(Deep) Reinforcement Learning - Policy Gradient

Thilo Stegemann
Hochschule für Technik und Wirtschaft
Master Student der Angewandten Informatik
12459 Berlin, Wilhelminenhofstraße 75A
Email: t.stegemann@gmx.de

***Abstract*—The abstract goes here.**

I. Reinforcement Learning (RL)

Sutton und Barto [1] definieren ein Standard Reinforcement Learning Framework, in diesem interagiert ein Agent mit einer Umgebung. Dieses standart RL-Framework wird nachfolgend ausführlich beschrieben, denn es ist eine elementare Grundlage für das Verständnis des Policy Gradient Verfahrens. Der Agent lernt, in einer ihm unbekannten Umgebung, durch Versuch und Irrtum eine Strategie π (eng. Policy). Der Agent kann in einer bestimmten Umgebung bestimmte Aktionen a ausführen. Ist die Menge der Aktionen begrenzt, dann ist es ein diskreter Aktionsraum A. Eine unbegrenzte Menge von Aktionen bezeichnet einen kontinuierlichen Aktionsraum. Die Umgebung bestimmt die Aktionsmenge und der Agent entscheidet welche Aktion aus dieser Menge ausgewählt wird. Eine Aktion kann eine Positionsangabe, eine Richtung oder etwas viel komplexeres sein.

A. Markov-Entscheidungsprozess

Sutton und Barto [1] definieren die Umgebung als einen Markov-Entscheidungsprozess (eng. **M**arkov **D**ecision **P**rocess, kurz MDP). Der Agent interagiert demnach mit einem MDP und erhält von diesem einen Zustand und eine Belohnung. Das MDP erhält vom Agenten eine ausgewählte Aktion. Ein MDP ist ein sequentielles Entscheidungsproblem. Eine Sequenz $...S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}...$ beschreibt die Interaktion des Agenten mit dem MDP. Ein Zustand ist eine Darstellung der Umwelt zu einem Zeitpunkt t. Die Zustände bestimmten, welche Aktionen ausgeführt werden können d.h. es existiert eine Funktion $A(S_t)$. Ergebnis dieser Funktion ist eine zulässige Menge von Aktionen A_t in einem Zustand S_t. Der Agent erhält eine direkte Belohnung R_{t+1} für das Ausführen einer Aktion A_t in einem Zustand S_t. Ein Beispiel: Es existiert ein Tic Tac Toe Spiel mit 9 Spielfeldern. Das Tic Tac Toe Spiel ist die Umgebung. Eine Darstellungsmöglichkeit des Spielfelds ist z.B. eine 3x3 Matrix. Jedes Element der Matrix wird mit einem Leerzeichen (' ') initialisiert. Der Startzustand $s_0 \in S$ ist eine 3x3 Matrix, indem jedes Element der Matrix ein Leerzeichen ist. Ein Spieler kann Kreuze ('X') in die Matrix einfügen und der andere Spieler Kreise ('O'). Der gesamte Zustandsraum S eines Tic Tac Toe Spiels, kann abgebildet werden. Die leeren Matrixelemente bestimmten die Positionen auf die Spielsteine gesetzt werden können. $A(s_0) = \{(0,0),(0,1),...,(2,1),(2,2)\}$ und die Policy $\pi(s_0) = a_0$ legt fest, welche Aktion a_0 im Startzustand s_0 ausgeführt werden soll ($a_0 \in A(s_0)$). Eine direkte Belohnung $R_{t+1}(s_t, a_t)$ ungleich 0 erhält der Agent für dieses Beispiel erst, wenn ein Spieler gewinnt oder verliert. Ein weiteres Beispiel: Ein Modellhubschrauber soll eigenständig lernen zu fliegen ohne abzustürzen und bestimmte Manöver auszuführen. In diesem Beispiel ist die Steuerung des Modellhubschraubers die zu lernende Strategie des RL Algorithmus. Eigenschaften der Umwelt sind unter anderem: Luftdruck, Position und Geschwindigkeit des Hubschraubers, Windgeschwindigkeit, Treibstoff. Die Modellierung der Zustände in diesem zweiten Beispiel ist ungleich komplexer verglichen mit dem ersten Beispiel. Zustände und Aktionen können sehr komplexe Objekte sein, dahingegen ist die Belohnung ein numerischer Wert und kein komplexes Objekt.

