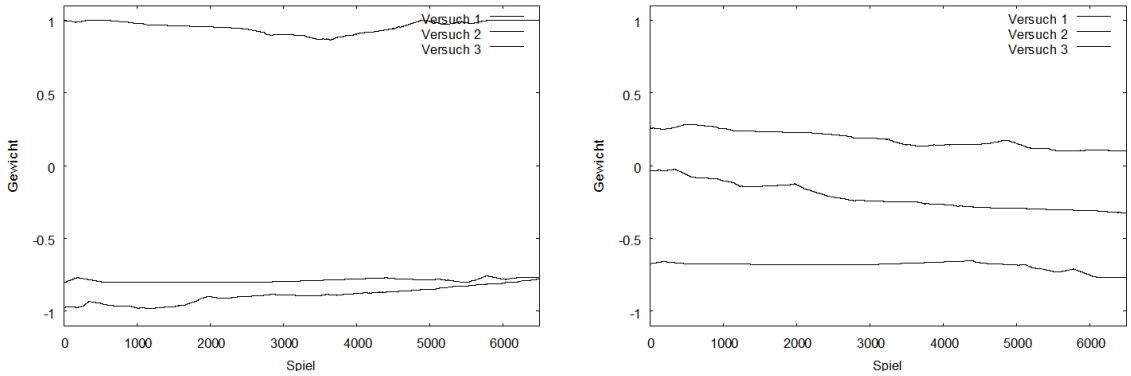


Abbildung 5.11: Absolute Anzahl der Gewinne in Versuch 1

derselben Versuchsgruppe, in denen diese Intervalle nicht auftreten. Im Gegenteil: Im ersten Versuch gewinnt die Alpha-Komponente vom 1145. bis zum 2710. Lernspiel kein Spiel, die über drei Durchläufe mit jeweils 1000 Testspielen gegen den Zufallsspieler gemessene mittlere Spielstärke des Ergebnispolynoms liegt bei diesem Versuch aber bei knapp 70 Prozent. Ein Ergebnispolynom mit einer mittleren Spielstärke von knapp 70 Prozent ist bemerkenswert, auch wenn für das Startpolynom schon eine mittlere Spielstärke von 50 Prozent gemessen wurde, denn keine der anderen Lernphasen hat ein so spielstarkes Ergebnispolynom hervorgebracht. Darüber hinaus haben die anderen beiden Versuche Ergebnispolynome erzeugt, die nach den Ergebnissen der ersten Testphase nicht spielstark sind, aber im Mittel dem jeweiligen Startpolynom überlegen sind: Im zweiten Versuch wurde aus dem Startpolynom mit einer Spielstärke von 15 Prozent ein Ergebnispolynom mit einer Spielstärke von 25 Prozent entwickelt und im dritten Versuch gewinnt das Startpolynom 30 Prozent der Testspiele gegen den Zufallsspieler und das Ergebnispolynom gewinnt 36 Prozent dieser Testspiele. Damit hat in jedem Versuch eine Verbesserung stattgefunden.

Die Darstellung der Verläufe der Gewichte zu den Features „Spielersteine in Ring 0“ und „Gegnersteine in Ring 0“ in Abbildung 5.12 verdeutlicht exemplarisch, dass sich die Gewichte in allen Versuchen wenig geändert haben und dass sich die jeweiligen drei Gewichte eines Features nicht ähnlich geworden sind. Die Werte für Q und Q* werden daher nicht berechnet.

Festzuhalten ist noch, dass die Gewichte des Ergebnispolynoms aus dem ersten Versuch, das die Spielstärke von 70 Prozent erreicht, nicht den Erwartungen negativer Gegnersteine entsprechen, aber die Spielergewichte liegen mehrheitlich im oberen Teil des Koordinatensystems.



(a) Spielersteine in Ring 0 in drei Versuchen (b) Gegnersteine in Ring 0 in drei Versuchen

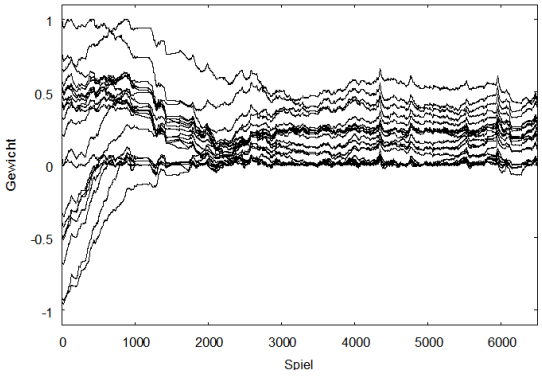
Abbildung 5.12: Verlauf zweier Gewichte in Versuchsgruppe D

5.6 Ergebnisse zur Versuchsgruppe E

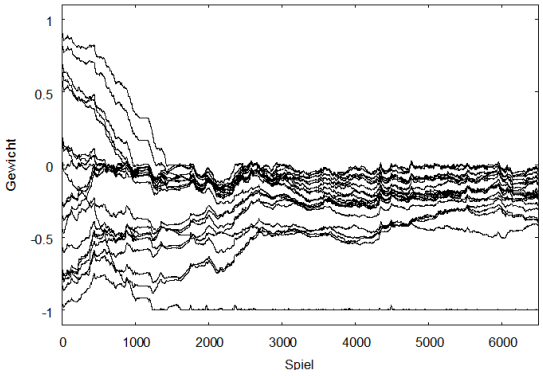
In Versuchsgruppe E wird ein Differenzierungsfaktor von $\gamma = 1,5$ benutzt; bisher hatten die Versuchsgruppen alle einen Differenzierungsfaktor von 1,0. Der Lernfaktor der Gruppe ist $\alpha = 0,1$.

In Abbildung 5.13, in der der Verlauf der Gewichte dargestellt ist, fällt auf, dass in keinem der drei Versuche Intervalle auftreten, in denen die Gewichte stark abfallen. Die Gewichte schwanken zwar, aber um kleine Beträge und ihre Graphen verlaufen trotz dieser Schwankungen am Ende der Lernphasen parallel zur x-Achse. Es gibt Intervalle, in denen sich Gewichte kaum ändern[10], trotzdem sind am Ende aller drei Lernphasen die Spielergewichte oberhalb der x-Achse und die Gegnergewichte darunter. Aus der Tabelle B.5 mit den Gewichten der drei Ergebnispolynome geht hervor, dass es keine positiven Gegnergewichte gibt und dass insgesamt vier negative Spielergewichte auftreten, ihre Beträge aber klein sind. Außerdem geht aus derselben Tabelle hervor, dass die größte Standardabweichung der drei Gewichte zu einem Feature nur 0,0829 beträgt.

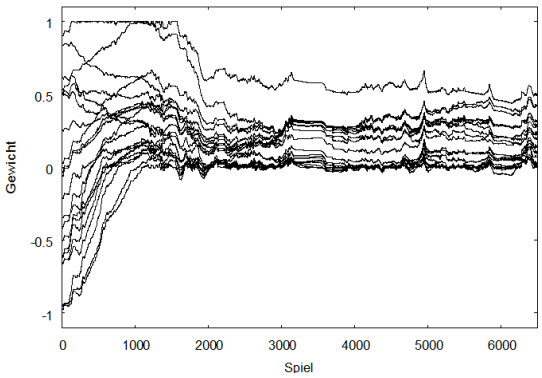
[10] und nach den Darstellungen der absoluten Häufigkeiten in der Abbildung D.1 gibt es auch in dieser Versuchsgruppe Intervalle, in denen die Alpha-Komponente nicht gewinnt



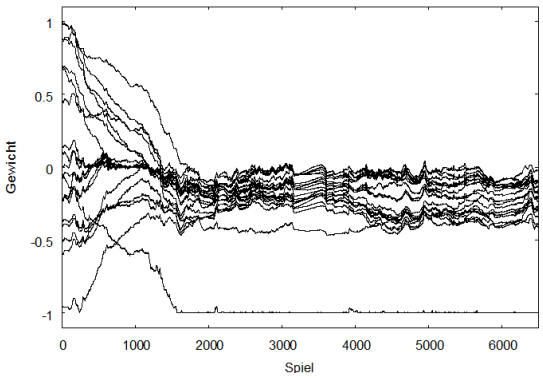
(a) Alle Spielergewichte in Versuch 1



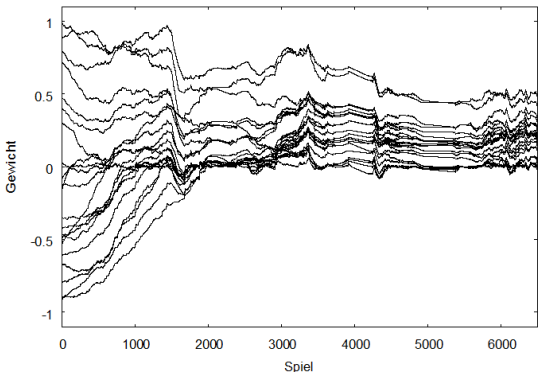
(b) Alle Gegnergewichte in Versuch 1



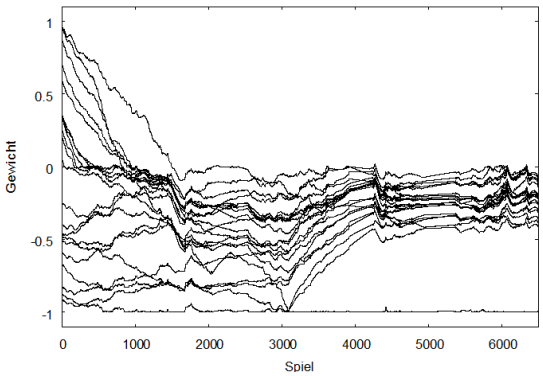
(c) Alle Spielergewichte in Versuch 2



(d) Alle Gegnergewichte in Versuch 2



(e) Alle Spielergewichte in Versuch 3



(f) Alle Gegnergewichte in Versuch 3

Abbildung 5.13: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe E

In Abbildung 5.14 wird aus den Per Feature-Darstellungen für sechs Features exemplarisch dargestellt, wie sich die Gewichte eines Features in den drei Versuchen entwickeln. Der Eindruck ist, dass sich die Ergebnispolynome ähnlich geworden sind. In den in Tabelle 5.8 dargestellten Kennzahlen, die auf den Standardabweichungen beruhen, drückt sich diese Ähnlichkeit ebenfalls aus: Die Werte $Q_{6500} = 1,153$ und $Q^*_{6000} = 682,4$ liegen deutlich unter den entsprechenden Werten, die in den Versuchsgruppe A, B und C gemessen wurden[11].

Gruppe	Q_0	Q_{6500}	Q^*_{5000}	Q^*_{5500}	Q^*_{6000}
Versuchsgruppe E	20,16	1,153	2315,0	1570,0	682,4

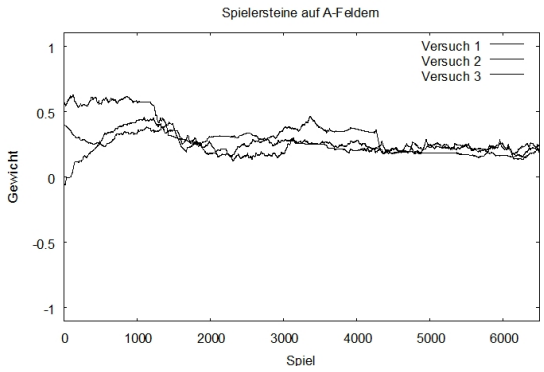
Tabelle 5.8: Kennzahlen mit den Standardabweichungen in Versuchsgruppe E

Das bedeutet, dass Othilies Lernverfahren in den drei Versuchen der Versuchsgruppe E - trotz unterschiedlicher, zufällig bestimmter Startpolynome - auf ähnliche Ergebnisse gekommen ist.

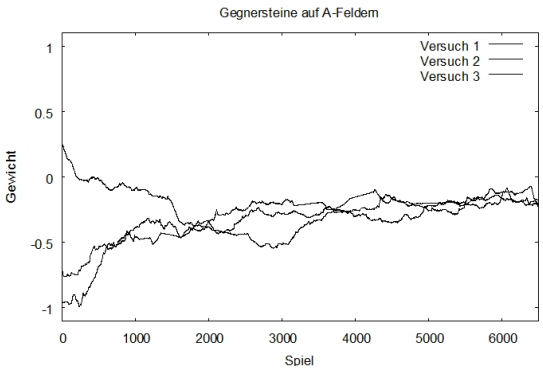
Die in drei Durchläufen mit jeweils 1000 Testspielen gegen den Zufallsspieler ermittelten Spielstärken sind in Tabelle 5.9 dargestellt. In drei Versuchen erreichen die Ergebnispolynome vergleichbare mittlere Spielstärken um 60 Prozent. Das ist das erste Mal, dass alle Ergebnispolynome einer Versuchsgruppe eine derart ähnliche Spielstärke erreichen und außerdem haben die meisten bisherigen Ergebnispolynome kleinere Spielstärken erreicht[12]. Trotzdem muss festgehalten werden, dass für das Startpolynom im ersten Versuch eine mittlere Spielstärke von 82,9 Prozent ermittelt wurde. Das bedeutet, dass sich das Ergebnispolynom gegenüber dem Startpolynom durch das Lernverfahren verschlechtert hat.

[11] Die kleinsten dieser Kennzahlen wurden in Versuchsgruppe B beobachtet: Q_{6500} für die drei Ergebnispolynome hatte den Wert 5,841 und Q^*_{6000} für den letzten Teil der Lernphase hatte den Wert 2599,0

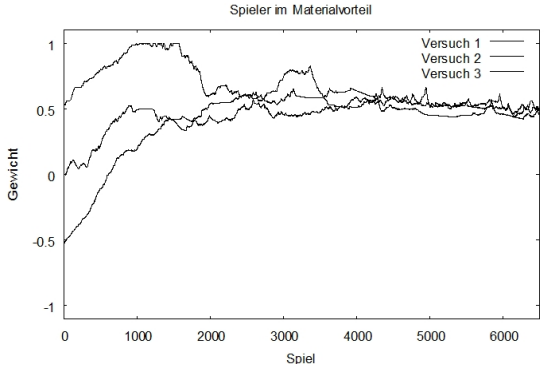
[12] Ergebnispolynome mit einer vergleichbaren oder besseren Spielstärke wurden nur im ersten Versuch der Versuchsgruppe C (58,6 Prozent)und im ersten Versuch der Versuchsgruppe D (69,0 Prozent) erreicht.



