Tennis
Ein Bleistiftspiel mit künstlicher Intelligenz

von Arne Ring, Oxford, Großbritannien

Als Schüler hatte ich etwa vor 25 Jahren das erste Mal Kontakt mit der $\sqrt{\text{WURZEL}}$. Verschiedene Artikel hatten mich zu näherer Beschäftigung angeregt, am stärksten ein Beitrag zur eigenen Programmierung von künstlicher Intelligenz beim Spiel „TIC-TAC-TOE“ [1].

Künstliche Intelligenz bedeutete dabei, dass man mithilfe eines Computers geeignete Spielstrategien finden möchte, die zum Gewinn führen. Eine Spielstrategie ist hierbei eine Regel, welchen Zug man bei einem bestimmten Spielstand ausführen soll. Diese Regel kann einerseits ein einzelner Zugvorschlag sein, oder aber auch mehrere mögliche Züge umfassen, von denen einer im Spiel zufällig ausgewählt wird. Das „Training“ der Computerstrategie kann mithilfe des Spielens gegen sich selbst oder gegen einen menschlichen Gegner erfolgen.

Lange hatte ich nach weiteren Spielen gesucht, in denen ich das vorgestellte Konzept einsetzen könnte. Ich fand schließlich eine strategische Umsetzung von „Tennis“ im Buch „Bleistiftspiele“ von M. Mala [2], welches etwas komplexer als TIC-TAC-TOE ist und sich sehr gut für die Implementierung eignete.

1 Das Bleistiftspiel Tennis

Tennis ist ein strategisches Spiel für zwei Spieler, bei dem der Ball durch Wahl geeigneter Zahlen auf die Seite des Gegners „geschlagen“ wird. Wir geben hier die Spielregeln entsprechend [2] wieder.

Das Spielfeld besteht aus je zwei Feldern pro Spieler, wobei zu Beginn der Ball auf der Mittellinie liegt. Wir bezeichnen die Mittellinie mit $X = 0$, bei $X = 1$ kam der Ball einmal auf der Seite von Spieler B (Innenfeld) auf, bei $X = -2$ (Außenfeld) zweimal auf der Seite von Spieler A (Abbildung 1). Die Aufteilung des Spielfeldes soll verdeutlichen, dass im Gegensatz zum richtigen Tennis der Ball zwei Mal auf jeder Seite aufkommen darf, bevor er zurückgeschlagen werden muss.

Jeder Spieler $i \in \{A, B\}$ hat zu Beginn $P_{i,0} = 50$ Punkte. Er wählt im Zug t eine Zahl $Z_{i,t}$ mit

$$1 \leq Z_{i,t} \leq P_{i,t}$$

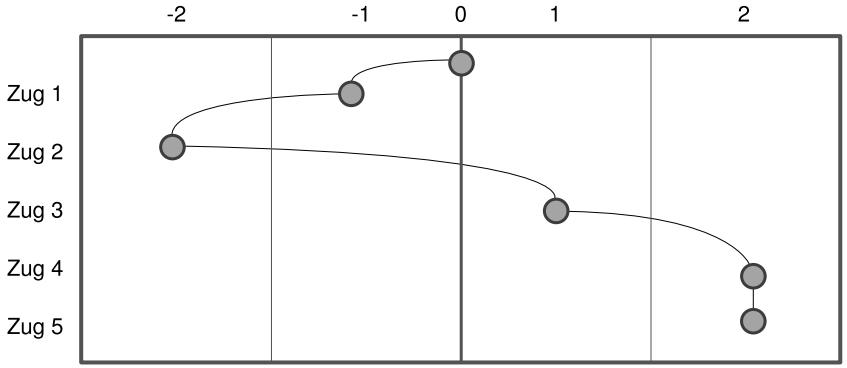

Abb. 1 Tennis-Spielfeld mit einem möglichen Spielverlauf

(außer wenn $P_{i,t} = 0$, dann $Z_{i,t} = 0$), das entspricht sozusagen seiner eingesetzten Schlagkraft. Der Ball wird dann in Richtung des Spielers geschlagen, dessen gewählte Zahl kleiner war: Lag der Ball vorher im Feld des Spielers, der mehr geboten hat, wird er auf das Innenfeld des anderen Spielers geschlagen. Lag der Ball bereits im Feld des Spielers, der weniger geboten hat, wird er vom Innenfeld ins Außenfeld bzw. vom Außenfeld ins Aus (d. h. $X_{t+1} = \pm 3$) geschlagen. Schließlich verbleibt der Ball an Ort und Stelle, wenn beide Spieler dieselbe Zahl gewählt haben ($X_{t+1} = X_t$). Außerdem verringert sich die Punktezahl beider Spieler um die gewählte Zahl:

$$P_{i,(t+1)} = P_{i,t} - Z_{i,t}\,.$$

Abbildung 1 verdeutlicht einen möglichen Spielverlauf. In der Wikipedia [3] sind noch weitere Beispiele zu finden.