Die numerische Belohnung ist eine Bewertung der Aktion des Agenten. Der Agent versucht diese numerische Belohnung, über die Zeit, zu maximieren. Der Agent kann eine Belohnung von $+10$ erhalten (Hubschrauber um 360 Grad gedreht), wenn er in einem Zustand s eine Aktion a mit $s \in S$ und $a \in A$ ausführt. Der Agent kann auch eine negative numerische Belohnung von z.B. -23 erhalten (Hubschrauber abgestürzt). Der genaue Wert der Belohnung wird von der Umgebung definiert und der Agent kann diesen Wert nicht direkt beeinflussen oder verändern. Allein die Entscheidungen d.h. die Aktionsauswahl des Agenten entscheidet über die erhaltene Belohnung r. Die Umgebung definiert eine Belohnungsfunktion $R_s^a = \mathbb{E}\{r_{t+1}|s_t = s, a_t = a\}$. Diese Funktion ist eine Abbildung von Zustand-Aktionspaaren auf erwartete Belohnungen.

In einer realen Umgebung kann es passieren, dass der Agent selbst (z.B. ein einfacher Laufroboter oder ein Hubschrauber) eine Signalstörung empfängt oder eine Fehlfunktion seiner mechanischen Steuereinheiten hat. Die Umgebungsgegebenheiten können sich ebenfalls verschieben oder verändern (z.B. wechselnde Windrichtungen, Windstärken-Schwankungen, andere autonom agierende Agenten). Man bezeichnet diese Möglichkeiten als die Dynamiken der Umgebung. Abgebildet werden diese Dynamiken mittels Wahrscheinlichkeiten. Die Zustandsübergangswahrscheinlichkeit das ein Agent in Zustand s_t mit der Aktion a_t in den Zustand s'_t übergeht ist

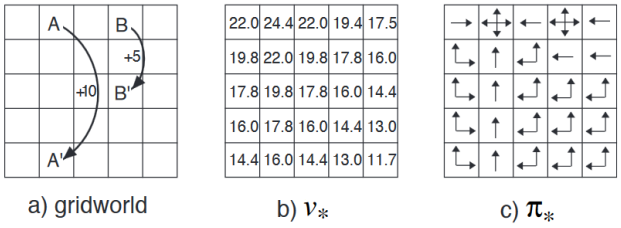

Fig. 1. Optimale Lösungen des Rasterwelt Beispiels nach [1].

definiert durch die Wahrscheinlichkeit $P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$. Die Umgebungsdynamik beeinflusst auch die Policy π und die Belohnungsfunktion R^a_s.

B. Wertefunktionsapproximation

Die nachfolgenden Ausführungen basieren auf den Erläuterungen von D. Silver vgl. [4]: Eine nicht approximierte Wertefunktion ist eine Wertetabelle. In dieser Wertetabelle hat jeder Zustand s einen Eintrag der Form $V(s)$ oder jedes Zustands-Aktionspaar s, a hat einen Eintrag $Q(s, a)$. Für große Markov-Entscheidungsprozesse existieren zu viele Zustände und Aktionen, um diese in der Wertetabelle speichern zu können. Das eigentliche Lernen der Wertefunktion ist zudem sehr langsam, weil für jeden Zustand ein individueller Wert gelernt werden muss. Die Lösung für große MDPs ist daher eine geschätzte (eng. estimated) Wertefunktion. Die Schätzung erfolgt mittels Funktionsapproximation:

$V_\theta(s) \approx V^\pi(s)$, dabei ist $V^\pi(s)$ die wahre Zustands-Wertefunktion, diese definiert wie viel erwartete Belohnung kann der Agent wirklich erhalten, wenn er der Strategie π folgt und sich in Zustand s befindet.