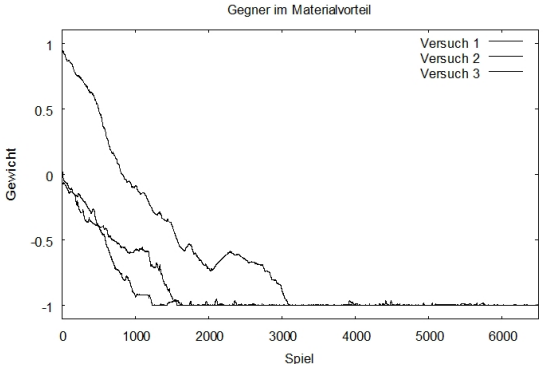
(a) Spielersteine auf A-Feldern



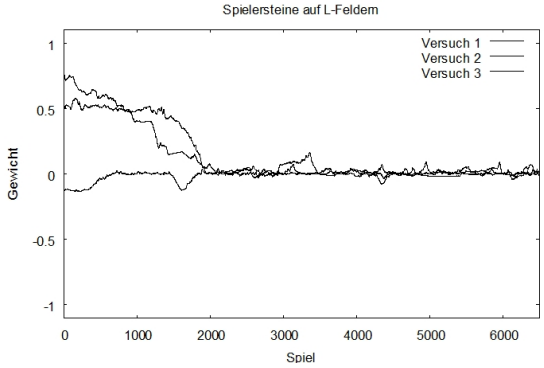
(b) Gegnersteine auf A-Feldern



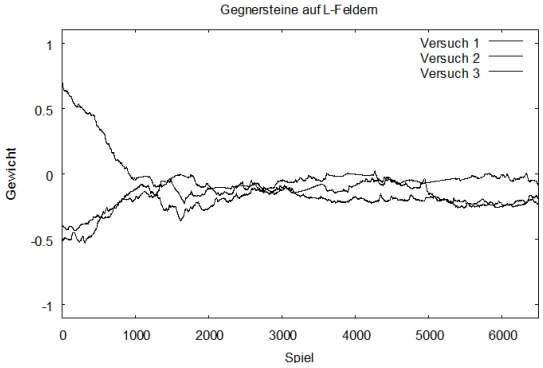
(c) Spieler im Materialvorteil



(d) Gegner im Materialvorteil



(e) Spielersteine auf L-Feldern



(f) Gegnersteine auf L-Feldern

Abbildung 5.14: Darstellung der Gewichte zu einigen Features in Versuchsgruppe E

Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	82,7 %	83,4 %	82,6 %	82,9 %	0,36 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	63,7 %	62,2 %	64,7 %	63,5 %	1,03 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	41,1 %	37,8 %	41,4 %	40,1 %	1,63 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	62,3 %	63,6 %	64,0 %	63,3 %	0,73 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	40,9 %	41,7 %	40,6 %	41,1 %	0,46 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	57,8 %	60,9 %	58,6 %	59,1 %	1,31 %

Tabelle 5.9: Ergebnisse der ersten Testphasen in der Versuchsgruppe E

5.7 Test mit einem anderen Zufallsspieler

Der Zufallsspieler der ersten Testphase benutzt zwar zufällige Gewichte, aber er setzt ein Spielbaum-Suchverfahren ein. Damit ist er kein Spieler, der aus der Menge der gültigen Züge einen beliebigen Zug auswählt, ohne ein Suchverfahren einzusetzen. Für die erste Testphase wurde der Zufallsspieler, der das Spielbaum-Suchverfahren einsetzt, gegenüber dem Zufallsspieler bevorzugt, der kein Suchverfahren einsetzt, um dem Einwand zu begegnen, der Zufallsspieler sei nur wegen des fehlenden Spielbaum-Suchverfahrens unterlegen.

Trotzdem wird mit dem Begriff Zufallsspieler häufig ein Spieler assoziiert, der aus der Menge der gültigen Züge zufällig einen auswählt und es stellt sich die Frage, wie spielstark die Ergebnispolynome gegen einen solchen Zufallsspieler sind. Aus diesem Grund wurde ein dritter Test eingeführt, der der ersten Testphase sehr ähnlich ist: In Testspielen spielt der neue Zufallsspieler, der die Züge zufällig auswählt, gegen die Start- und die Ergebnispolynome. Für ein Polynom wird die Spielstärke wieder als Mittelwert über drei Durchläufe mit jeweils 1000 Testspielen gemessen.

Für diesen neuen Test wird der TestManagerEins benutzt, wobei in der Zeile 5 des

Listings 4.5 das RandomWeightsPlayer-Objekt gegen den neuen Zufallsspieler - ein RandomPlayer-Objekt - ersetzt wird.

In der Tabelle 5.10 sind die mittleren Spielstärken der Start- und der Ergebnispolynome aus den Versuchsgruppen A, B, C, D und E dargestellt. Nicht dargestellt sind die Standardabweichungen der jeweils drei Durchläufe; sie sind alle kleiner als 2 Prozent.

Versuchsgruppe	Versuch	Startpolynom	Ergebnispolynom
A	1	71,0	78,6
A	2	64,9	73,4
A	3	67,3	74,4
B	1	76,8	78,0
B	2	79,9	72,7
B	3	56,2	73,7
C	1	54,5	81,7
C	2	66,6	72,3
C	3	71,5	79,9
D	1	73,2	86,0
D	2	27,2	38,1
D	3	49,3	63,2
E	1	92,8	85,0
E	2	60,1	85,7
E	3	62,2	83,2

Tabelle 5.10: Ergebnisse des dritten Tests

Beim Vergleich der Spielstärken in der ersten Testphase mit den Spielstärken, die im dritten Test ermittelt wurden, wird deutlich, dass der Zufallsspieler, der das Spielbaum-Suchverfahren benutzt, deutlich spielstärker ist als der Zufallsspieler, der einen zufälligen Zug auswählt, denn die Spielstärken aller Polynome sind im dritten Test signifikant größer. In der Versuchsgruppe E wurden beispielsweise für die drei Ergebnispolynome mittlere Spielstärken von 85,0, 85,7 und 83,2 Prozent gemessen. Demnach ist die Spielbaum-Suche, die einige Züge vorausschaut, dem Zufall überlegen, auch wenn die Gewichte der Bewertungsfunktion bei der Suche zufällig sind.

Werden die Spielstärken der Polynome des dritten Tests untereinander verglichen, ergeben sich dieselben Resultate wie bei der ersten Testphase: Die Ergebnispolynome der Versuchsgruppen A, B, C und D erreichen unterschiedliche Spielstärken und die Ergebnispolynome der Versuchsgruppe E erreichen ähnliche Spielstärken. Die Spielstärken der der Ergebnispolynome in Versuchsgruppe E sind mit ungefähr 85 Prozent hoch bzw. meist höher als in

den anderen Versuchsgruppen, allerdings gibt es ein spielstärkeres Polynom: Das Startpolynom des ersten Versuchs in Versuchgruppe E erreicht 92,8 Prozent. Generell gilt wieder, dass sich die Ergebnispolynome im Vergleich zum jeweiligen Startpolynom verschlechtern können: Der zweite Versuch der Versuchsgruppe B und der erste Versuch der Versuchsgruppe E sind zwei Beispiele. Die Spielstärken der Ergebnispolynome aus Versuchsgruppe D, die den kleinen Lernfaktor α hat, haben sich in jedem Versuch gegenüber der des Startpolynoms verbessert.

5.8 Zusammenfassung der Ergebnisse

5.8.1 Versuchsgruppen A, B und C

In den Versuchsgruppen A, B und C wurde versucht, die Ähnlichkeit der Ergebnispolynome durch Veränderung des Lernfaktors α - bei konstantem Differenzierungsfaktor von $\gamma = 1,0$ - zu erreichen. Dahinter stand die Idee, dass die Reduzierung der bei größeren Lernfaktoren beobachteten Schwankungen, dazu führte, dass die Ergebnispolynome der Versuche einer Versuchsgruppe gegen dasselbe Ergebnis konvergierten.

Verlauf der Gewichte

In diesen drei Versuchsgruppen konnte in der Tat beobachtet werden, dass sich die Schwankungen der Gewichte mit kleineren Lernfaktoren reduzieren ließen, eine Steigerung der Ähnlichkeit der Ergebnispolynome konnte dadurch aber nicht erreicht werden. Die Verläufe der Gewichte in den Lernphasen zeigten so unterschiedliche Ergebnisse, dass auf die Per Feature-Darstellung verzichtet wurde. Der Vergleich der Kennzahlen Q_{6500} und Q^* drückten ebenfalls aus, dass sich durch die Verkleinerung des Lernfaktors α keine Ähnlichkeit der Ergebnispolynome erreichen lässt: Bei der Versuchsgruppe C, die von den drei Versuchsgruppen den kleinsten Lernfaktor hat, wurden die größten Werte beobachtet.

Bis auf eine Ausnahme[13] hat Othilies Lernverfahren Spielergewichte produziert, die mehrheitlich oberhalb der x-Achse liegen, und Gegnergewichte, die mehrheitlich unterhalb der x-Achse liegen. Dabei gibt es am Anfang der Lernphase ein Intervall, in dem die Gewichte ihre anfänglichen, zufällig bestimmten Werte zwischen -1 und 1 überwinden und mehrheitlich ober- bzw. unterhalb der x-Achse liegen. Je kleiner der Lernfaktor ist, umso größer ist dieses Intervall. Die Versuche der Versuchsgruppe B und C zeigen, dass die Gewichte diesen intuitiven Erwartungen positiver Spieler- und negativer Gegnergewichte besonders deutlich

[13] Die Ausnahme ist der dritte Versuch der Versuchsgruppe C, in dem sich die Spieler- und die Gegnergewichte über ein sehr großes Intervall sehr wenig geändert haben.

entsprechen, wenn es innerhalb der Lernphase keine großen Intervalle gab, in denen die Alpha-Komponente nicht oder kaum gewinnt, und die Gewichte entsprechen nicht oder weniger den intuitiven Erwartungen, wenn es diese Intervalle in der Lernphase gibt - obwohl keine allgemeingültige Aussage gemacht werden, kann, ob die Gewichte in diesen Intervallen nun stark fallen, steigen oder sich kaum ändern.

Erste Testphasen

In mehreren Versuchen konnte keine Aussage getroffen werden, ob sich die mittlere Spielstärke des Ergebnispolynoms gegenüber der des Startpolynoms verbesserte, verschlechterte oder ob es sich um zwei Polynome mit vergleichbarer Spielstärke handelt. Der Grund ist, dass sich die beiden Werte in diesen Versuchen nicht signifikant unterschieden haben.

Die Unterschiedlichkeit der jeweiligen drei Ergebnispolynome einer Versuchsgruppe drückt sich auch in den mittleren Spielstärken der Ergebnispolynome aus: In keiner Versuchsgruppe wurden drei vergleichbare mittlere Spielstärken der Ergebnispolynome gemessen. Die größte gemessene Spielstärke eines Ergebnispolynoms betrug 58,6 Prozent, die anderen Werte lagen bei ungefähr 50 Prozent oder sogar nur bei ungefähr 45 Prozent. Damit gewinnen die Ergebnispolynome, bis auf eine Ausnahme, höchstens die Hälfte der Testspiele gegen den Zufallsspieler, das Lernverfahren hat also keine spielstarken Ergebnispolynome hervorgebracht.

Durch das Lernverfahren wurden im Vergleich zum Startpolynom spielstärkere und spielschwächere Ergebnispolynome erzeugt. Einmal hat sich die Spielstärke des Ergebnispolynoms von 42,5 Prozent gegenüber der Spielstärke des Startpolynoms von 64,5 Prozent erheblich verschlechtert und in einem anderen Versuch hat sich die Spielstärke des Ergebnispolynoms von 58,6 Prozent gegenüber der des Startpolynoms von 24,9 Prozent erheblich verbessert. Die anderen signifikanten Änderungen lagen im Bereich von ungefähr $\pm$ 10 Prozent.

Bei dem spielstärksten Ergebnispolynom, das aus dem ersten Versuch der Versuchgruppe C stammt und das die 58,6 Prozent der Testspiele gegen den Zufallsspieler gewinnt, entsprechen die Gewichte deutlich den Erwartungen positiver Spielergewichte und negativer Gegnergewichte. Trotzdem gilt nicht, dass die Spielstärke genau dann höher ist, wenn die Gewichte diesen Erwartungen entsprechen: Das spielstärkste Polynom, das in den Versuchsgruppen A, B und C getestet wurde, ist das Startpolynom des zweiten Versuchs der Versuchsgruppe B. Gegen den Zufallsspieler gewinnt es 64,3 Prozent der Testspiele. Nach Abbildung 5.4 sind die Spieler- und die Gegnergewichte dieses Startpolynoms über das Intervall [-1, 1] verteilt.

Zweite Testphasen

Die Ergebnisse der zweiten Testphasen zeigten, dass die relative Gewinnhäufigkeit der gelernten Polynome P_1 bis P_{6500} stark von der Spielstärke des Startpolynoms P_0 abhängig ist. An einigen Stellen konnte abgelesen werden, dass sich die Spielstärke der gelernten Polynome gegenüber der des Startpolynoms veränderte. Nicht beobachtet wurde, dass die relative Gewinnhäufigkeit der gelernten Polynome gegen Ende der Testphasen deutlich steigt, so wie es beobachtet werden würde, wenn das Lernverfahren mit fortschreitender Lernphase Polynome erzeugte, die spielstärker als das Startpolynom sind. Beobachtet wurde lediglich eine leichte Steigerung in einigen Versuchen.

Dritter Test

Der Zufallsspieler des dritten Tests, der zufällig einen Zug wählt, ist nicht so spielstark wie der Zufallsspieler der ersten Testphase, der ein Spielbaum-Suchverfahren mit zufälligen Gewichten benutzt, denn im dritten Test sind die Gewinnhäufigkeiten für alle Polynome größer. Dennoch spiegeln die Ergebnisse des dritten Tests die Ergebnisse der ersten Testphase: Die Ergebnispolynome der jeweiligen Versuchsgruppen erreichen unterschiedliche Spielstärken und die Spielstärke eines Ergebnispolynoms kann sich im Vergleich zum Startpolynom verschlechtern.

5.8.2 Versuchsgruppe D

Die Versuchsgruppe D gehört zu den Versuchsgruppen A, B und C, weil sie zu dem Versuch gehört, die Ähnlichkeit der Ergebnispolynome durch die Veränderung des Lernfaktors α zu erreichen. Die Versuchsgruppe wurde zuerst verworfen, denn Gewichte der drei Ergebnispolynome haben sich wegen des kleinen Lernfaktors kaum von den Gewichten der Startpolynome unterschieden, so dass sich die drei Ergebnispolynome in keiner Weise ähnlich geworden sind. Dies zeigt sich auch in den Spielstärken, die in der ersten Testphase und im dritten Test gemessen wurden. Die Versuchsgruppe wurde trotzdem kurz vorgestellt, weil in jedem Versuch eine Verbesserung der Spielstärke beobachtet worden ist und weil im ersten Versuch ein Ergebnispolynom entstanden ist, für das eine Spielstärke von knapp 70 Prozent ermittelt wurde - obwohl sich die Gewichte des Startpolynoms nur wenig von den Gewichtes des Ergebnispolynoms unterschieden. Dies ist das spielstärkste Ergebnispolynom der Versuchgruppen A, B, C, D und E. Wobei die Spielergewichte meist positiv waren; die Gegnergewichte waren dagegen über das Intervall [-1, 1] verteilt.