Das Spiel ist gewonnen, wenn entweder der Ball ein drittes Mal auf der Seite des Gegners aufkommen würde, oder wenn beide Spieler keine Punkte mehr haben und der Ball auf dem gegnerischen Feld liegen bleibt. Wenn man eine Reihe von Spielen absolviert hat, gewinnt der Spieler mit den meisten Gewinnpunkten. Abweichend von [2] schlage ich vor, 2 Gewinnpunkte für das Herausschlagen und 1 Gewinnpunkt beim Liegenbleiben im Feld zu vergeben.

Der Reiz des Spiels besteht darin, dass man einen Ball**vorteil** auf dem Spielfeld zunächst nur durch einen Punkt**nachteil** erreicht und man deshalb versuchen muss, eine Zahl zu wählen, die nur wenig größer als die des Gegner ist, damit dieser Nachteil nicht zu groß wird. Eine optimale Strategie ist bei 50 Startpunkten nicht offensichtlich – soll man zunächst eine kleine Punktzahl wählen, damit sich der Gegner verausgabt und man selbst leichter zurückschlagen kann, oder soll man versuchen, möglichst schnell sowohl in Ball- als auch Punktvorteil zu gelangen, indem man immer abwechselnd eine große und eine kleine Zahl wählt?

Im Gegensatz zum TIC-TAC-TOE werden die Züge nicht abwechselnd, sondern gleichzeitig durchgeführt, sodass man die möglichen Gegnerzüge bei der eigenen Auswahl mit berücksichtigen muss. Es scheint jedoch nicht ratsam, nur eine einzige Strategie zu verfolgen, denn wenn man ausrechenbar ist, könnte der Gegner relativ leicht die Oberhand gewinnen.

Spielergebnis		**Nächster Zug Spieler B**		
		1	2	3
Nächster Zug Spieler A	1	1	−2	−2
	2	1	1	−2
	3	−1	1	1
	4	−1	−1	1

Tab. 1 Bestmögliches Endergebnis (Gewinnpunkte), wenn beim Stand von $P_A = 4$, $P_B = 3$ und $X = -2$ der *nächste* Zug jeden Spielers entsprechend der Tabelle erfolgt.

Ein Beispiel verdeutlicht den Nutzen einer zufälligen Strategie im späten Spielverlauf. Nehmen wir an, Spieler A hat noch 4 Punkte, Spieler B nur drei, und der Ball liegt im Außenfeld von Spieler A ($P_A = 4$, $P_B = 3$, $X = -2$). Aus dieser Situation kann A den Ball nicht mehr ins Aus schlagen, doch alle anderen drei Spielausgänge sind noch möglich. Tabelle 1 gibt das beste endgültige Spielergebnis wieder, das nach der Wahl des ersten Zuges erreicht werden kann. Man erkennt, dass Spieler B den Ball in einigen Fällen herausschlagen kann, aber nur, wenn Spieler A im ersten Zug „1“ oder „2“ spielen würde. Wenn Spieler A deshalb nur „3“ oder „4“ spielen würde, würde Spieler B einfach „1“ ziehen, und würde sich damit auf jeden Fall einen Gewinnpunkt sichern. Wenn Spieler A in dieser Situation gewinnen möchte, muss Spieler A auch die Züge „1“ und „2“ in Betracht ziehen, auch wenn er damit riskiert, gleich zwei Punkte zu verlieren.

Im Endspiel, wenn beide nur wenige Punkte haben, können in vielen Fällen Gewinnstrategien für einen der Spieler angegeben werden, auch hier sind in Wikipedia einige Fälle vorgestellt. Doch es ist nicht klar, bei welcher Punktzahl das Endspiel beginnt, und wie (und ob) man das jeweilige Endspiel überhaupt erreicht. Es ist zu bezweifeln, dass man bei der Startpunktzahl von 50 eine optimale Strategie direkt berechnen kann. An dieser Stelle kommt das Training einer künstlichen Intelligenz ins Spiel – mithilfe des Computers sollen günstige (im besten Fall optimale) Strategien ermittelt werden.

2 Training des Computers

Eine Spielstellung wird durch das Tripel (P_A, P_B, X) gekennzeichnet, wobei P_i die mögliche Punktzahl beider Spieler bedeutet und X der Ballort. (Man kann sich überlegen, dass der Zeitpunkt t für den weiteren Spielverlauf nicht wichtig ist.) Jeder Spieler hat 50 Punkte, und es gibt 5 Orte für den Ball, sodass sich (inklusive der Möglichkeit, 0 Punkte zu haben) $51 \cdot 51 \cdot 5 = 13005$ Spielstellungen ergeben.