$Q_\theta(s, a) \approx Q^\pi(s, a)$, dabei ist $Q^\pi(s, a)$ die wahre Q-Funktion, welche für ein Zustands-Aktionspaar s, a definiert, wie hoch die wahre zu erwartende akkumulierte Belohnung ist, unter Berücksichtigung der Strategie π.

$V_\theta(s)$ und $Q_\theta(s, a)$ sind Funktionsapproximationen, welche sich an die wahren Wertefunktionen annähern sollen. Die approximierten Funktionen generalisieren von bereits besuchten/bekannten Zuständen auf unbekannte/noch nicht besuchte Zustände. In der Regeln ist die Anzahl der Parameter im Parametervektor θ viel kleiner als die Anzahl der Zustände des MDPs. Die Annäherung erfolgt durch die Modifikation des Parametervektors θ (wird auch oft als w bezeichnet). Dieser Parametervektor wird mittels Monte-Carlo oder Temporaler-Differenz gelernt. Im nächsten Abschnitt wird das gerade beschriebene Konzept der Funktionsapproximation auf die Strategiefunktion angewendet.

C. Strategiefunktionsapproximation

Eine Strategie π (eng. Policy) definiert das Verhalten des Agenten als Funktion von Zuständen s, mit $s \in S$ und S ist der gesamte Zustandsraum des MDPs. Bei allen Wertefunktionsbasierten Methoden wird eine Wertefunktion gelernt und mittels dieser kann implizit eine Strategie erstellt werden (z.B. mit $\epsilon - greedy$ vgl. [4]). Innerhalb dieser Arbeit werden speziell die Methoden erläutert, welche Strategien π_θ (bzw. den Parametervektor θ) direkt lernen (ausgenommen der Actor-Critic-Methode, diese verwendet sowohl Strategie- als auch Wertefunktionen). Eine stochastische approximierte Strategiefunktion wird als $\pi(s, a, \theta) = Pr\{a_t = a | s_t = s, \theta\}$ geschrieben. $\pi(s, a, \theta)$ wird oft auch als $\pi(s, a)$ geschrieben. Die Strategie ist stochastisch (nicht-deterministisch) d.h. sie beschreibt Wahrscheinlichkeiten für das Auswählen von Aktionen. Eine deterministische (nicht-stochastische) Strategie legt jeweils exakt eine Aktion für einen Zustand fest. Eine weitere Beschreibung der Strategiefunktion wäre $\pi_\theta(s, a) = P[a|s, \theta]$, d.h. $\pi_\theta(s, a)$ gibt die Wahrscheinlichkeitsverteilung über den möglichen Aktionen a im Zustand s an und durch den Parametervektor θ kann diese Wahrscheinlichkeitsverteilung verändert werden. Das optimieren des Parametervektors θ der Strategiefunktion π_θ ist Ziel des Policy Gradient Verfahrens und wird im nächsten Abschnitt ausführlich behandelt.

II. Policy Gradient

Die nachfolgenden Erläuterungen basieren auf der wissenschaftlichen Arbeit unter anderem von R. S. Sutton [2]. Der Agent soll eine approximierte optimale Strategie $\pi_\theta(s, a)$ mittels des Policy Gradienten Verfahrens erlernen. Für ein besseres Verständnis des Policy Gradient Verfahrens wird zuerst die allgemeine gradientenbasierte lokale Optimierung erläutert und anschließend wird erklärt was eine Zielfunktion ist und wie diese dargestellt werden kann. Die letzten beiden Teilabschnitte beschreiben die Anwendung des Gradientenverfahrens auf die Zielfunktion und eine Unterscheidung verschiedener Policy Gradient Methoden.