5.8.3 Versuchsgruppe E

Die Versuchsgruppe E ist die Versuchsgruppe, die mit dem Differenzierungsfaktor $\gamma = 1,5$ gestartet wurde, durch den Gewichtsänderungen in Richtung der x-Achse verstärkt werden können und den es beim Lernen mit Temporaler Differenz nicht gibt. Der Wert 1,5 ist willkürlich gewählt und die Versuchsgruppe ist einfach ein Test für die Auswirkung des Differenzierungsfaktors gewesen.

Die Verläufe der Gewichte in dieser Versuchgruppe zeigen, dass alle Gewichte gegen Ende der Lernphasen Werte annehmen, die deutlich den Erwartungen positiver Spielergewichte und negativer Gegnergewichte entsprechen. Die Gewichte schwanken zwar häufig, aber um kleine Beträge. Die Per Feature-Darstellung hat für einige Features exemplarisch gezeigt, dass das Lernverfahren in drei Versuchen auf ein ähnliches Ergebnis gekommen ist und der Wert Q_{6500}, der ein berechenbares Maß für die Ähnlichkeit der drei Ergebnispolynome darstellen soll, hat einen Wert von 1,153, der in keiner der Versuchsgruppen A, B und C erreicht wurde.

Die ähnlichen Ergebnispolynome erreichen in der ersten Testphase ähnliche Spielstärken um die 60 Prozent; im Detail 63,5, 63,3 und 59,1 Prozent. Eine Spielstärke von 60 Prozent, ist nicht viel, aber größer als die Spielstärken, die die Ergebnispolynome in den Versuchsgruppen A, B und C erreicht haben. Allerdings hat das Startpolynom des ersten Versuchs in der ersten Testphase eine Spielstärke von 82,9 Prozent erreicht, dass bedeutet einmal, dass sich die Spielstärke durch das Lernverfahren verschlechtern kann und dass bedeutet auch, dass es Gewichte gibt, die zu Bewertungsfunktionen führen, die spielstärker sind als die Bewertungsfunktionen, die das Lernverfahren hervorgebracht hat.

Im dritten Test, in dem die Spielstärken der Polynome wieder höher sind, als in der ersten Testphase, erreichen die drei Ergebnispolynome mit 85,0, 85,7 und 83,2 Prozent wieder vergleichbare Spielstärken.

Auch wenn keine gute Spielstärke gegenüber dem Zufallsspieler der ersten Testphase erreicht wurde, auch wenn es bessere Polynome mit zufällig bestimmten Gewichten gibt, auch wenn sich das Lernverfahren in einem Versuch zu einer Verschlechterung der Spielstärke geführt hat, ist erreicht worden, dass das Lernverfahren aus zufällig bestimmten Startwerten in drei Versuchen ein ähnliches Ergebnis erzeugt hat.

6 Modifikation der temporalen Differenz

6.1 Motivation

In Anhang E, der ein Beispiel für den Horizonteffekt darstellt, sind die Bewertungen für acht Spielpositionen aus der Sicht von Weiß angegeben. Nachdem Weiß einen Stein gesetzt hat, steigen die Bewertungen an: In der Spielposition in Abbildung E.1b ist die Bewertung von vorher 0,0169 auf 0,0242 angestiegen und in Abbildung E.1d von vorher 0,0182 auf 0,0228. Nachdem Weiß gesetzt hat sind die Bewertungen demnach deutlich höher als in Spielpositionen, die durch Züge von Schwarz entstehen.
Die Bewertungen dieser Spielpositionen lassen erkennen, dass die Bewertung stark davon abhängt, ob die Spielposition durch einen Zug von Schwarz oder einen Zug von Weiß entstanden ist. Vor diesem Hintergrund soll ein Blick darauf geworfen werden, welche Spielpositionen benutzt werden, um δ zu bestimmen: Othilie zieht zum Bestimmen der Differenz δ von der Bewertung einer Spielposition, die meistens durch einen Zug von Schwarz entstanden ist[14], die Bewertung einer Spielposition ab, die immer durch einen Zug von Weiß entstanden ist. D.h. in den meisten Fällen werden bei der Bestimmung von δ zwei Spielpositionen benutzt, die nicht vergleichbar sind. Die Werte δ können dadurch zu negativen Werten tendieren.
In einer Modifikation soll untersucht werden, ob es sich als vorteilhaft erweist, für die Differenz zwei Spielpositionen heranzuziehen, die durch einen Zug desselben Spielers entstanden sind.

6.2 Änderungen am Programmcode

Wenn es das Ziel ist, in den meisten Fällen für die Bestimmung von δ zwei Spielpositionen zu benutzen, die durch einen Zug desselben Spielers entstanden sind, könnte die Tiefe des Suchbaums um einen ungeraden Wert erhöht werden. Da eine Erhöhung der Suchbaum-Tiefe gleichzeitig zu einer Verbesserung des Spielbaum-Suchverfahrenn führt, wird eine

[14] Die anderen Fälle entstehen dadurch, dass Schwarz in der vorletzten Ebene des Spielbaums auf dem Pfad zum bestmöglichen Blatt aussetzen müsste.

andere Änderung vorgenommen, um die Vergleichbarkeit mit den Ergebnissen der bisher durchgeführten Versuchsgruppen zu gewährleisten: Für die Differenz wird nicht mehr die gespeicherte Spielposition des letzten Zugs der Alpha-Komponente benutzt, sondern die aktuelle Spielposition, für die das Spielbaum-Suchverfahren gestartet wird. Wenn x_b die Spielposition ist, die nach dem Spielbaum-Suchverfahren in der aktuellen Spielposition x_z die bestmögliche ist, dann wird δ wie folgt berechnet:

$$\delta = P(x_b) - P(x_z) \tag{6.1}$$

Auf diese Weise beträgt die Tiefe des Suchbaums immer noch fünf und in den meisten Fällen werden zwei Spielpositionen verglichen, die durch Züge von Schwarz entstanden sind.

Listing 6.1 stellt den veränderten Programmcode des LearningPlayers dar. Die wesentliche Änderung betrifft die Zeilen 8 und 9; auf das Speichern einer Spielposition für die nächste Bestimmung von δ kann verzichtet werden.

```
1  getNextMove(ply, othilieModel){
2
3          ergebnis = alphaBetaSearch(ply, OthilieModel, player);
4          zug = ergebnis.getSpielzug();
5          backedUpScore = ergebnis.getScore();
6          backedUpAnzZuege = ergebnis.getAnzahlZuege();
7
8          currentScore = othilieModel.score(gewichte, player);
9          currentAnzZuege = othilieModel.getAnzahlZuege();
10
11         delta = ((backedUpScore/backedUpAnzZuege)-(currentScore/currentAnzZuege));
12
13         featuresVektor = getFeatures(player);
14         for i in {1, 2, ..., gewichte.length}{
15
16                 diffFak = 1.0;
17                 if((gewichte[i] >= 0.0 and delta < 0.0)
18                         || (gewichte[i] < 0.0 and delta >= 0.0)){
19                         diffFak = differnenzierungsfaktor;
20                 }
21
22                 gewichte[i] += lernrate * delta * featuresVektor[i] * diffFak;
23         }
24
25         Normalisierung aller Gewichte auf [-1.0, 1.0];
26
27         return zug;
28 }
```

Listing 6.1: Modifizierte Bestimmung eines Zugs beim LearningPlayer

6.3 Ergebnisse der Versuchsgruppe F

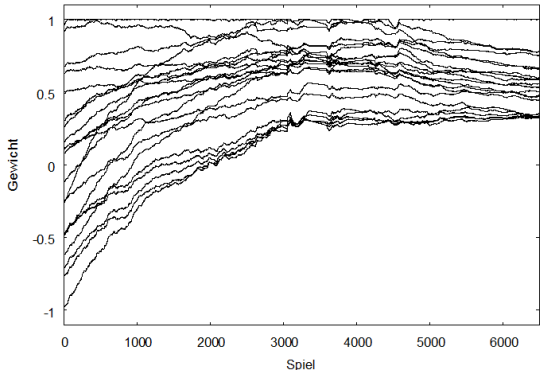
In Tabelle 6.1 sind die Parameter der Versuchsgruppe F dargestellt; es sind dieselben Parameter, die in Versuchsgruppe B benutzt worden sind. Der Grund, dass für den Lernfaktor der Wert 0,1 benutzt wird ist wieder, dass der Wert 1,0 nach den Ergebnissen der Versuchsgruppe A wegen der großen Schwankungen zu groß erschien und dass die Werte 0,05 und 0,01 nach den Ergebnissen der Versuchsgruppen C und D zu klein erschienen, weil sich die Gewichte in sehr großen Intervallen mindestens in einigen Versuchen sehr wenig geändert haben. Der Differenzierungsfaktor von $\gamma = 1,0$ wurde gewählt, um die Gewichtsänderungen des Lernverfahrens - soweit dies mit Parametern möglich ist - den von Sutton beschriebenen Gewichtsänderungen anzupassen. Wie in den anderen Versuchgruppen starten die Alpha- und die Beta-Komponente mit denselben zufälligen Gewichten die Lernphase. Um Zeit zu sparen, wird die zweite Testphase weggelassen.

	Versuchsgruppe F
Versuche	3
Lernspiele	6500
Lernfaktor α	0,1
Differenzierungsfaktor γ	1,0
Testdurchläufe	3
Testspiele pro Durchlauf	1000

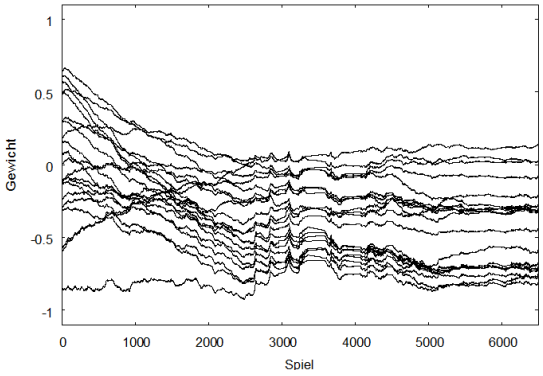
Tabelle 6.1: Übersicht über die Paramenter der Versuchsgruppe F

Die Gewichte entsprechen am Ende der jeweiligen Lernphase deutlich den intuitiven Erwartungen, dass Spielergewichte als positiv und Gegnergewichte als negativ zu bewerten seien, denn es gibt nach Tabelle B.6, in der die Gewichte der drei Ergebnispolynome dargestellt sind, und nach Abbildung 6.1 keine negative Spielergewichte und kaum positives Gegnergewicht.

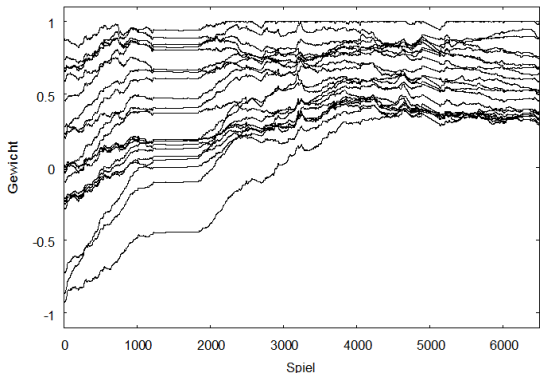
Nach den in Abbildung 6.1 dargestellten Verläufen der Gewichte in der Versuchsgruppe F haben die Gewichte der Ergebnispolynome - im Gegensatz zu den Versuchen der anderen Versuchsgruppen - größere Beträge, d.h. sie sind weiter über das Intervall [-1, 1] verteilt. Es fällt auf, dass im Gegensatz zu Versuchsgruppe B, die mit denselben Parametern gestartet wurde, in keinem Versuch Intervalle auftreten, in denen die Gewichte stark fallen. Auch nicht im dritten Versuch, in dem die Alpha-Komponente von den ungefähr letzten 250 Lernspielen kein Spiel gewinnt.



