Maschinelles Lernen: Reinforcement Learning (Bestärkendes Lernen)

Ideen der Informatik

Kurt Mehlhorn

Juli 2021

Übersicht

- Was ist bestärkendes Lernen (Reinforcement Learning)?
- Stand der Kunst
- Computerschach
  - Geschichte
  - Computerschach vor 2015
  - Computerschach seit 2015

Bestärkendes Lernen (Reinforcement Learning)

- Agent (Computerprogramm) interagiert mit seiner Umwelt, indem er sie beobachtet und Aktionen ausführt.
- Die Umwelt verändert sich; zum Teil als Reaktion auf die Aktionen des Agenten und zum Teil unabhängig davon. Die Reaktion der Umgebung kann eine Belohnung/Bestrafung beinhalten.
- Der Agent möchte eine Strategie entwickeln, die langfristig zu einer möglichst hohen Belohnung führt.
- Wir reden also über Situationen, die sich zumindest in ähnlicher Form wiederholen, und Umgebungen, die sich in gewissem Umfang vorhersagbar verhalten.
- Für Menschen die häufigste Art zu lernen.
- Schwierigkeiten: Belohnung ist nicht direkt; nur partielles Verständnis der Umgebung; erprobtes versus neues Verhalten.

Beispiele

- Kontrolle der Bewegung eines Roboters.
  - Roboter in der industriellen Praxis: Position der Teile genau definiert; Bewegung wird dem Roboter beigebracht; eingeschränkte Sensorik.
  - Roboter in der aktuellen Forschung: Position der Teile muss aus Kamerabild geschlossen werden; Roboter muss Bewegung lernen; Sensorik = Sehen, Tastsinn, Rückmeldung der Stellmotoren, ...
- Spiele: Schach, Go, Atari-Spiele.

ZDF

Levine et al, JMLR, 2016.

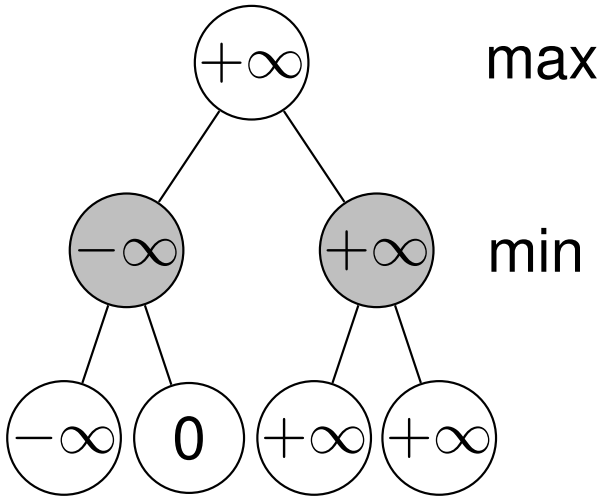
Computerschach, Geschichte

- Turing, von Neumann, Zuse haben über Schach nachgedacht.
- Schach und Go sind schwierige Spiele. Glaube bis 70er: Wenn Computer Schach können, dann folgt auch alles andere.
- H. A. Simon (1957): Computer schlägt Schachweltmeister vor 1967.
- 1997: IBM's Deep Blue schlägt Weltmeister Garri Kasparov 3.5 zu 2.5. IBM hatte 5 Mio Dollar in die Entwicklung der Maschine und des Programms investiert. Entwicklung wurde von Großmeistern unterstützt.
- 2021: Programm Stockfish braucht keine teure Hardware mehr und ist jedem Menschen weit überlegen.
- 2017: AlphaZero erreicht ähnliche Spielstärke wie Stockfish durch Deep Reinforcement Learning.
- Vorsicht: Evolution hat Menschen nicht für Schach optimiert, sondern für Sehen, Sprechen, Gefühle zeigen, ...

Computerschach: Minimax

Schach ist im Prinzip ein triviales Spiel.