A. Gradientenbasierte lokale Optimierung

Definition eines Gradienten nach G. Hoever [3, vgl. S. 213] und D. Silver [4]: In einem Gradienten werden die verschiedenen partiellen Ableitungen einer Zielfunktion $J(\theta)$ zusammengefasst. Zu einer Zielfunktion $J : \mathbb{R}^n \to \mathbb{R}$ heißt (falls die partiellen Ableitungen existieren)

$$\nabla_\theta J(\theta) := grad\, J(\theta) := \left(\frac{\partial J(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n}\right)$$

Gradient von J im Punkt θ. (∇J wird "nabla J" gelesen.) Der Gradient einer Funktion weist in die Richtung des Steilsten Anstiegs. Senkrecht zum Gradienten ändert sich der Funktionswert nicht. G.Hoever [3, vgl. S. 214] liefert ebenfalls eine Erklärung des Gradientenverfahrens: Sucht man ausgehend von einem Startpunkt $\theta^{(0)}$ eine Maximalstelle (Gradient Ascent), so geht man ein Stück in die Richtung des Gradienten, also

$$\theta^{(1)} = \theta^{(0)} + \alpha_0 \cdot \nabla J(\theta^{(0)}),$$
$$\theta^{(2)} = \theta^{(1)} + \alpha_1 \cdot \nabla J(\theta^{(1)}),$$

allgemein:

$$\theta^{(i+1)} = \theta^{(i)} + \alpha_i \cdot \nabla J(\theta^{(i)}).$$

Dabei beschreibt α_i die Schrittweite (auch Lernrate genant). Sucht man eine Minimalstelle (Gradient Descent), so setzt man entsprechend

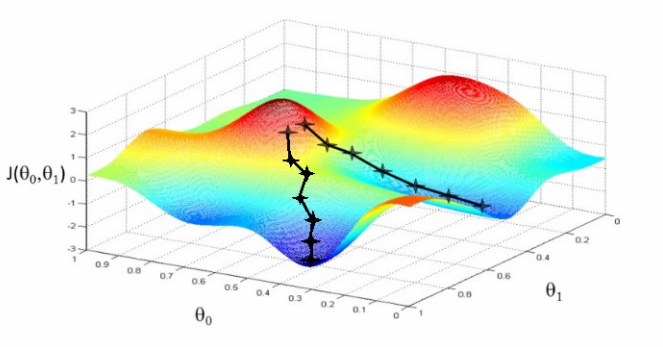

Fig. 2. Gradienten Abstiegsverfahren nach Andrew Ng [5].

$$\theta^{(i+1)} = \theta^{(i)} - \alpha_i \cdot \nabla J(\theta^{(i)}).$$

Angewendet wird dieses Verfahren u.a. bei der univariaten und multivariaten Regression (siehe maschinelles Lernen: vorhersage von kontinuierlichen Werten [6]). Die Kostenfunktion J berechnet die Qualität einer Hypothese h_θ (z.B. mittels dem Fehler der kleinsten Quadrate, eng. least-squared-errors). Eine Kostenfunktion J ist demnach eine Funktion der Hypothesen h_θ. Eine gute Veranschaulichung des Verfahrens liefert Andrew Ng: In Abbildung 2 ist ein dreidimensionaler Funktionsplot abgebildet. Die Z-Achse beschreibt den Funktionswert von $J(\theta_0, \theta_1)$. J ist eine Kostenfunktion von zwei Parametern θ und berechnet einen Kostenwert für verschiedene Werte dieser Parameter. Parameter θ_0 bezeichnet die X-Achse und Parameter θ_1 die Y-Achse. Die dunkelroten Stellen markieren Funktionsmaxima von J und die dunkelblauen Stellen markieren Funktionsminima. Das Gradienten Abstiegsverfahren findet für die Kostenfunktion J ein lokales Minimum (die lokal geringsten Kosten), jedoch kann das Verfahren für unterschiedliche Startpositionen unterschiedliche Minima finden.