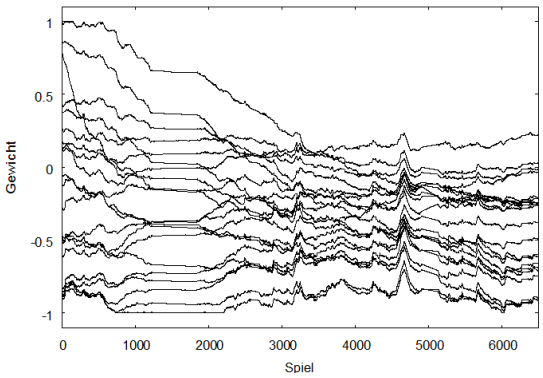
(a) Alle Spielergewichte in Versuch 1



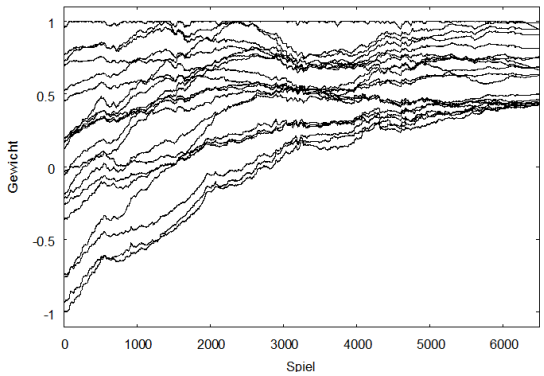
(b) Alle Gegnergewichte in Versuch 1



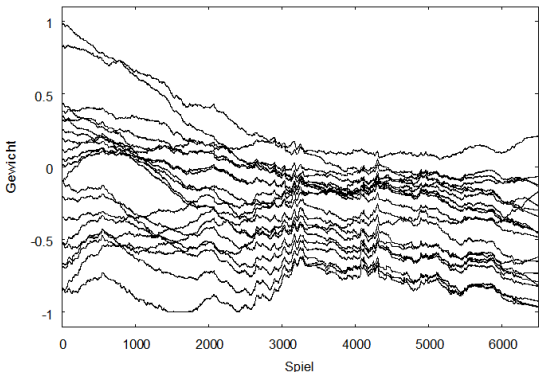
(c) Alle Spielergewichte in Versuch 2



(d) Alle Gegnergewichte in Versuch 2



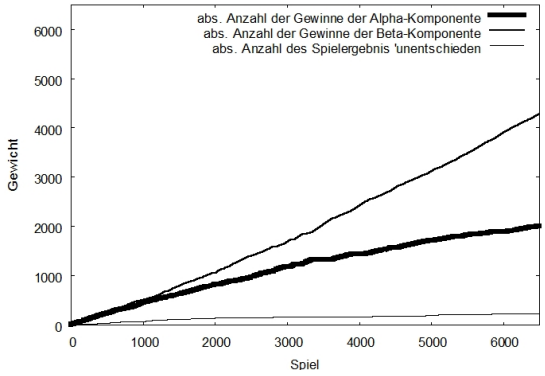
(e) Alle Spielergewichte in Versuch 3



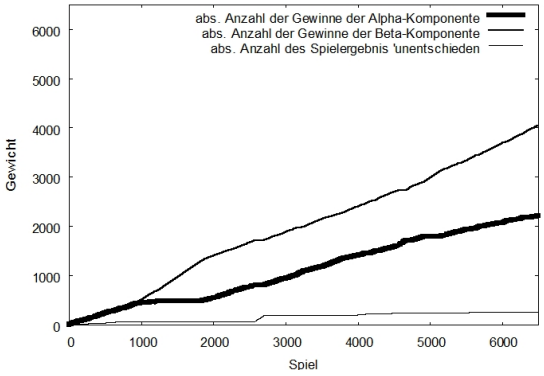
(f) Alle Gegnergewichte in Versuch 3

Abbildung 6.1: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe F

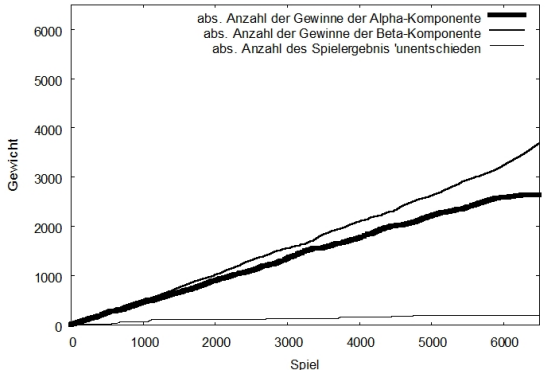
In Versuchsgruppe B wurde im zweiten Versuch beobachtet, dass die Gewinnhäufigkeit der Alpha-Komponente innerhalb der Lernphase bei nur 13,5 Prozent lag und dass zu diesem Versuch gehörende Ergebnispolynom in der ersten Testphase eine Spielstärke von nur 42,5 Prozent erreicht hat. Die Gewinnhäufigkeiten, die die Alpha-Komponente innerhalb der Lernphasen in der Versuchsgruppe F erreicht, haben dagegen mit 30,8, 34,0 und 40,5 Prozent alle einen hohen Wert.



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



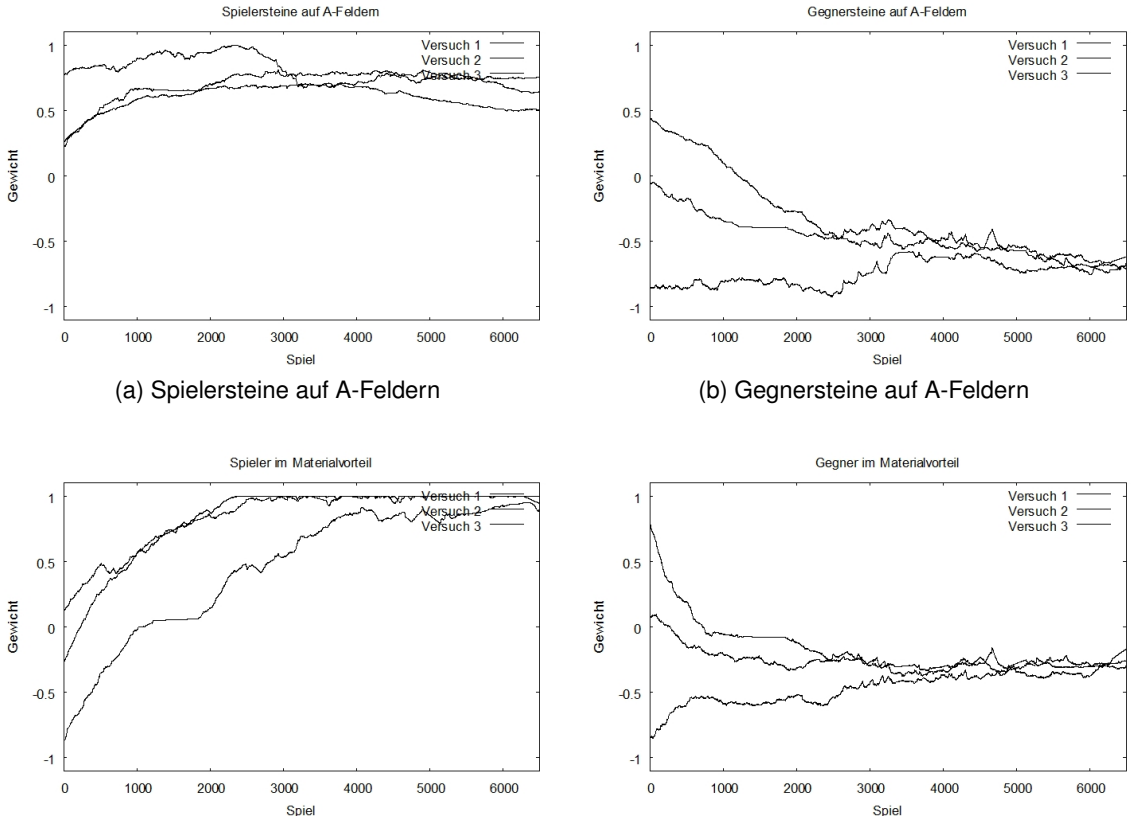
(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung 6.2: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe F

Nach der Abbildung 6.3, die exemplarisch für vier Features die Verläufe der Gewichte in den drei Versuchen darstellt, sind sich die jeweiligen Gewichte der Ergebnispolynome ähnlich geworden, wenn auch nicht so ähnlich, wie in Versuchsgruppe E, die die ähnlichsten Ergebnisse hervorgebracht hat. Die Werte Q_{6500} und Q^*_{6500} liegen in Versuchsgruppe E mit 1,153 bzw. 682,4 unter denen der Versuchsgruppe F, denn nach der Tabelle 6.2 werden nur 2,355 und 1036,0 erreicht.

Gruppe	Q_0	Q_{6500}	Q^*_{5000}	Q^*_{5500}	Q^*_{6000}
Versuchsgruppe F	15,538	2,355	3162,0	2046,0	1036,0

Tabelle 6.2: Kennzahlen mit den Standardabweichungen in Versuchsgruppe F



(a) Spielersteine auf A-Feldern

(b) Gegnersteine auf A-Feldern

(c) Spieler im Materialvorteil

(d) Gegner im Materialvorteil

Abbildung 6.3: Darstellung der Gewichte zu einigen Features in Versuchsgruppe F

Nach den Ergebnissen der ersten Testphasen, die in Tabelle 6.3 dargestellt sind, sind die Spielstärken der drei Ergebnispolynome mit 67,0, 71,8 und 66,8 Prozent etwas spielstärker,

als die Ergebnispolynome der Versuchgruppe E, die von den ersten fünf Versuchsgruppen drei ähnlich spielstarke Ergebnispolynome hervorgebracht hat, denn dort wurden die Werte 63,5, 63,3 und 59,1 Prozent gemessen. Außerdem sind die drei Spielstärken der Ergebnispolynome in Versuchgruppe F ähnlich; die größte Differenz beträgt fünf Prozent.

Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	40,5 %	43,0 %	40,7 %	41,4 %	1,1343 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	66,8 %	67,6 %	66,7 %	67,0 %	0,4028 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	60,3 %	59,6 %	56,2 %	58,7 %	1,7907 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	71,1 %	72,8 %	71,6 %	71,8 %	0,7134 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	61,4 %	64,1 %	61,9 %	62,5 %	1,1728 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	68,8 %	65,7 %	65,9 %	66,8 %	1,4166 %

Tabelle 6.3: Ergebnisse der ersten Testphasen in der Versuchsgruppe F

Die im dritten Test gegen den Zufallsspieler, der kein Spielbaum-Suchverfahren benutzt, gemessenen Spielstärken liegen wieder über den in der ersten Testphase gemessenen Spielstärken. Hier kann nur eine leichte und zum Teil auch keine signifikante Verbesserung im Vergleich mit der Versuchsgruppe E festgestellt werden: Die Ergebnispolynome erreichen in jeweils drei Durchläufen mit 1000 Testspielen mittlere Spielstärken von 86,5, 88,7 und 85,9 Prozent, wobei die jeweiligen Standardabweichungen unter 1 Prozent liegen.

Versuchsgruppe	Versuch	Startpolynom	Ergebnispolynom
F	1	62,3	86,5
F	2	78,4	88,7
F	3	83,6	85,9

Tabelle 6.4: Ergebnisse des dritten Tests in Versuchsgruppe F

6.4 Fazit

Nach einer Modifikation, nach der für die Bestimmung von δ zwei Spielpositionen benutzt werden, die meistens durch Züge von Schwarz entstanden sind, ließen sich in Versuchsgruppe F verbesserte Ergebnisse beobachten, wobei die Verbesserungen nicht auf eine Vergrößerung der Suchbaum-Tiefe zurückzuführen ist, da diese unverändert ist. Die wesentlichen Verbesserungen sind, dass

- es trotz eines Intervalls innerhalb einer Lernphase, in dem die Alpha-Komponente nicht gewinnt, die Gewichte nicht stark fallen,
- Spielergewichte am Ende der Lernphase immer positiv und Gegnergewichte meist negativ sind und dass
- die Ergebnispolynome in der ersten Testphase Spielstärken von ungefähr 65 bis 70 Prozent erreichen.

Diese Ergebnisse zeigen, dass die Wahl der Spielpositionen, mit den δ bestimmt wird, entscheidend ist. Es ist erfolgreicher, zwei Spielpositionen zu benutzen, die in den meisten Fällen durch denselben Spieler entstanden sind.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Othilies Lernverfahren ist inspiriert von Samuels Dame-Programm, das die Gewichte für eine Bewertungsfunktion lernt, die die Form eines linearen Polynoms mit mehreren Unbekannten hat, wobei die Werte der Unbekannten durch die Ausprägung bestimmter Features der Spielpositionen bestimmt werden. Von Samuel übernommen wurde der Selfplay-Modus, bei dem eine Alpha-Komponente, die versucht die Gewichte zu lernen, gegen eine Beta-Komponente spielt, die die aktuellen Gewichte von der Alpha-Komponente bekommt, wenn die Alpha-Komponente zehn Lernspiele gewonnen hat. Übernommen wurde außerdem die Idee, aus den Bewertungen von zwei Spielpositionen eine Differenz δ zu berechnen und als Basis für die Gewichtsänderungen zu benutzen sowie Samuels grundlegenden Aussagen, wie die Gewichte zu ändern sein (vgl. Tabelle 4.1).
Die Gewichtsänderungen, die von Othilies Lernverfahren vorgenommen werden, sind nicht von Samuel übernommen worden, sondern an die Gewichtsänderungen des Lernens mit Temporaler Differenz angelehnt, für die Sutton und Dyan die Konvergenz gegen die optimalen Werte bewiesen haben. Diese Gewichtsänderungen von Othilie bewirken - wie die Gewichtsänderungen von Sutton - die Änderung der Gewichte in Richtung der späteren Bewertung und setzen damit Samuels Überlegungen aus Tabelle 4.1 um. Die Unterschiede zu Suttons Lernen mit Temporaler Differenz sind,

- dass relative Bewertungen für die Bestimmung der Differenz benutzt werden, indem die Bewertungen jeweils durch die Anzahl der Steine geteilt werden, die auf dem Spielfeld liegen,
- dass die Gewichtsänderungen sofort nach der Zugbestimmung vorgenommen werden und nicht am Ende des Spiels,
- dass es einen Differenzierungsfaktor gibt und
- dass für die Differenz nicht zwei aufeinander folgende, beobachtete Spielpositionen benutzt werden, sondern zwei Spielpositionen, von denen nur eine beim System beobachtet wird, während die zweite durch Vorausschau mittels eines Spielbaum-Suchverfahrens bestimmt wird.

Vor der Modifikation wird δ durch

$$\delta = P_z(x_b) - P_z(x_{z-1}) \tag{7.1}$$

und nach der Modifikation durch

$$\delta = P_z(x_b) - P_z(x_z) \tag{7.2}$$

bestimmt, so dass nach der Modifikation in den meisten Fällen zwei Spielpositionen verglichen werden, die durch Züge von Schwarz entstanden sind.

Vor der Modifikation ist es nicht gelungen bei einem Differenzierungsfaktor von $\gamma = 1,0$ in den Versuchsgruppen A, B, C und D jeweils drei ähnliche Ergebnisse zu erzeugen; ähnliche Ergebnisse wurden nur in der Versuchsgruppe E, die mit einem Differenzierungsfaktor von $\gamma = 1,5$ gestartet wurde, erzielt. Die Ergebnispolynome der Versuchsgruppe E erreichen in der ersten Testphase mittlere Spielstärken von ungefähr 60 Prozent.

Nach der Modifikation ist es mit der Versuchsgruppe F gelungen, ähnliche Ergebnisse zu erzielen, allerdings sind die Ergebnisse nicht so ähnlich, wie in Versuchsgruppe E. Dafür liegen die gemessenen Spielstärken in der Versuchsgruppe F leicht über denen der Versuchsgruppe E: In der ersten Testphase wurden Werte von ungefähr 65 bis 70 Prozent gemessen.

Diese Ergebnisse nach der Modifikation zeigen, dass es erfolgreicher ist, zwei Spielpositionen für die Bestimmung der temporalen Differenz zu benutzen, die in den meisten Fällen durch denselben Spieler entstanden sind.

Die Ergebnisse der ersten Testphase bedeuten, dass die Ergebnispolynome dem Zufallsspieler der ersten Testphase nicht deutlich überlegen sind, wobei dieser Zufallsspieler ein Spielbaum-Suchverfahren benutzt, jedoch bei jedem Zug mit neuen, zufälligen Gewichten. Dieser Zufallsspieler darf nicht mit einem Spieler verwechselt werden, der jedes Mal irgendeinen Zug aus der Menge der gültigen Züge wählt. In einem dritten Test, bei dem der Zufallsspieler jedes Mal irgendeinen Zug auswählt, erreichen die Ergebnispolynome der Versuchsgruppe E mittlere Spielstärken von 85,0, 85,7 und 83,2 Prozent und die Ergebnispolynome der Versuchsgruppe F erreichen 86,5, 88,7 und 85,9 Prozent. D.h. bei Othello und den verwendeten Features für eine Bewertungsfunktion ist der Einsatz eines Spielbaum-Suchverfahrens dem zufälligen Bestimmen eines Zugs deutlich überlegen.