- Stelle den Spielbaum auf (Die maximale Anzahl von Zügen ist beschränkt).
- Bewerte die Blätter (= Endstellungen) mit $+\infty$ (Weiß gewinnt), 0 (Remis) und $-\infty$ (Schwarz gewinnt). Werte sind aus der Sicht von Weiß.
- Propagiere die Bewertungen zur Wurzel.
  - Weißer Knoten (max Knoten) = größter Wert eines Kindes.
  - Schwarzer Knoten (min Knoten) = kleinster Wert eines Kindes.



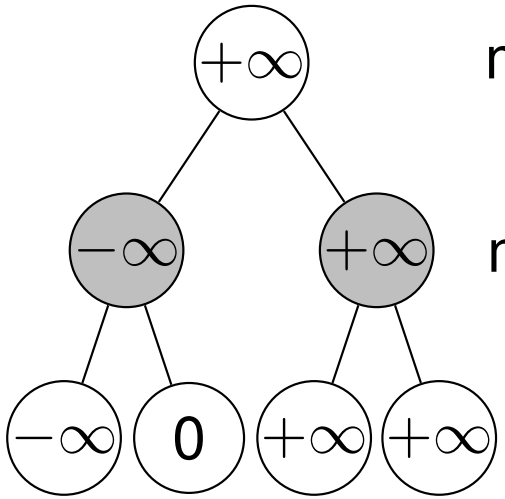
Weißer Knoten = Weiß ist am Zug.

Problem: Der Baum ist sehr groß. 10^{60} Knoten bei jeweils 10 Zugmöglichkeiten und Spieldauer = 30 Doppelzüge.

Computerschach: Minimax

Schach ist im Prinzip ein triviales Spiel.

- Stelle den Spielbaum auf (Die maximale Anzahl von Zügen ist beschränkt).
- Bewerte die Blätter (= Endstellungen) mit $+\infty$ (Weiß gewinnt), 0 (Remis) und $-\infty$ (Schwarz gewinnt). Werte sind aus der Sicht von Weiß.
- Propagiere die Bewertungen zur Wurzel.
  - Weißer Knoten (max Knoten) = größter Wert eines Kindes.
  - Schwarzer Knoten (min Knoten) = kleinster Wert eines Kindes.

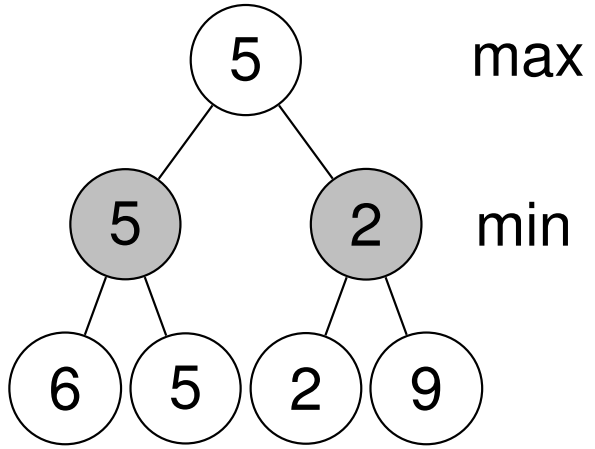


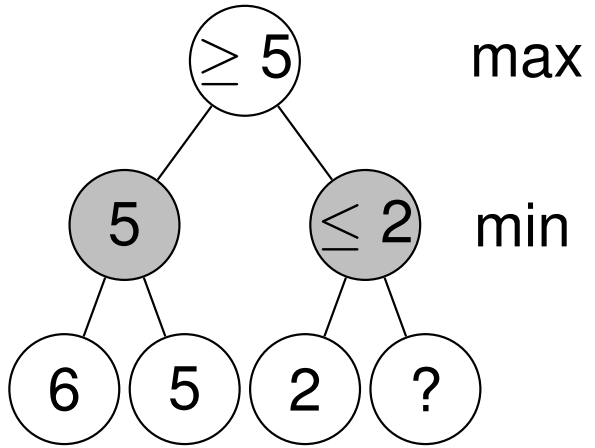
Weißer Knoten = Weiß ist am Zug.