Bei Problemen des Reinforcement Learnings ist es ganz ähnlich. Die Kostenfunktion der Hypothesen $J(h_\theta)$ ist, bei RL Problemen, als eine Gewinnfunktion $\rho(\pi_\theta)$ der erwarteten akkumulierten Belohnungen zu verstehen, d.h. $\rho(\pi_\theta)$ ist eine Funktion der Strategien und π_θ bezeichnet die verschiedenen Strategien. Die Strategien sind sozusagen die Hypothesen. Die Gewinnfunktion $\rho(\pi_\theta)$ ist die Zielfunktion des Agenten und der Agent versucht eine Strategie zu erlernen für die, die Zielfunktion maximal wird. Der Agenten verändert die Strategie mittels des Parametervektors θ unter Verwendung eines Gradienten Anstiegsverfahrens. Wie genau sieht die mit dem Gradienten Anstiegsverfahren zu maximierende Funktion $\rho(\pi)$ aus?

B. Zielfunktionen

Die Qualität (Performance) einer Policy berechnet die Zielfunktion (eng. objective function oder criterion) $\rho(\pi)$. R. S. Sutton [2] unterscheidet zwei verschiedene Formulierungen der Zielfunktion $\rho(\pi)$:

$$\begin{aligned}\rho_{avR}(\pi) &= \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\{r_1 + r_2 + ... + r_n|\pi\} \\ &= \sum_s d^\pi(s) \sum_a \pi(s,a) R_s^a,\end{aligned}$$

ist die durchschnittliche Belohnung, diese bewertet die Strategie bezüglich der zu erwartenden Langzeitbelohnung pro Schritt. Eine Langzeitbelohnung (eng. long-term reward) ist die Aufsummierung der Belohnungen für eine Entscheidungssequenz. Der Term $d^\pi(s) = \lim_{t\to\infty} Pr\{s_t = s|s_0, \pi\}$ beschreibt die stationäre Verteilung der Zustände unter π, d.h. $d^\pi(s)$ gibt an, wie hoch die Wahrscheinlichkeit dafür ist, jeden Zustand s_t zu erreichen, bedingt durch die Policy π und unabhängig vom Startzustand s_0, wenn t gegen unendlich strebt. Sprachliche Formulierung der Funktion $\rho_{avR}(\pi)$: Es existiert eine Wahrscheinlichkeit, dass sich der Agent in Zustand s befindet und es existiert eine Wahrscheinlichkeit, dass der Agent eine Aktion a auswählt (a abhängig von π). Berechnet wird die durchschnittliche Belohnung pro Zeitschritt, wobei die Belohnung R_s^a von den beiden vorher erwähnten Wahrscheinlichkeiten s und a abhängt. Der Wert eines Zustands-Aktionspaares, unter Berücksichtigung einer Policy π, ist für die Zielfunktion der durchschnittlichen Belohnung $\rho_{avR}(\pi)$ wie folgt definiert:

$$Q^\pi(s,a) = \sum_{t=1}^{\infty} \mathbb{E}\{r_t - \rho_{avR}(\pi)|s_0 = s, a_0 = a, \pi\},$$

mit $\forall s \in S, a \in A$. Die Funktion berechnet die Summe der Differenzen zwischen einer sofortigen Belohnung (r_t) in jedem Zeitschritt t und der Zielfunktion $\rho_{avR}(\pi)$, also der durchschnittlichen Belohnung pro Zeitschritt, für ein Zustands-Aktionspaar unter Verwendung einer Policy π. R. S. Sutton notiert die Zielfunktion $\rho_{avR}(\pi)$ als $\rho(\pi)$, in dieser Arbeit wird für die durchschnittliche erwartete Belohnung pro Zeitschritt die Notation $\rho_{avR}(\pi)$ verwendet (für **av**erage **R**eward ähnlich D. Silver [4]). Die zweite Formulierung der Zielfunktion $\rho(\pi)$ berücksichtigt einen genau festgelegten Startzustand s_0 und ausschließlich von diesem Startzustand ausgehende Langzeitbelohnungen:

$$\rho_{s_0}(\pi) = \mathbb{E}\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t|s_0, \pi\}$$

und

$$Q^\pi(s,a) = \mathbb{E}\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k}|s_t = s, a_t = a, \pi\}.$$