7.2 Kritische Betrachtungen und Ausblick

Die ersten Testphasen der Versuchsgruppen E und F haben ergeben, dass mit einer Gewinnhäufigkeit von ungefähr 60 Prozent bzw. 65 bis 70 Prozent keine besonders spielstar-

ken Ergebnispolynome erzeugt wurden und dass es zu den verwendeten Features Gewichte gibt, die zu spielstärkeren Polynomen führen[15].
Vor diesem Hintergrund soll im Folgenden ein Blick auf verschiedene Aspekte geworfen werden: Eine verbesserte Bewertungsfunktion, die Verwendung einer anderen Tiefe beim Spielbaum-Suchverfahren, die verwendeten relativen Bewertungen und der Nutzen eines kleinen Lernfaktors.

7.2.1 Verbesserte Bewertungsfunktion

Eine verbesserte Bewertungsfunktion ist nicht nur eine Frage bei der Verbesserung der Spielstärke bei einem guten Lernverfahren, sondern auch eine Frage bei der Verbesserung des Lernverfahrens: Othilies Lernverfahren betrachtet zwei Spielpositionen und ändert die Gewichte in Richtung der späteren Spielposition. Dahinter steht die Idee, dass die Bewertung der späteren Spielposition die bessere Voraussage auf das Spielergebnis macht. Probleme können durch den Horizonteffekt entstehen, wenn die verwendete Bewertungsfunktion für die Spielpositionen keine guten Voraussagen auf das Spielergebnis macht, weil ein schlechter Verlauf des Spiels erst in späteren Spielpositionen gemessen wird. In Anhang E ist der Horizonteffekt beispielhaft für eine Othello-Spielposition dargestellt. In diesem Beispiel, das mit einer von Rose (2005) übernommenen Spielposition beginnt, ist dargestellt, wie Schwarz aus einer Spielposition, in der Weiß deutlich führt, das Spiel in einen deutlichen Gewinn umwandeln kann, ohne dass Weiß die Möglichkeit hat, sich dagegen zu wehren. Alle acht Spielpositionen werden aus der Sicht von Weiß mit dem Ergebnispolynom des ersten Versuchs der Versuchsgruppe E bewertet und die Bewertungen zeigen, dass Weiß mit dieser Bewertungsfunktion, die Bedrohung in den Spielpositionen erst spät messen kann.
Das Problem ist, dass die verwendete Bewertungsfunktion auf der Anzahl an Steinen beruht, die Spieler und Gegner auf dem Spielfeld haben, und dass Weiß am Anfang deutlich mehr Steine hat. Die Bewertungsfunktion misst nicht, dass Weiß aus strategischen Gesichtspunkten schon verloren hat, wenn Schwarz richtig spielt. Eine Erweiterung der Feature-Menge um strategische Features, die strategische Bedrohungen in den Spielpositionen messen, könnte die Bewertungsfunktion verbessern, indem sie den Horizonteffekt mildert.
Aus den Aspekten die Rose (2005) zu zahlreichen Othello-Strategien beschreibt, spielen folgende Aspekte in dem Beispiel in Anhang E eine Rolle: Weiß hat nur wenig Züge und diese sind ungünstig für Weiß, zwei Ecken werden von Schwarz attackiert und Weiß hat zwar am Anfang viele Steine, aber es handelt sich nicht um stabile Steine im Sinne von Rose (2005), denn Weiß verliert davon bis zum Ende des Spiels viele.

[15] Das Startpolynom des ersten Versuchs der Versuchsgruppe E hat im Mittel 82,9 Prozent der Testspiele in der ersten Testphase gewonnen.

7.2.2 Tiefe des Spielbaum-Suchverfahrens

Eine andere Möglichkeit, den Horizonteffekt zu mildern, besteht in der Steigerung der Tiefe des Suchbaums. Bei den in Kapitel 5 beschriebenen Versuchen wurde von allen Spielern beim Lernen und beim Testen eine feste Suchtiefe von fünf verwendet. Um die Voraussagen beim Lernen zu verbessern, könnte versucht werden, diesen Wert hochzusetzen. Da die Anzahl der Blätter im Suchbaum mit der Tiefe exponentiell wächst ist die Tiefe nur begrenzt steigerbar. Eine Alternative ist, nur ruhige Stellungen zu bewertetn. D.h. repräsentiert ein Blatt im unvollständigen Suchbaum keine ruhige Stellung, wird der Suchbaum in diesem Blatt weiter aufgebaut. Beispielsweise könnte eine Othello-Spielposition, in der eine Ecke attackiert wird, als unruhig erkannt und der Suchbaum in diesem Blatt weiter aufgebaut werden, bis die Ecke eingenommen oder die Attacke abgewehrt ist.

7.2.3 Relative Bewertungen

Othilies Lernverfahren ist eine Modifikation des Lernens mit Temporaler Differenz, für das Sutton und Dyan die Konvergenz gegen optimale Werte gezeigt haben, und Othilies Lernverfahren erzeugt keine optimalen Gewichte. Um Othilies Qualität zu verbessern, könnten die in Kapitel 7.1 aufgezählten Unterschiede ganz oder teilweise beseitigt werden. Besonderes Augenmerk könnte auf den Punkt der relativen Bewertungen gelegt werden: Bei Bestimmung von δ werden die beiden Bewertungen durch die Anzahl der Steine geteilt, die jeweils auf dem Spielfeld liegen. Formal gezeigt wurde von Sutton, dass Gewichtsänderungen nach der Formel 2.12 optimale Gewichte für die Bewertungsfunktion P erzeugen und Othilies Lerverfahren benutzt bei einem Differenzierungsfaktor von $\gamma = 1,0$ Gewichtsänderungen nach

$$\Delta w_z = \alpha \cdot (\frac{P_b}{\tau_b} - \frac{P_{z-1}}{\tau_{z-1}}) \cdot \nabla_w P_z \tag{7.3}$$

wobei τ_i die Anzahl der Steine beschreibt, die jeweils auf dem Spielfeld liegen[16]. Durch die Definition $P_{rel} := \frac{P}{\tau_i}$ kann diese Formel zwar theoretisch Suttons Formel für die Gewichtsänderungen genähert werden, allerdings erzeugt Othilies Lernverfahren damit optimale Gewichte für die Bewertungsfunktion P_{rel}, während die Zugbestimmung in den Lern- und Testphasen P und nicht P_{rel} benutzt.

Die Motivation für Benutzung relativer Bewertungen war, dass positive δ vermieden werden sollten, die sich lediglich aus dem Grund ergeben, dass bei der späteren Spielposition mehr Steine auf dem Spielfeld liegen. Da im Verlauf eines Spiels generell mehr Steine auf dem

[16] Nach der Modifikation der temporalen Differenz ergibt sich eine leicht veränderte Formel: statt z-1 wird z benutzt.

Spielfeld liegen, können die relativen Bewertungen dazu führen, dass sich die Gewichtsänderungen im Verlauf des Spiels verkleinern, d.h. je näher das Spiel dem Ende ist, umso kleiner sind die Beträge der Gewichtsänderungen. Dieser Effekt könnte eintreten, wenn sich bei negativen Gegnergewichten und positiven Spielergewichten die Beträge der entsprechenden Terme aufheben und so der Effekt der ungewollten positiven δ überhaupt nicht eintritt.

7.2.4 Kleiner Lernfaktor

Der erste Versuch der Versuchsgruppe D mit dem kleinen Lernfaktor von $\alpha = 0,01$ hat gezeigt, dass durch kleine Veränderungen der Gewichte aus einem mittelstarken Polynom ein Polynom erzeugt werden kann, das in der ersten Testphase eine Spielstärke von ungefähr 70 Prozent erreicht. Es stellt sich daher die Frage, ob dieses Ergebnispolynom oder die drei Ergebnispolynome der Versuchsgruppe E durch eine weitere Lernphase dieser Art weiter verbessert werden können, indem die Gewichte sozusagen feinjustiert werden. Alternativ stellt sich die Frage, ob ein kleiner Lernfaktor generell sinnvoll ist und ähnliche Ergebnisse dadurch erzielt werden können, dass die Lernphase deutlich vergrößert wird.

A Klassendiagramm zur grafischen Oberfläche

Das folgende Klassendiagramm veranschaulicht, wie das Observer-Pattern in der grafischen Oberfläche umgesetzt wurde:

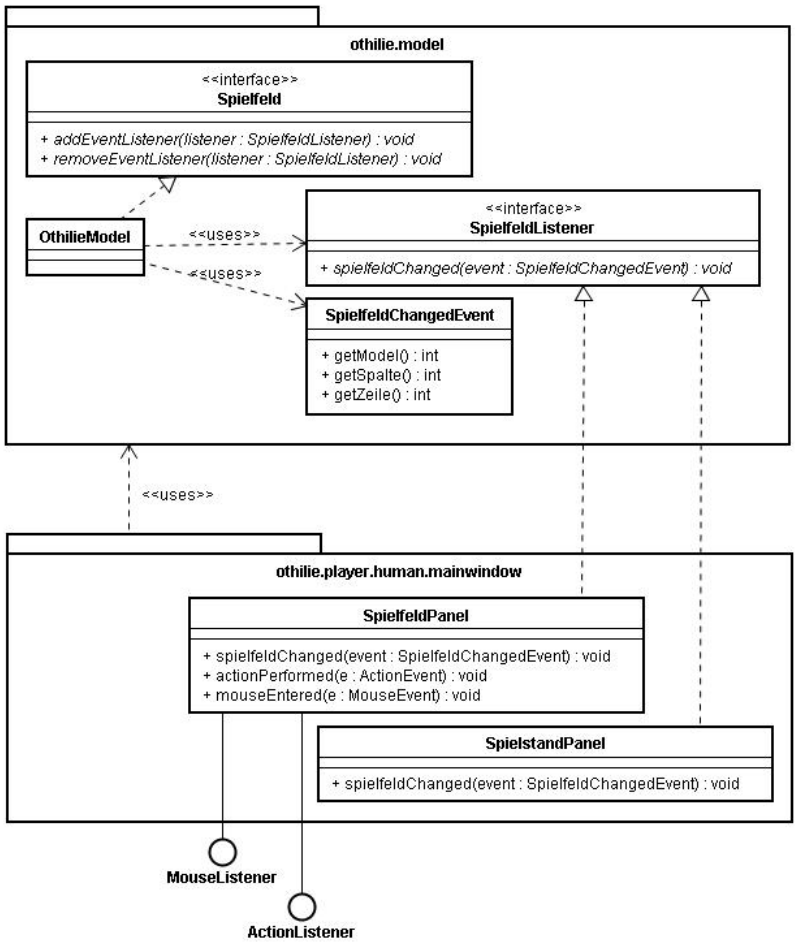


Abbildung A.1: Klassendiagramm der grafischen Oberfläche zum Spielen

Aus Gründen der Übersichtlichkeit sind die einzelnen uses-Beziehungen von den beiden Observer-Klassen im Package othilie.player.human.mainwindow zu den Elementen im Package othilie.model weggelassen und durch eine uses-Beziehung zwischen den beiden Packages ersetzt worden. Tatsächlich ist es so, dass die beiden Observer-Klassen alle Elemente im Package othilie.model benutzen.

Die Anordnung der Klassen und Interfaces in den beiden Packages ist so gewählt, dass es keine wechselseitigen uses-Beziehungen zwischen den beiden Packages gibt; die Elemente im Package othilie.model kennen die Observer-Klassen nicht.

B Gewichte der Ergebnispolynome

B.1 Ergebnispolynome in Versuchsgruppe A

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
1	Spielersteine auf A-Feldern	0,3084	-0,1967	-0,0819	0,009956	0,2162
3	Spielersteine auf C-Feldern	0,3412	-0,1293	-0,0237	0,06273	0,2016
4	Spielersteine auf D-Feldern	0,3027	-0,1305	-0,1383	0,01129	0,2061
5	Spielersteine auf E-Feldern	0,342	-0,08317	-0,2189	0,0133	0,2389
6	Spielersteine auf F-Feldern	0,3162	-0,06668	-0,078	0,05717	0,1832
7	Spielersteine auf G-Feldern	0,1737	-0,1184	-0,1007	-0,01512	0,1337
8	Spielersteine auf H-Feldern	0,243	-0,2376	-0,02139	-0,005318	0,1965
9	Spielersteine auf I-Feldern	0,3206	-0,09668	-0,04862	0,05843	0,1864
10	Spielersteine auf J-Feldern	0,2258	-0,1646	-0,07276	-0,003848	0,1666
11	Spielersteine auf K-Feldern	0,2506	-0,1776	-0,1081	-0,01172	0,1876
12	Spielersteine auf L-Feldern	0,108	-0,3045	-0,1609	-0,1191	0,171
13	Spielersteine auf M-Feldern	0,3924	-0,1391	-0,04676	0,06884	0,2319
14	Spielersteine auf N-Feldern	0,3149	-0,2104	-0,1408	-0,01208	0,233
15	Spielersteine auf O-Feldern	0,1627	-0,2572	-0,1977	-0,09741	0,1855
16	Spielersteine auf P-Feldern	0,01918	-0,4013	-0,1565	-0,1795	0,1724
17	Gegnersteine auf A-Feldern	-0,4538	-0,2471	-0,6561	-0,4523	0,167
18	Gegnersteine auf B-Feldern	-0,4061	-0,4177	-0,5668	-0,4636	0,07319
19	Gegnersteine auf C-Feldern	-0,4022	-0,4515	-0,5972	-0,4836	0,0828
20	Gegnersteine auf D-Feldern	-0,3732	-0,5156	-0,5781	-0,489	0,08571
21	Gegnersteine auf E-Feldern	-0,4462	-0,3979	-0,6371	-0,4937	0,1033
22	Gegnersteine auf F-Feldern	-0,4581	-0,4773	-0,6723	-0,5359	0,09676
23	Gegnersteine auf G-Feldern	-0,2675	-0,4052	-0,6073	-0,4267	0,1395
24	Gegnersteine auf H-Feldern	-0,2928	-0,445	-0,6717	-0,4698	0,1557
25	Gegnersteine auf I-Feldern	-0,4591	-0,4755	-0,6554	-0,53	0,08891
26	Gegnersteine auf J-Feldern	-0,3732	-0,446	-0,6754	-0,4982	0,1288
27	Gegnersteine auf K-Feldern	-0,4877	-0,5291	-0,6744	-0,5638	0,08006
28	Gegnersteine auf L-Feldern	-0,3198	-0,4087	-0,6156	-0,4481	0,1239
29	Gegnersteine auf M-Feldern	-0,5104	-0,5887	-0,6788	-0,5926	0,06879
30	Gegnersteine auf N-Feldern	-0,4648	-0,5496	-0,5838	-0,5327	0,04999
31	Gegnersteine auf O-Feldern	-0,3733	-0,4733	-0,5707	-0,4725	0,08058