In jeder Spielstellung können maximal 51 verschiedene Züge gemacht werden. Das bedeutet, dass der Computer eine Spielmatrix der Größe 13005×51 verwalten muss. In jeder Spielstellung soll erfasst werden, ob die jeweilige Strategie erfolgreich war oder nicht. Je öfter die Strategie erfolgreich war, desto eher kann man annehmen, dass dieser Zug günstig war. Im späteren Spielverlauf sollten diese günstigen Spielstrategien öfter gewählt werden, um die Gewinnwahrscheinlichkeit zu erhöhen.

Zu Beginn sollten alle möglichen Züge bei jeder Spielstellung gleichwahrscheinlich gewählt werden können. Dies kann man durch einen Vektor wie z. B. $(a_j) = (0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, \dots)$ darstellen, wobei „1“ ein erlaubter und „0“ ein verbotener Zug ist. Die Wahrscheinlichkeit für einen möglichen Zug bei einem Zustand wird dann als

$$p_j = \frac{a_j}{\sum_i a_i}$$

angenommen. Für alle Züge, die später zum Sieg geführt haben, wird das gewählte a_i um 1 erhöht, sodass im Laufe des „Trainings“ beispielsweise der Vektor $(0, 123, 78, 12, 1, 4, 32, 21, 1, 0, 0, \dots)$ entsteht. Die ersten beiden Möglichkeiten führten im Verlauf vieler Spiele oft zum Gewinn, die anderen Züge eher selten. Wenn man eine gleichverteilte Pseudo-Zufallszahl mit dem Computer generiert, können die Züge entsprechend der so eingestellten Wahrscheinlichkeit ermittelt werden.

Nach einer größeren Anzahl Spiele werden sich bestimmte Züge als geeignete Gewinnzüge herausstellen. Da diese Züge nun häufiger gespielt werden, kann der Computer wiederum mögliche „Abwehrstrategien“ dagegen entwickeln, wenn er auch die Rolle von Spieler B einnimmt. Auf dieser Weise erhofft man sich, dass man im weiteren Verlauf des „Trainings“ einer stabilen Strategie zustrebt, also der Algorithmus gegen eine (die?) tatsächliche Gewinnstrategie konvergiert. Ob das bei diesem Spiel der Fall ist, müsste man mathematisch noch zeigen.

Wichtig ist hierbei insbesondere, dass eine Strategie so lange eine zufällige Komponente hat, so lange sie noch nicht deterministisch zum Sieg führt, denn sonst kann der Gegner oft eine einfache Gegenstrategie entwickeln. Außerdem kann eine Strategie für einen „untrainierten“ Gegner als erfolgreich erscheinen, während sie durch einen trainierten Algorithmus jedoch leicht abgewehrt werden könnte.

Wenn der Computer nur gegen sich selbst spielt, ist jedoch nicht gewährleistet, dass tatsächlich optimale Strategien gefunden werden. Vergleichbar mit Fällen in der Numerik wird vielleicht nur ein „lokal optimales“ Verfahren ermittelt, dass sich nicht mit menschlichen Gegnern messen kann. Umgekehrt kann das Training mit menschlichen Gegnern oft nur im begrenzten Maßstab erfolgen, denn schon 100 Spiele gegen eine unerfahrene Strategie werden sicherlich als langweilig empfunden.

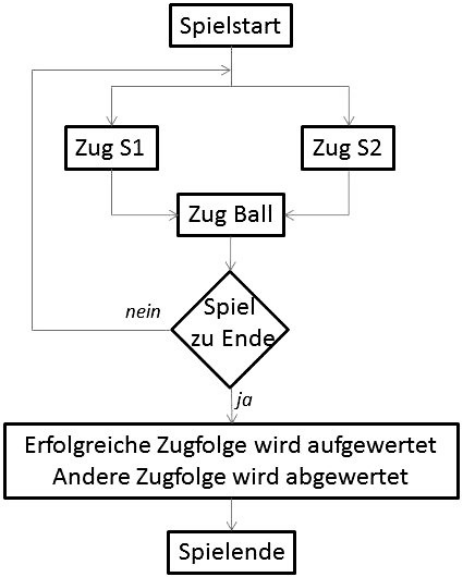

Abb. 2 Prinzip des Spiel- und Lernalgorithmus von Tennis

3 Umsetzung

Für die Umsetzung des Spiels habe ich die Programmiersprache R gewählt, die frei verfügbar ist [4] und relativ einfach zu erlernen. Weiteres hierzu und das entstandene Programm können auf der $\sqrt{\text{WURZEL}}$-Homepage unter

bleistifttennis_simple.R oder bleistifttennis_extended.R

heruntergeladen werden.