Problem: Der Baum ist sehr groß. 10^{60} Knoten bei jeweils 10 Zugmöglichkeiten und Spieldauer = 30 Doppelzüge.

Computerschach (bis 2015): Minimax, Alphabeta, Bewertungsfunktion

- Definiere mit Hilfe von Schachexperten eine Bewertungsfunktion für Stellungen, z.B., Dame = 10, Turm = 5, Läufer = 3, . . . , ungedeckter Bauer = -1, ...
- Entwickle den Spielbaum ausgehend von der augenblicklichen Stellung, solange die Zeit reicht.
- Bewerte die Blätter des Baumes und propagiere mit Hilfe von Minimax die Bewertung zur Wurzel.
- α-β-Abschneiden: Überspringe Teile des Baumes, die irrelevant sind, z.B., den Knoten mit der Beschriftung ? und alles was darunter ist.





Computerschach (seit 2015): Monte-Carlo Baumsuche und Bewertung durch neuronales Netz

- Stellung = Position der Figuren + Wer ist am Zug.
- Neuronales Netz N für die Funktion

  $$Q(s, a) = \text{Wert des Zuges } a \text{ in der Stellung } s.$$

- Das Netz N wird zufällig initialisiert.
- Ideal wäre $Q(s, a) = \begin{cases} 1 & \text{falls } a \text{ optimal in } s, \\ 0 & \text{sonst.} \end{cases}$
- optimal = bestmögliches Ergebnis für den Spieler am Zug.
- Wie können wir Q verbessern? Wir machen mehrere Millionen Spiele unter Verwendung von Q und benutzen den Ausgang der Spiele, um Q zu verbessern. Q bestimmt die Züge von Weiß und Schwarz (Spiele gegen sich selbst).

Computerschach (seit 2015): Monte-Carlo Baumsuche und Bewertung durch neuronales Netz

- Neuronales Netz N für die Funktion
$$Q(s,a) = \text{Wert des Zuges } a \text{ in der Stellung } s.$$
- Wie können wir $Q(s,\ )$ verbessern? Wir spielen viele (zufällige) Spiele beginnend in s.
  - Sei $q_0 \in [0,1]$, z.B., $q_0 = 0.95$; q_0 wächst mit Lernfortschritt.
  - Wenn wir in Stellung s' sind, wähle $q \in [0,1]$ zufällig.
    - Falls $q < q_0$, spiele a' mit Wahrscheinlichkeit $\frac{Q(s',a')}{\sum_{b'} Q(s',b')}$ (Ausnutzen des bekannten Wissens).
    - Falls $q \geq q_0$, spiele einen zufälligen Zug a' (Erforsche Neuland).
  - Tabelliere $W(s,a) = $ Anzahl der gewonnenen Spiele mit erstem Zug a.
  - Wir ändern die Gewichte des Netzes so ab, dass sich die Verteilungen $\frac{Q(s,a)}{\sum_b Q(s,b)}$ und $\frac{W(s,a)}{\sum_b W(s,b)}$ ähnlicher werden.
- Eine sehr natürliche Vorgehensweise.

Silver et al: Science 2018.

Zusammenfassung, Reinforcement Learning

- Sehr erfolgreich bei Spielen (Schach, Go, Atari-Spiele).
- Beginnt mit Kenntnis der Spielregeln und Null Schachwissen.
- Schachwissen wird erworben durch Hunderte Millionen Spiele gegen sich selbst.
- Schlägt Großmeister nach wenigen Tagen (auch bei Go); ELO 3200.
- Findet neue herausragende Züge.

- Anwendungen: Robotersteuerung, selbstfahrende Autos, Steuerung von Fertigungsstraßen, Optimierung von Abläufen.