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
32	Gegnersteine auf P-Feldern	-0,5666	-0,591	-0,7805	-0,646	0,09558
33	Spielersteine im Ring 0	0,3395	-0,1179	-0,08301	0,04618	0,2079
34	Gegnersteine im Ring 0	-0,4261	-0,4748	-0,6128	-0,5045	0,0791
35	Spielersteine im Ring 1	0,2547	-0,1595	-0,08273	0,004159	0,1799
36	Gegnersteine im Ring 1	-0,3713	-0,4646	-0,6421	-0,4927	0,1123
37	Spielersteine im Ring 2	0,1738	-0,2464	-0,1556	-0,07608	0,1805
38	Gegnersteine im Ring 2	-0,3936	-0,4704	-0,6203	-0,4948	0,09411
39	Spielersteine	0,2619	-0,1727	-0,1011	-0,003983	0,1903
40	Gegnersteine	-0,4117	-0,478	-0,6338	-0,5078	0,09312
41	Spieler im Materialvorteil	0,4276	0,007298	0,06267	0,1658	0,1864
42	Gegner im Materialvorteil	-0,9932	-0,9996	-0,9996	-0,9975	0,003033

B.2 Ergebnispolynome in Versuchsgruppe B

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
1	Spielersteine auf A-Feldern	0,2069	-0,0544	0,03077	0,06109	0,1088
2	Spielersteine auf B-Feldern	0,2523	0,1389	-0,002053	0,1297	0,104
3	Spielersteine auf C-Feldern	0,2696	0,07704	-0,069	0,09254	0,1387
4	Spielersteine auf D-Feldern	0,2709	0,2906	-0,1879	0,1246	0,2211
5	Spielersteine auf E-Feldern	0,265	0,3174	0,0276	0,2033	0,1261
6	Spielersteine auf F-Feldern	0,2691	0,1802	0,02556	0,1583	0,1006
7	Spielersteine auf G-Feldern	0,1631	0,1533	-0,1093	0,06905	0,1262
8	Spielersteine auf H-Feldern	0,174	0,1484	-0,1077	0,07156	0,1272
9	Spielersteine auf I-Feldern	0,3254	0,205	0,04954	0,1933	0,1129
10	Spielersteine auf J-Feldern	0,2066	0,2099	-0,02359	0,131	0,1093
11	Spielersteine auf K-Feldern	0,187	0,1074	-0,07874	0,07189	0,1114
12	Spielersteine auf L-Feldern	0,1048	0,02039	-0,1716	-0,01546	0,1157
13	Spielersteine auf M-Feldern	0,3223	0,3036	0,06295	0,2296	0,1181
14	Spielersteine auf N-Feldern	0,1983	0,1763	-0,04465	0,11	0,1097
15	Spielersteine auf O-Feldern	0,1228	-0,1592	-0,1838	-0,0734	0,1391
16	Spielersteine auf P-Feldern	-0,04825	-0,05217	-0,09712	-0,06585	0,02217
17	Gegnersteine auf A-Feldern	-0,324	-0,449	-0,6947	-0,4892	0,154
18	Gegnersteine auf B-Feldern	-0,3675	-0,6265	-0,6891	-0,561	0,1392
19	Gegnersteine auf C-Feldern	-0,3776	-0,4088	-0,7164	-0,5009	0,1529
20	Gegnersteine auf D-Feldern	-0,3722	-0,5937	-0,5824	-0,5161	0,1019
21	Gegnersteine auf E-Feldern	-0,3511	-0,6308	-0,728	-0,57	0,1598
22	Gegnersteine auf F-Feldern	-0,3588	-0,6571	-0,815	-0,6103	0,1892
23	Gegnersteine auf G-Feldern	-0,3285	-0,6306	-0,68	-0,5464	0,1554
24	Gegnersteine auf H-Feldern	-0,3046	-0,3585	-0,6932	-0,4521	0,1719
25	Gegnersteine auf I-Feldern	-0,387	-0,5911	-0,8299	-0,6027	0,181

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
26	Gegnersteine auf J-Feldern	-0,3154	-0,5113	-0,7778	-0,5348	0,1895
27	Gegnersteine auf K-Feldern	-0,3172	-0,4423	-0,7744	-0,5113	0,1929
28	Gegnersteine auf L-Feldern	-0,3368	-0,4161	-0,6498	-0,4675	0,1329
29	Gegnersteine auf M-Feldern	-0,4135	-0,7834	-0,86	-0,6856	0,1949
30	Gegnersteine auf N-Feldern	-0,3109	-0,5146	-0,7676	-0,531	0,1868
31	Gegnersteine auf O-Feldern	-0,3326	-0,2554	-0,6351	-0,4077	0,1638
32	Gegnersteine auf P-Feldern	-0,4714	-0,3648	-0,7564	-0,5308	0,1653
33	Spielersteine im Ring 0	0,2723	0,167	-0,01418	0,1417	0,1183
34	Gegnersteine im Ring 0	-0,3704	-0,584	-0,7313	-0,5619	0,1482
35	Spielersteine im Ring 1	0,2035	0,1654	-0,05207	0,1056	0,1126
36	Gegnersteine im Ring 1	-0,3256	-0,513	-0,7469	-0,5285	0,1723
37	Spielersteine im Ring 2	0,1351	0,009159	-0,145	-2,416E-4	0,1146
38	Gegnersteine im Ring 2	-0,3273	-0,3815	-0,6866	-0,4651	0,1582
39	Spielersteine	0,2051	0,126	-0,05537	0,09192	0,109
40	Gegnersteine	-0,3511	-0,518	-0,7291	-0,5327	0,1547
41	Spieler im Materialvorteil	0,4742	0,3511	0,1455	0,3236	0,1356
42	Gegner im Materialvorteil	-0,9993	-0,7974	-1,0	-0,9323	0,09533

B.3 Ergebnispolynome in Versuchsgruppe C

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
1	Spielersteine auf A-Feldern	0,2345	0,1672	0,5668	0,3228	0,1747
2	Spielersteine auf B-Feldern	0,226	-0,003799	0,3343	0,1855	0,141
3	Spielersteine auf C-Feldern	0,2462	0,1464	0,284	0,2256	0,05806
4	Spielersteine auf D-Feldern	0,2271	0,4065	-0,127	0,1689	0,2217
5	Spielersteine auf E-Feldern	0,3655	0,3782	0,9644	0,5694	0,2794
6	Spielersteine auf F-Feldern	0,2594	-0,1864	-0,5368	-0,1546	0,3258
7	Spielersteine auf G-Feldern	0,1605	0,08319	0,07201	0,1052	0,03936
8	Spielersteine auf H-Feldern	0,06253	-0,1167	-0,4238	-0,1593	0,2008
9	Spielersteine auf I-Feldern	0,3636	-0,142	-0,4187	-0,06566	0,3239
10	Spielersteine auf J-Feldern	0,2288	0,3497	0,5699	0,3828	0,1412
11	Spielersteine auf K-Feldern	0,1865	0,112	0,5497	0,2828	0,1912
12	Spielersteine auf L-Feldern	0,0995	-0,1033	0,2085	0,06822	0,1292
13	Spielersteine auf M-Feldern	0,3895	0,3015	-0,5412	0,04995	0,4195
14	Spielersteine auf N-Feldern	0,2561	0,3173	0,4236	0,3323	0,06919
15	Spielersteine auf O-Feldern	0,01439	0,1699	-0,3349	-0,0502	0,2111
16	Spielersteine auf P-Feldern	0,06886	-0,1248	0,352	0,09867	0,1958
17	Gegnersteine auf A-Feldern	-0,3282	-0,1209	-0,5543	-0,3345	0,177
18	Gegnersteine auf B-Feldern	-0,2651	-0,289	-0,5164	-0,3568	0,1132
19	Gegnersteine auf C-Feldern	-0,2968	-0,4748	-0,8354	-0,5356	0,224

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
20	Gegnersteine auf D-Feldern	-0,2888	-0,6778	0,1215	-0,2817	0,3263
21	Gegnersteine auf E-Feldern	-0,3946	-0,6028	-0,1266	-0,3747	0,1949
22	Gegnersteine auf F-Feldern	-0,316	-0,2862	-0,7045	-0,4356	0,1906
23	Gegnersteine auf G-Feldern	-0,1992	-0,3361	-0,9558	-0,497	0,3292
24	Gegnersteine auf H-Feldern	-0,1505	-0,1958	-0,9596	-0,4353	0,3712
25	Gegnersteine auf I-Feldern	-0,4027	-0,4245	0,7314	-0,03196	0,5398
26	Gegnersteine auf J-Feldern	-0,2701	-0,2793	0,9538	0,1348	0,5791
27	Gegnersteine auf K-Feldern	-0,192	-0,2748	-0,5148	-0,3272	0,1369
28	Gegnersteine auf L-Feldern	-0,2993	-0,4982	-0,6622	-0,4866	0,1484
29	Gegnersteine auf M-Feldern	-0,4263	-0,3235	-0,5958	-0,4485	0,1122
30	Gegnersteine auf N-Feldern	-0,2628	-0,1534	-0,328	-0,2481	0,07208
31	Gegnersteine auf O-Feldern	-0,2198	-0,3537	0,1047	-0,1562	0,1925
32	Gegnersteine auf P-Feldern	-0,5294	-0,3046	-0,5336	-0,4559	0,107
33	Spielersteine im Ring 0	0,3015	-0,001035	0,2039	0,1681	0,1261
34	Gegnersteine im Ring 0	-0,3286	-0,4826	-0,1226	-0,3113	0,1475
35	Spielersteine im Ring 1	0,165	0,3098	-0,6897	-0,07163	0,441
36	Gegnersteine im Ring 1	-0,2363	-0,07183	-0,804	-0,3707	0,3137
37	Spielersteine im Ring 2	0,08553	-0,09248	0,6886	0,2272	0,3342
38	Gegnersteine im Ring 2	-0,271	-0,2332	0,6628	0,05283	0,4316
39	Spielersteine	0,2183	0,05921	-0,367	-0,02981	0,2471
40	Gegnersteine	-0,3076	-0,4954	-0,8554	-0,5528	0,2273
41	Spieler im Materialvorteil	0,4865	0,1415	0,7365	0,4548	0,244
42	Gegner im Materialvorteil	-1,0	-0,9998	-0,3346	-0,7781	0,3136

B.4 Ergebnispolynome in Versuchsgruppe D

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
1	Spielersteine auf A-Feldern	0,07902	-0,7205	0,4593	-0,06073	0,4917
2	Spielersteine auf B-Feldern	0,4033	-0,4278	-0,1976	-0,07399	0,3504
3	Spielersteine auf C-Feldern	0,3481	0,5319	0,04152	0,3072	0,2023
4	Spielersteine auf D-Feldern	0,4575	0,4888	0,3413	0,4292	0,0635
5	Spielersteine auf E-Feldern	0,9228	0,4224	0,4787	0,608	0,2238
6	Spielersteine auf F-Feldern	0,08056	0,2384	0,3668	0,2286	0,1171
7	Spielersteine auf G-Feldern	0,04087	-0,5917	0,4122	-0,04619	0,4144
8	Spielersteine auf H-Feldern	0,1493	-0,1691	-0,7858	-0,2685	0,3882
9	Spielersteine auf I-Feldern	-0,4703	0,3575	0,1016	-0,003755	0,3461
10	Spielersteine auf J-Feldern	0,05552	-0,3905	0,8105	0,1585	0,4957
11	Spielersteine auf K-Feldern	0,5665	-0,2793	-0,929	-0,2139	0,6123
12	Spielersteine auf L-Feldern	-0,6831	-0,5324	0,3047	-0,3036	0,4345
13	Spielersteine auf M-Feldern	0,4019	-0,9336	0,579	0,01576	0,6752

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
14	Spielersteine auf N-Feldern	0,2992	0,516	-0,7987	0,005497	0,5755
15	Spielersteine auf O-Feldern	0,3419	-0,7205	0,05567	-0,1076	0,4489
16	Spielersteine auf P-Feldern	0,4353	0,4023	0,09338	0,3103	0,154
17	Gegnersteine auf A-Feldern	-0,3778	0,4377	0,4291	0,163	0,3824
18	Gegnersteine auf B-Feldern	-0,8376	-0,9999	-0,91	-0,9158	0,06638
19	Gegnersteine auf C-Feldern	-0,2571	0,5216	0,4541	0,2396	0,3522
20	Gegnersteine auf D-Feldern	-0,087	0,6489	-0,8129	-0,08365	0,5968
21	Gegnersteine auf E-Feldern	-0,9533	0,2931	0,5198	-0,04678	0,6476
22	Gegnersteine auf F-Feldern	0,3801	0,3292	0,2024	0,3039	0,07475
23	Gegnersteine auf G-Feldern	-0,7072	0,2795	-0,7335	-0,3871	0,4714
24	Gegnersteine auf H-Feldern	0,4325	-0,6535	-0,08473	-0,1019	0,4435
25	Gegnersteine auf I-Feldern	0,45	0,7224	-0,2992	0,2911	0,4319
26	Gegnersteine auf J-Feldern	-0,2814	-0,3908	-0,08895	-0,2537	0,1248
27	Gegnersteine auf K-Feldern	0,6414	-0,4516	0,1237	0,1045	0,4464
28	Gegnersteine auf L-Feldern	0,6254	-0,3224	-0,4133	-0,03677	0,4697
29	Gegnersteine auf M-Feldern	0,1103	-0,3141	-1,0	-0,4013	0,4574
30	Gegnersteine auf N-Feldern	-0,4441	0,4918	-0,6763	-0,2096	0,5049
31	Gegnersteine auf O-Feldern	0,1621	-0,968	-0,06732	-0,2911	0,4877
32	Gegnersteine auf P-Feldern	0,34	0,6681	0,3002	0,4361	0,1648
33	Spielersteine im Ring 0	1,0	-0,7704	-0,7801	-0,1835	0,8369
34	Gegnersteine im Ring 0	0,1008	-0,7677	-0,3241	-0,3303	0,3546
35	Spielersteine im Ring 1	0,05486	-0,7832	-0,068	-0,2655	0,3695
36	Gegnersteine im Ring 1	0,2633	0,158	-0,8295	-0,1361	0,4922
37	Spielersteine im Ring 2	0,6956	0,6801	0,3828	0,5862	0,1439
38	Gegnersteine im Ring 2	-0,4753	0,5351	0,03543	0,03174	0,4125
39	Spielersteine	-0,5743	-0,3492	-0,5702	-0,4979	0,1051
40	Gegnersteine	0,1515	0,7373	0,603	0,4973	0,2506
41	Spieler im Materialvorteil	0,05827	0,3563	0,421	0,2785	0,158
42	Gegner im Materialvorteil	-0,8999	0,8341	-0,05311	-0,03962	0,708

