Reinforcement Learning

Anmerkung: Die Grenzen zu Dynamic Programming verschwimmen.

Aus Zusammenfassung

- Hier betrachten wir die Umwelt nicht nur propabilistisch, wir kennen die Umwelt nicht einmal.
- Das heißt, der Agent kennt die Umwelt gar nicht, das heißt die einzige Chance, mit der Umwelt umzugehen, ist die Umwelt zu explorieren. - Das nennt sich **Reinforcement Learning**
- Methodisch passiert hierbei folgendes: Aus dynamischem Programmieren mit einem Modell der Welt (Sie nehmen eine Wertefunktion, die Ihnen die optimale Strategie beschreibt. Die Wertefuntion ist rekursiv aufgebaut und sie können zurückiterieren und so die optimale Entscheidung treffen) wird **Q-Learning** oder ältere Formen wie TD-Learning oder SARSA.
- Es ist wichtig zu verstehen, dass wenn ich ein Modell, das ich kenne, stochastisch mache indem ich es ersetze durch Daten, hierbei Q-Learning herauskommt.

Der Agent, der eine Entscheidung treffen soll, hat keine Ahnung von der Umwelt, er kann nur mit der Blackbox interagieren. Die Blackbox gibt nur einen Reward für die Aktion aus.

Es ist eine stochastische Variante von dynamischem Programmieren.

Motivation

Reinforcement Learning bedeutet es, zu lernen, in einer unbekannten Umgebung möglichst gut zu performen, also die Umwelt kennen zu lernen und möglichst hohe Rewards zu bekommen.

Es geht weiterhin um MDPs.

Lernen in MDPs

Bei der Interaktion mit der Umwelt dammelt der Agent Datenblöcke aus `State, Action, immediate Reward,next State`. Was können wir hieraus lernen?

- Model-based RL:
    - Lernen, den nächsten State zu schätzen: `P(s'|s,a)`
    - Lernen, den immediate Reward zu schätzen: `R(r|s,a)`
- Model-free RL:
    - Lernen, die Value eines States oder eines State-Action-Paars zu schätzen: `V(s) / Q(s,a)`
- Policy Search:
    - Schätze den "Policy Gradient"
    - Oder: Nutze die Black Box

Q-Learning

Bei Q-Learning wird gegenüber der Q-Funktion der `TD`-Error (Temporal Difference) kompensiert:

Q-Funktion: `Q*(s,a) = R(s,a) + \gamma \sum{s'} P(s'|s,a) max{a'} Q*(s',a')`

Q-Learning: `Q_new(s,a) = Q_old(s,a) + \alpha [r + \gamma max{a'} Q_old(s',a') - Q_old(s,a)]`

Reinforcement:

- wenn man mehr Reward bekommt als erwartet, erhöht sich `Q(s,a)`
- wenn man weniger Reward bekommt als erwartet, sinkt `Q(s,a)`

Q-Learning wird wiederholt, bis man mit dem Ergebnis zufrieden ist.

Epsilon-greedy action selection

Hier wird nicht wie bisher die Aktion mit der höchsten Value gewählt, sondern manchmal (mit der Wahrscheinlichkeit `\epsilon`) eine zufällige Aktion.

Exploration

Epsilon-greedy Exploration

Hier wird mit `\Epsilon`-G-A-S eine Aktion ausgewählt, um zufällig auch andere Aktionen zu erforschen.

Optimistic Initialization

Anstatt mit `0` werden alle `Q(s,a)` mit einem Wert höher als `R_max` initialisiert. So wird sichergestellt, "dass ein `Q(s,a)`so lange ausprobiert wird, bis man weiß, dass es nicht gut ist".

R-MAX

Modell-basiertes RL. Alle Rewards sind `R_max`, wenn

Klausurrelevant ist

- "Konzeptionell ist das Wichtigste überhaupt bei Reinforcement Learning, dass man die Grundlage von Reinforcement Learning versteht: Dass man Q-Learning ableiten kann durch eine stochastische Varianz und dynamisches Normieren
- "Ein Agent läuft in der Welt rum, wie oft müsste er zufällig bis zum Ziel kommen bis er am Start eine Value != 0 hat (einfacher als in Übung)
- Epsilon-greedy Exploration sollte man wissen