Dabei ist $\gamma \in [0,1]$ ein Abschwächungsfaktor ($\gamma = 1$ ist nur in endlichen abzählbaren Sequenzen einzusetzen). Die stationäre Verteilung der Zustände $d^\pi(s)$ ist bei der Startzustandsformulierung eine abgeschwächte Gewichtung der vorgefundenen Zustände angefangen in Zustand s_0 und unter Berücksichtigung der Policy π: $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr\{s_t = s|s_0, \pi\}$.

C. Bedeutung des Policy Gradient

Das Policy Gradient Verfahren nach unter anderem R. S. Sutton [2] und D. Silver [4] funktioniert wie folgt: Sei $\rho(\theta)$ eine beliebige Strategiezielfunktion. Policy Gradient Algorithmen suchen ein lokales Maximum der Strategiefunktion $\rho(\theta)$, mittels des ansteigenden Gradienten der Strategie, bezüglich der Parameter θ:

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \approx \alpha \nabla_\theta \rho(\theta).$$

Der Gradient der Strategiezielfunktion $\rho(\theta)$ ist der Vektor der partiellen Ableitungen:

$$\nabla_\theta \rho(\theta) = \frac{\partial \rho}{\partial \theta} = \begin{pmatrix} \frac{\partial \rho(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \rho(\theta)}{\partial \theta_n} \end{pmatrix}.$$

D. Monte-Carlo Policy Gradient

Das Monte-Carlo Policy Gradient Verfahren nach R. S. Sutton [2] und D. Silver [4] berechnet den Gradienten der Strategiefunktion analytisch ohne eine Wertefunktion zu verwenden.

Softmax Policy: ist eine alternative Darstellung der Strategiefunktion $\pi_\theta(s, a)$ im Gegensatz zu ϵ-geedy. Der Ausdruck $\phi(s, a)\top\theta$ steht für eine Gewichtung von Aktionen, welche lineare Kombination von Merkmalen verwenden. Dieser Ausdruck ergibt einen Wert und der Wert gibt an, wie sehr die konkrete Aktion a bevorzugt ausgewählt werden soll. $\phi(s, a)$ ist das ausgewählte Merkmal einer Aktion in einem Zustand und θ ist die Gewichtung dieses Merkmals. Softmax Policy wird ausschließlich für diskrete Aktionsräume verwendet und nicht bei kontinuierlichen Aktionsräumen. Um den Ausdruck $\phi(s, a)\top\theta$ in eine Wahrscheinlichkeit umzuwandeln, wird dieser exponenziert und normalisiert.

$$\pi_\theta(s, a) \propto e^{\phi(s,a)\top\theta}$$

d.h. die Wahrscheinlichkeit das die Aktion a wirklich ausgewählt wird, ist Proportional zum exponenzierten Wert der gewichteten linearen Kombination der Merkmale. Eine lineare Kombination von Merkmalen in z.B. einem Labyrinth mit einem diskreten Aktionsraum von Hoch, Runter, Links und Rechts: Jede der Vier Aktionen erhält Merkmale, je höher die Punktzahl (eng. score) der Merkmale ist, wenn die gewichtete Summe der linearen Merkmale berechnet wird, desto höher ist die Wahrscheinlichkeit die Aktion des Merkmals zu wählen.