B.5 Ergebnispolynome in Versuchsgruppe E

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
1	Spielersteine auf A-Feldern	0,2033	0,237	0,1952	0,2118	0,01808
2	Spielersteine auf B-Feldern	0,2409	0,2215	0,2245	0,2289	0,008542
3	Spielersteine auf C-Feldern	0,2444	0,3072	0,1974	0,2497	0,04497
4	Spielersteine auf D-Feldern	0,2274	0,2888	0,2014	0,2392	0,03667
5	Spielersteine auf E-Feldern	0,3145	0,2817	0,313	0,3031	0,01511
6	Spielersteine auf F-Feldern	0,2716	0,1956	0,2551	0,2408	0,03267
7	Spielersteine auf G-Feldern	0,0233	0,1363	0,1144	0,09135	0,04894

Nr.	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ
8	Spielersteine auf H-Feldern	0,003071	0,1005	0,0471	0,05022	0,03983
9	Spielersteine auf I-Feldern	0,3945	0,4212	0,3421	0,3859	0,03289
10	Spielersteine auf J-Feldern	0,1932	0,1363	0,2065	0,1787	0,03045
11	Spielersteine auf K-Feldern	0,02838	0,06743	0,02642	0,04074	0,01889
12	Spielersteine auf L-Feldern	-0,004934	0,009804	-0,007889	-0,001006	0,007739
13	Spielersteine auf M-Feldern	0,4243	0,4223	0,4219	0,4228	0,001051
14	Spielersteine auf N-Feldern	0,1616	0,02023	0,1731	0,1183	0,06953
15	Spielersteine auf O-Feldern	-0,001714	0,01512	8,98E-4	0,004768	0,007397
16	Spielersteine auf P-Feldern	0,001095	0,002183	-0,008043	-0,001588	0,004586
17	Gegnersteine auf A-Feldern	-0,2248	-0,1814	-0,2066	-0,2043	0,0178
18	Gegnersteine auf B-Feldern	-0,2283	-0,2028	-0,2154	-0,2155	0,01043
19	Gegnersteine auf C-Feldern	-0,2162	-0,2876	-0,1878	-0,2306	0,042
20	Gegnersteine auf D-Feldern	-0,2068	-0,2706	-0,2029	-0,2268	0,03104
21	Gegnersteine auf E-Feldern	-0,3103	-0,2969	-0,3122	-0,3065	0,006779
22	Gegnersteine auf F-Feldern	-0,2337	-0,1334	-0,2572	-0,2081	0,05366
23	Gegnersteine auf G-Feldern	-0,0719	-0,1577	-0,131	-0,1202	0,03586
24	Gegnersteine auf H-Feldern	-0,03036	-0,1098	-0,05787	-0,06599	0,03292
25	Gegnersteine auf I-Feldern	-0,3791	-0,3873	-0,3529	-0,3731	0,01466
26	Gegnersteine auf J-Feldern	-0,2064	-0,09376	-0,2471	-0,1824	0,06487
27	Gegnersteine auf K-Feldern	-0,09975	-0,1387	-0,1008	-0,1131	0,01812
28	Gegnersteine auf L-Feldern	-0,1844	-0,2349	-0,08503	-0,1681	0,06226
29	Gegnersteine auf M-Feldern	-0,3914	-0,3379	-0,4065	-0,3786	0,02943
30	Gegnersteine auf N-Feldern	-0,1381	-0,02351	-0,2261	-0,1293	0,08295
31	Gegnersteine auf O-Feldern	-0,04435	-0,1745	-0,1094	-0,1094	0,05311
32	Gegnersteine auf P-Feldern	-0,4192	-0,4132	-0,382	-0,4048	0,01634
33	Spielersteine im Ring 0	0,291	0,3165	0,2744	0,294	0,01731
34	Gegnersteine im Ring 0	-0,2771	-0,2858	-0,2912	-0,2847	0,005813
35	Spielersteine im Ring 1	0,1075	0,1053	0,1481	0,1203	0,01967
36	Gegnersteine im Ring 1	-0,1248	-0,0921	-0,1919	-0,1363	0,04156
37	Spielersteine im Ring 2	-7,16E-4	0,02029	1,348E-4	0,00657	0,009708
38	Gegnersteine im Ring 2	-0,1005	-0,18	-0,0996	-0,1267	0,03768
39	Spielersteine	0,1209	0,1394	0,114	0,1248	0,01073
40	Gegnersteine	-0,2063	-0,213	-0,2073	-0,2089	0,002941
41	Spieler im Materialvorteil	0,4539	0,4925	0,4911	0,4792	0,01786
42	Gegner im Materialvorteil	-1,0	-1,0	-0,9998	-0,9999	7,855E-5

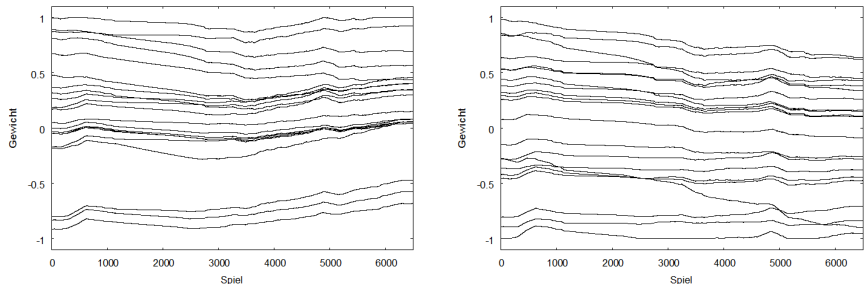
B.6 Ergebnispolynome in Versuchsgruppe F

Nr. (i)	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ_i
1	Spielersteine auf A-Feldern	0,5027	0,6454	0,7517	0,6332	0,102

Nr. (i)	Name	Versuch 1	Versuch 2	Versuch 3	Mittelwert	σ_i
2	Spielersteine auf B-Feldern	0,6588	0,5356	0,687	0,6271	0,06576
3	Spielersteine auf C-Feldern	0,5563	0,6914	0,6641	0,6373	0,05831
4	Spielersteine auf D-Feldern	0,5931	0,7533	0,7582	0,7015	0,07671
5	Spielersteine auf E-Feldern	0,7513	0,7631	0,9106	0,8083	0,07248
6	Spielersteine auf F-Feldern	0,5842	0,5947	0,6849	0,6213	0,04517
7	Spielersteine auf G-Feldern	0,3518	0,3441	0,4256	0,3738	0,03673
8	Spielersteine auf H-Feldern	0,4428	0,3765	0,4363	0,4185	0,02984
9	Spielersteine auf I-Feldern	0,7733	0,8945	0,9472	0,8717	0,07282
10	Spielersteine auf J-Feldern	0,4723	0,373	0,4637	0,4363	0,04492
11	Spielersteine auf K-Feldern	0,3368	0,3686	0,433	0,3795	0,03998
12	Spielersteine auf L-Feldern	0,3349	0,293	0,4239	0,3506	0,05461
13	Spielersteine auf M-Feldern	0,7698	0,9945	0,9883	0,9175	0,1045
14	Spielersteine auf N-Feldern	0,353	0,3843	0,4447	0,394	0,03804
15	Spielersteine auf O-Feldern	0,3304	0,3061	0,4373	0,3579	0,05701
16	Spielersteine auf P-Feldern	0,6008	0,4995	0,6272	0,5759	0,05502
17	Gegnersteine auf A-Feldern	-0,678	-0,7107	-0,62	-0,6696	0,03751
18	Gegnersteine auf B-Feldern	-0,7211	-0,6028	-0,7925	-0,7055	0,07822
19	Gegnersteine auf C-Feldern	-0,7636	-0,7503	-0,7328	-0,7489	0,0126
20	Gegnersteine auf D-Feldern	-0,5921	-0,6633	-0,8183	-0,6912	0,09444
21	Gegnersteine auf E-Feldern	-0,7046	-0,9123	-0,9667	-0,8612	0,1129
22	Gegnersteine auf F-Feldern	-0,6775	-0,6154	-0,6511	-0,648	0,02545
23	Gegnersteine auf G-Feldern	-0,3103	-0,2487	-0,452	-0,337	0,08512
24	Gegnersteine auf H-Feldern	-0,2097	-0,2175	-0,2999	-0,2424	0,04083
25	Gegnersteine auf I-Feldern	-0,8101	-0,9004	-0,9255	-0,8787	0,04954
26	Gegnersteine auf J-Feldern	-0,3086	-0,2594	-0,4591	-0,3424	0,08495
27	Gegnersteine auf K-Feldern	-0,2822	-0,2061	-0,2688	-0,2524	0,03315
28	Gegnersteine auf L-Feldern	0,02502	0,03577	-0,135	-0,02473	0,07809
29	Gegnersteine auf M-Feldern	-0,7505	-0,9153	-0,9602	-0,8753	0,09017
30	Gegnersteine auf N-Feldern	-0,2788	-0,2447	-0,3423	-0,2886	0,04045
31	Gegnersteine auf O-Feldern	0,02398	-0,01177	-0,06776	-0,01852	0,03776
32	Gegnersteine auf P-Feldern	0,1341	0,2269	0,2104	0,1905	0,0404
33	Spielersteine im Ring 0	0,6662	0,6837	0,8135	0,7211	0,06567
34	Gegnersteine im Ring 0	-0,7077	-0,6929	-0,8227	-0,7411	0,058
35	Spielersteine im Ring 1	0,4687	0,4641	0,4995	0,4774	0,01571
36	Gegnersteine im Ring 1	-0,3196	-0,3812	-0,4498	-0,3835	0,05318
37	Spielersteine im Ring 2	0,3306	0,3292	0,4527	0,3709	0,05791
38	Gegnersteine im Ring 2	-0,08532	-0,03022	-0,1241	-0,07987	0,03851
39	Spielersteine	0,5337	0,5205	0,6353	0,5632	0,05129
40	Gegnersteine	-0,4445	-0,4917	-0,484	-0,4734	0,02067
41	Spieler im Materialvorteil	1,0	0,8962	0,943	0,9464	0,04245
42	Gegner im Materialvorteil	-0,3054	-0,263	-0,1708	-0,2464	0,05619

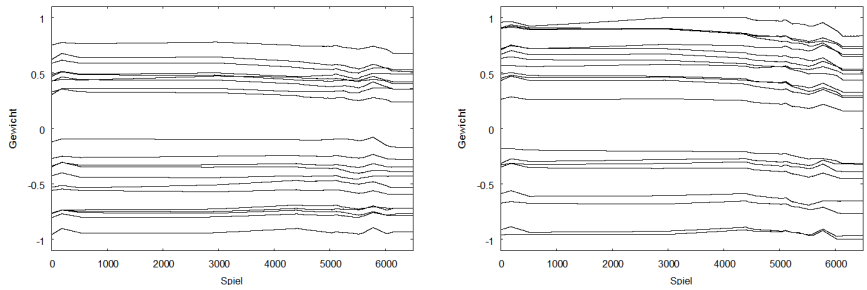
C Ergebnisse zur Versuchsgruppe D

C.1 Verlauf der Gewichte in den Lernphasen



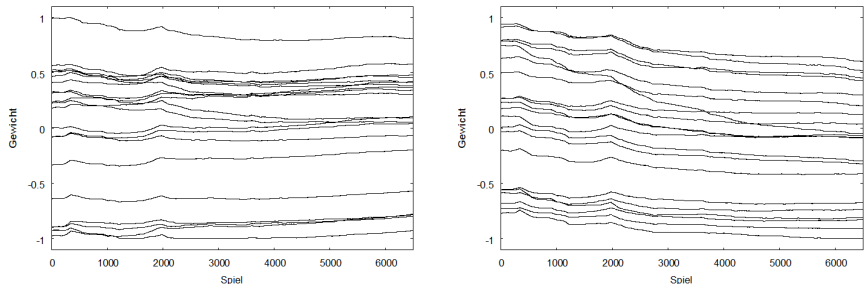
(a) Spielergewichte in Versuch 1

(b) Gegnergewichte in Versuch 1



(c) Spielergewichte in Versuch 2

(d) Gegnergewichte in Versuch 2

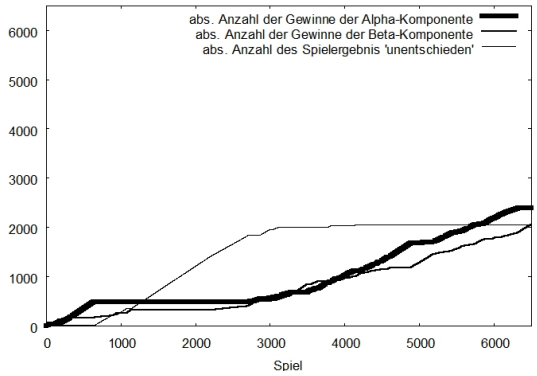


(e) Spielergewichte in Versuch 3

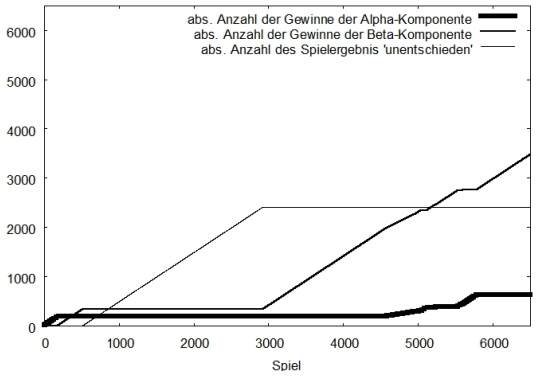
(f) Gegnergewichte in Versuch 3

Abbildung C.1: Verlauf der Gewichte in den Lernphasen der Versuchsgruppe D

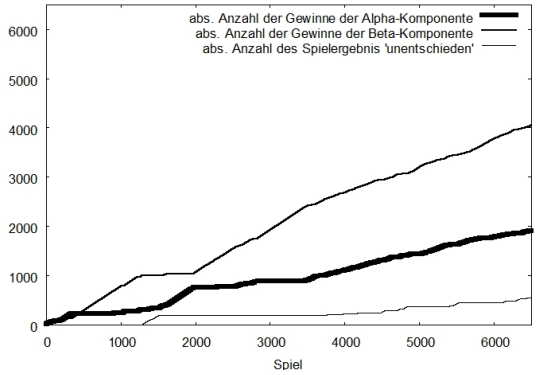
C.2 Gewinnhäufigkeiten in den Lernphasen