Auf meinem Rechner konnte ich etwa 100 bis 200 Spiele pro Sekunde durchführen. Das bedeutet, dass ca. 500.000 Spiele pro Stunde gespielt werden können, was zu sehr guten Spielstrategien führte. Beim Vergleich zu einem (nicht sehr trainierten) menschlichen Gegner (also mir) gewann der Algorithmus etwa 50% der Spiele. Allerdings ist es durchaus möglich, Lücken in den Computerstrategien zu finden, die der Computer kaum gespielt hat, sodass er noch keine stabile und optimale Zugauswahl durchführt.

4 Ausblick

Es lohnt sich hierbei darüber nachzudenken, ob man noch bessere Möglichkeiten entwickeln kann, wie der Computer „lernt“, als nur das einfache Inkrementieren der Wahrscheinlichkeiten der Gewinnzüge. Besseres Lernen bedeutet in diesem Fall, dass die Konvergenz zu einem stabilen Algorithmus schneller erfolgt.

Eine Möglichkeit ist dabei, dass der Computer auch aus Niederlagen lernt, und dass so die Wahrscheinlichkeit dieses Zuges wieder sinkt. Wichtig ist dann jedoch, dass jeweils erlaubte Züge auch immer noch eine positive Wahrscheinlichkeit behalten, gewählt zu werden, denn wenn ein Zug einmal die Wahrscheinlichkeit 0 besitzt, dann wird dieser Zug nie mehr gewählt werden, was den Spielraum unnötig einschränken würde.

Wir hatten bereits erwähnt, dass das Lernen des Algorithmus dazu führt, dass der Computer weniger „ausprobiert“. Das bedeutet, dass Züge, die nicht oft getestet wurden, ebenso selten gespielt werden wie Züge, die zur Niederlage führten – und vielleicht würden sie zu einer optimalen oder zumindest besseren Strategie führen? Es gibt verschiedene Möglichkeiten, dieses Problem anzugehen. Beispielsweise könnte man die Bewertungsfunktion verändern, um den Gewinn selten gewählter Züge höher zu bewerten. Oder man führt ein Update der gelernten Strategie nur alle x Spiele durch, sodass Gewinnzüge nicht sofort öfter gewählt werden. Außerdem könnte es sich lohnen, wenn man nicht immer nur bei der Punktzahl 50 beginnt, sondern gezielt den Mittelteil des Spiels trainiert (mit Startpunktzahlen von 10 bis 40). Letztlich sollte es angestrebt werden, dass Spielstände (P_A, P_B, X) öfter trainiert werden, wenn $|P_A - P_B|$ klein ist, da gute Spieler den Abstand der Punkte eher gering halten werden.

Ein letzter Gedanke im Rahmen des Lernens von künstlicher Intelligenz ist die Einbeziehung einer Vorausberechnung des Computers. Auch wenn die Computerstrategien aus den bisherigen Zügen lernen, können sie schlecht mit ungewöhnlichen Gegnerzügen umgehen. Wenn man das Programm erweitert und die Wertung von möglichen Spielzügen vorausberechnet (mit einem Suchbaum, ähnlich guten Schachprogrammen), könnte man die Intelligenz des Computers im Tennis-Spiel noch

weiter verbessern. Allerdings bedeutet dies auch deutlich längere Laufzeiten jedes Spiels, denn der Vorteil des Lernens aus der Spielmatrix ist gerade die Schnelligkeit der Zugauswahl.

5 Literaturanregungen

Eine Einführung in die Spieltheorie ist in [5] enthalten. Dort wurde auch ein Wettbewerb von Spielalgorithmen für das „wiederholte Gefangenendilemma“ beschrieben. Ein weiteres schönes Beispiel für das Lernen mit künstlicher Intelligenz ist eine Umsetzung des Spiels „Schere-Stein-Papier“ [6].

Literatur

[1] Joachim Puhl: *Ein Computer lernt TIC-TAC-TOE.* $\sqrt{\text{WURZEL}}$ 9/1987

[2] M. Mala: *Das große Buch der Block- und Bleistiftspiele.* Hugendubel, 1998.

[3] *Tennis (Bleistiftspiel).* Wikipedia, 2012.

[4] R Development Core Team: *R: A Language and Environment for Statistical Computing.* Vienna, Austria, 2011, http://www.R-project.org

[5] O. Häggström: *Streifzüge durch die Wahrscheinlichkeitstheorie.* Springer, 2005.

[6] *Rock-Paper-Scissors: You vs. the Computer.* New York Times website. http://www.nytimes.com/interactive/science/rock-paper-scissors.html, 2012.