Gradient der Softmax Policy (Softmax Score Funktion): ist die Differenz des Merkmals der tatsächlich ausgewählten Aktion und dem durchschnittlichen Merkmal für alle Aktionen die hätten ausgewählt werden können:

$$\nabla_\theta \log \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}[\phi(s, \cdot)]$$

d.h. die Parameter der Strategiefunktion werden in die Richtung des Gradienten angepasst, wenn das Merkmal der tatsächlich ausgewählten Aktion, im Gegensatz zu allen anderen Merkmalen und deren Aktionen, öfter auftritt (hohe Punktzahl des Merkmals) und eine gute Belohnung erhält. Allgemein ist der Gradient von $\pi_\theta(s, a)$ gleich $\pi_\theta(s, a)\nabla_\theta \log \pi_\theta(s, a)$ nach der Berechnung:

$$\begin{aligned} \nabla_\theta \pi_\theta(s, a) &= \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)} \\ &= \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a). \end{aligned}$$

Einschritt MDP: ist keine Entscheidungssequenz, sondern ein einziger Zeitschritt in einem MDP. Starte in einem Zustand $s \sim d(s)$, wähle eine Aktion a entsprechend der Strategiefunktion $\pi_\theta(s, a)$ und erhalte eine Belohnung definiert durch die Funktion R_s^a:

$$J(\theta) = \mathbb{E}[R_{s,a}] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(s, a) R_{s,a}.$$

Die Schreibweise $J(\theta)$ der Zielfunktion $\rho_{avR}(\theta)$ ist von D. Silver [4], daher $J(\theta) \sim \rho_{avR}(\pi)$. Der Gradient dieser Zielfunktion ist entsprechend:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) R_{s,a} \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) R_{s,a}], \end{aligned}$$

d.h. einzig die Policy $\pi_\theta(s, a)$ ist abhängig von θ, darum muss der Gradient nur von dieser Policy gebildet werden. Dieser Aktualisierungsschritt benötigt kein Modell, denn die Policy $\pi_\theta(s, a)$ ist bekannt. $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) R_{s,a}]$ bedeutet: Der Erwartungswert $\mathbb{E}$ unterliegt einer Strategiefunktion π_θ, sprich er ist abhängig von einem Startzustand s und einer ausgewählten Aktion a der Strategiefunktion $\pi_\theta(s, a)$. Der durch die Strategiefunktion bedingte Erwartungswert wird berechnet durch den Erwartungswert von $\nabla_\theta \log \pi_\theta(s, a)$ (Punktzahl) multipliziert mit der sofortigen Belohnung (eng. immediate reward) $R_{s,a}$.

Policy Gradient Theorem: ist die Substitution der sofortigen Belohnung $R_{s,a}$ mit der Langzeitbelohnung $Q^{\pi_\theta}(s, a)$ aus der Berechnung des Gradienten der Zielfunktion:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)],$$

Das Policy Gradient Theorem verallgemeinert das Einschritt MDP auf eine Sequenz von Zeitschritten (Mehrfachschritt MDP). Das Theorem ist auf die Startzustandszielfunktion und auf die durchschnittliche Langzeitbelohnung Zielfunktion anwendbar. Ein Nachteil dieses Verfahrens ist die relativ hohe Varianz. Das Actor-Critic Policy Gradient Verfahren (detaillierte Informationen von u.a. R. S. Sutton [1] und D. Silver [4]) verwendet eine Strategiefunktionen und eine Wertefunktionen für das lernen/aktualisieren der Parametervektoren und verringert somit die Varianz. Das Actor-Critic Verfahren ist sehr fortgeschritten und wird oft in der Praxis eingesetzt. Auf eine genaue Erläuterung dieses Verfahrens muss auf Grund des begrenzten Umfangs dieser Arbeit verzichtet werden.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2rd ed. Cambridge, Massachusetts, London, England: MIT Press, 2012.

[2] R. S. Sutton and D. McAllester and S. Singh and Y. Mansour, *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, 180 Park Avenue, Florham Park, NY 07932: AT&T Labs, 1999.

[3] G. Hoever, *Höhere Mathematik kompakt*, 2. Auflage, Springer Spektrum, 2014.

[4] D. Silver, *Online Course Reinforcement Learning*, http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html, London, England: Googel Deep Mind, 2015.

[5] A. Ng, *01 and 02: Introduction, Regression Analysis and Gradient Descent*, http://www.holehouse.org/mlclass, Stanford University Lectures, 2017.

[6] I. Goodfellow and Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.