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung C.2: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe D

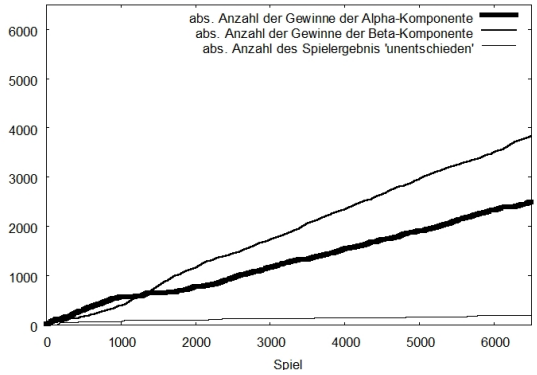
C.3 Ergebnisse der ersten Testphasen

Versuch 1	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	50,6 %	50,1 %	50,3 %	50,4 %	0,21 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	70,0 %	67,8 %	69,1 %	69,0 %	0,9 %
Versuch 2	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	14,0 %	16,1 %	15,7 %	15,3 %	0,91 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	23,3 %	26,5 %	24,3 %	24,7 %	1,34 %
Versuch 3	Gewinnhäufigkeiten des Polynoms P_0				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{x}$	σ_x
	31,3 %	29,6 %	27,8 %	29,6 %	1,43 %
	Gewinnhäufigkeiten des Polynoms P_{6500}				
	Durchlauf 1	Durchlauf 2	Durchlauf 3	Mittelwert $\bar{y}$	σ_y
	36,8 %	35,7 %	34,3 %	35,6 %	1,02 %

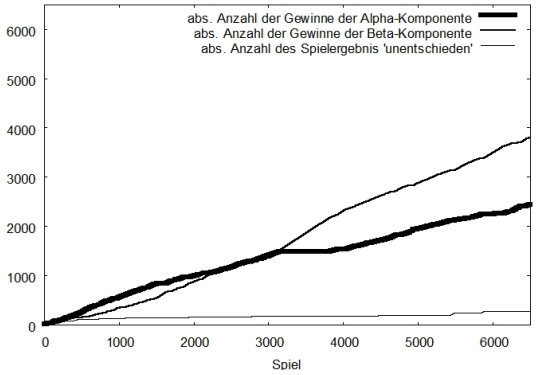
Tabelle C.1: Ergebnisse der ersten Testphasen in der Versuchsgruppe D

D Ergebnisse zur Versuchsgruppe E

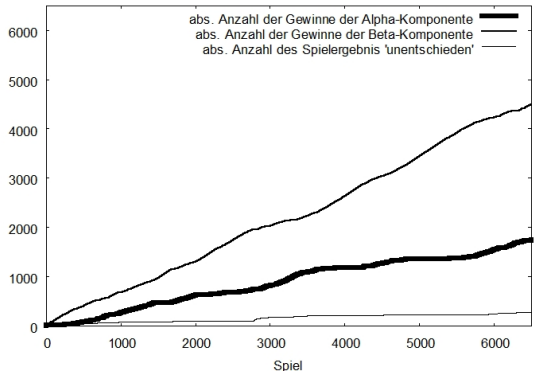
D.1 Gewinnhäufigkeiten in den Lernphasen



(a) Absolute Anzahl der Gewinne in Versuch 1



(b) Absolute Anzahl der Gewinne in Versuch 2



(c) Absolute Anzahl der Gewinne in Versuch 3

Abbildung D.1: Gewinnhäufigkeiten in den Lernphasen der Versuchsgruppe E

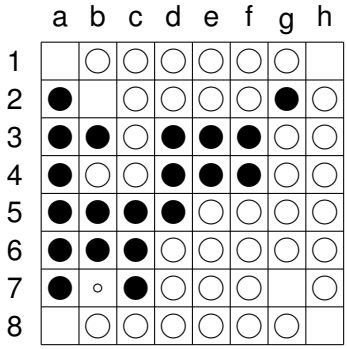
E Beispiel zum Horizonteffekt bei Othello

In den Abbildungen E.1 wird für einige Othello-Spielpositionen die Wirkung des Horizonteffekts dargestellt. Die Spielposition in E.1a ist Rose (2005) entnommen, der sein strategisches Wissen in zahlreichen Beispielen in Form von exemplarischen Spielfeld-Mustern und Spielzügen zur Verfügung stellt. Die Spielpositionen in E.1b bis E.1h stellen eine mögliche Folge von Spielzügen bis zum Ende des Spiels dar. Dabei kennzeichnen die kleinen Kreise, welcher Spieler welchen Spielzug in der gegebenen Spielposition ausführen wird. Die Bewertungen bei jeder Spielposition stellen Bewertungen für Weiß dar, sie sind mit dem Ergebnispolynom des ersten Versuchs der Versuchsgruppe E berechnet worden[17].

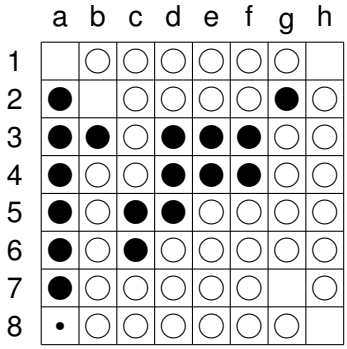
In der ersten Spielposition in E.1a führt Weiß mit 37 Steinen, denn Schwarz hat nur 20 Steine. Trotzdem schreibt Rose (2005) über diese Spielposition "White is dead", denn Schwarz kann aus dieser Spielposition durch passende Spielzüge die Spielposition in E.1h erreichen, in der Schwarz mit 52 zu 12 Steinen führt, so dass Weiß letztendlich verliert und sich nicht dagegen wehren kann. Der Grund ist, dass für Weiß in der Spielposition in E.1a nur zwei Züge möglich sind - 7b und 2b - und dass beide Züge ungünstig für Weiß sind: In dem Moment, in dem Weiß auf eines dieser Felder setzt, ist die benachbarte Ecke automatisch von Schwarz attackiert, d.h. Schwarz kann die Ecke im nächsten Zug einnehmen. Nachdem Schwarz die erste Ecke eingenommen hat, bleibt für Weiß nur der andere Zug, mit dem die zweite Ecke attackiert wird und danach von Schwarz eingenommen werden kann. Welchen der beiden Züge Weiß zuerst ausführt spielt dabei keine Rolle, das Ergebnis ist dasselbe. Im folgenden Spielverlauf hat Weiß keinen Zug mehr, so dass Schwarz alle leeren Felder besetzen kann, dabei eine Vielzahl von weißen Steinen umdreht und am Ende gewinnt.

Die Bewertungen in E.1a, E.1b, E.1c, E.1d und E.1e messen diese Bedrohung jedoch noch nicht; ihre Werte sind alle positiv, obwohl Weiß am Ende verlieren kann, wenn Schwarz die richtigen Züge einsetzt. Der Horizonteffekt tritt ein, wenn zum Beispiel eine dieser Spielpositionen im Blatt eines unvollständigen Suchbaums auftritt, denn die Tatsache, dass sich die Gewinnaussichten für Weiß im weiteren Spielverlauf verschlechtern kann und Weiß dann verliert, bleibt beim Einsatz dieser Bewertungsfunktion verborgen.

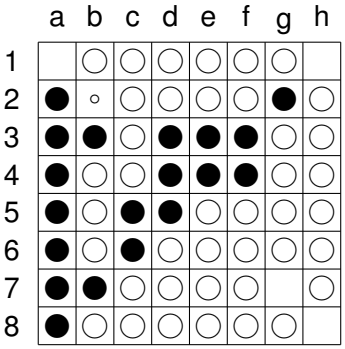
[17] Die Versuchsgruppe E wurde gewählt, weil sie von den ersten fnf Versuchsgruppen die ähnlichsten Ergebnisse hervorgebracht hat und der Versuch 1 wurde gewählt, weil er nach den Ergebnissen der ersten Testphase das spielstärkste Ergebnispolynom - also die spielstärkste Bewertungsfunktion - in der Versuchsgruppe E erzeugt hat.



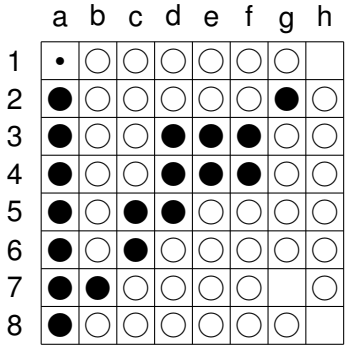
(a) Bewertung: 0.0169. Spielstand: 20:37. Quelle: Rose (2005)



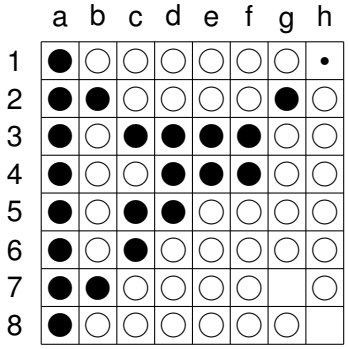
(b) Bewertung: 0.0246



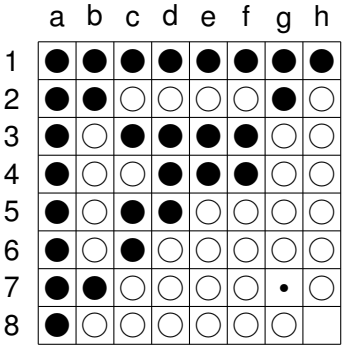
(c) Bewertung: 0.0182



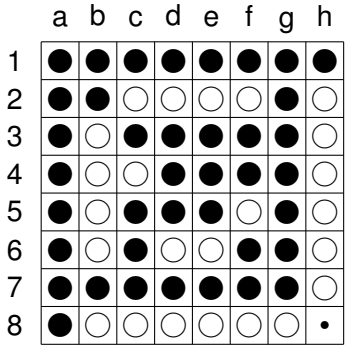
(d) Bewertung: 0.0228



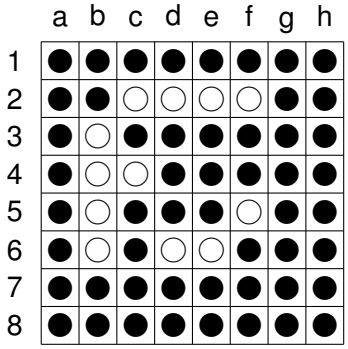
(e) Bewertung: 0.0166



(f) Bewertung: -0.0049



(g) Bewertung: -0.0579



(h) Bewertung: -1,0. Spielstand: 52:12. Schwarz hat gewonnen.

Abbildung E.1: Bewertungen des Spielers Weiß im Verlauf eines Othello-Spiels

Literaturverzeichnis

[Cooper 1998] COOPER, James W.: *Design Patterns*. Addison-Wesley, 1998

[Dayan 1992] DAYAN, Peter: The Convergence of TD(λ) for General λ. In: *Mach. Learn.* 8 (1992), Nr. 3-4, S. 341–362. – ISSN 0885-6125

[Hartz 2005] HARTZ, Johannes: *Entwicklung eines Schachprogramms für ressourcenarme Systeme wie PDAs*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2005. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/hartz.pdf`

[van den Herik u. a. 2002] HERIK, H. J. van den ; UITERWIJK, Jos W. H. M. ; RIJSWIJCK, Jack van: Games solved: now and in the future. In: *Artif. Intell.* 134 (2002), Nr. 1-2, S. 277–311. – ISSN 0004-3702

[Jin 2007] JIN, Lim Y.: *ON FORWARD PRUNING IN GAME-TREE SEARCH*. 2007. – URL `http://www.yewjin.com/papers/PhDThesisLimYewJin.pdf`

[von Luck 2004] LUCK, Kai von: *Intelligent Systems*. 2004. – URL `http://users.informatik.haw-hamburg.de/~luck/is/is-fol.pdf`

[von Neumann 1928] NEUMANN, John von: *Zur Theorie der Gesellschaftsspiele*. 1928. – URL `http://resolver.sub.uni-goettingen.de/purl?GDZPPN002272717`

[Poole u. a. 1998] POOLE, David ; MACKWORTH, Alan ; GOEBEL, Randy: *Computational Intelligence*. Oxford University Press, 1998. – ISBN 0-19-510270-3

[Reinefeld 2006] REINEFELD, Alexander: Entwicklung der Spielbaum-Suchverfahren: Von Zuses Schachhirn zum modernen Schachcomputer. In: FREYTAG, J.-C. (Hrsg.): *Aktuelle Themen im historischen Kontext*. Springer, 2006, S. S. 241–273. – ISBN 3-540-32742-8

[Rieck 2006] RIECK, Christian: *Spieltheorie*. 6. Auflage. Christian Rieck Verlag, 2006. – ISBN 3-924043-91-4

[Rose 2005] ROSE, Brian: *Othello: A Minute to Learn... A Lifetime to Master*. 2005. – URL `http://othellogateway.strategicviewpoints.com/rose/book.pdf`

[Russel und Norvig 2003] RUSSEL, Stuart ; NORVIG, Peter: *Künstliche Intelligenz*. 2. Auflage. PEARSON Studium, 2003. – ISBN 3-8273-7089-2

[Samuel 1959] SAMUEL, A. L.: *Some studies in machine learning using the game of checkers*. 1959. – URL http://www.research.ibm.com/journal/rd/441/samuel.pdf

[Shannon 1950] SHANNON, Claude E.: *Programming a Computer for Playing Chess*. 1950. – URL http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf

[Simon 1980] SIMON, Herbert A.: *Why Should Machines Learn?* 1980. – URL http://shelf1.library.cmu.edu/cgi-bin/tiff2pdf/simon/box00071/fld07604/bdl0001/doc0001/simon.pdf

[Sutton 1988] SUTTON, Richard S.: Learning to Predict by the Methods of Temporal Differences. In: *Mach. Learn.* 3 (1988), Nr. 1, S. 9–44. – ISSN 0885-6125

[Tesauro 1995] TESAURO, Gerald: Temporal difference learning and TD-Gammon. In: *Commun. ACM* 38 (1995), Nr. 3, S. 58–68. – ISSN 0001-0782

[Wrobel u. a. 2003] WROBEL, Stefan ; MORIK, Katharina ; JOACHIMS, Thorsten: Maschinelles Lernen und Data Mining. In: SCHNEEBERGER, J. (Hrsg.): *Handbuch der Künstlichen Intelligenz*. 4. Auflage. Oldenburg, 2003, S. 517–597. – ISBN 3-486-27212-8

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. Juli 2008 Miriam Tödten