(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. Einführung

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

II. 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. Sutton und Barto [1] definieren die Umgebung als einen Markov Entscheidungsprozess (eng. **M**arkov **D**ecision **P**rocess). Der Agent interagiert demnach mit einem MDP und erhält von diesem einen Zustand und eine Belohnung. Der MDP erhält vom Agenten eine ausgewählte Aktion.

A. Markov Decision Process

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 $R_s^a = E\{r_{t+1}|s_t = s, a_t = a\}$. 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)

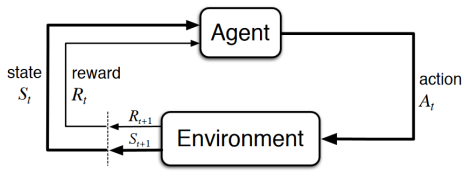

Fig. 1. Die Interaktion zwischen Agent und Umgebung nach [1].

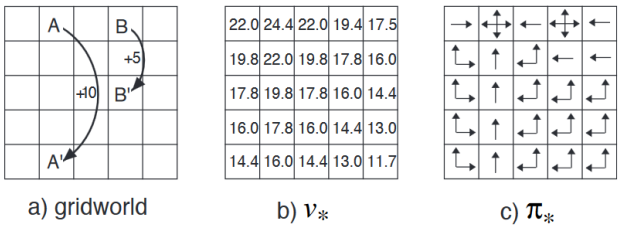

Fig. 2. Optimale Lösungen des Rasterwelt Beispiels nach [1].

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 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. Die Bewertungsfunktion (Value Function)

Eine Bewertungsfunktion V Bewertet wie gut jeder Zustand und/oder jede Aktion ist.

Dynamische Programmierung, Monte-Carlo-Methoden, Temporales-Differenzlernen

C. Die Strategie (Policy π)

Die Policy ist eine Prozedur, welche die Entscheidungsfindung des Agenten festlegt. Eine Policy π definiert das Verhalten des Agenten als Funktion von Zuständen s, mit $s \in S$ und S ist der gesamte Zustandsraum. Bei allen tabellarischen Aktion-Wert Methoden wird eine Wertefunktion gelernt und mittels dieser kann implizit eine Policy erstellt werden ($\epsilon - greedy$). Innerhalb dieser Arbeit werden speziell die Methoden erläutert, welche die Policy (bzw. die Policy Parameter) direkt lernen. Eine stochastische angenäherte Policy wird als $\pi(s, a, \theta) = Pr\{a_t = a|s_t = s, \theta\}$ geschrieben. Die Policy ist stochastisch (nicht-deterministisch) d.h. sie beschreibt Wahrscheinlichkeiten für das Auswählen von Aktionen. Eine deterministische (nicht-stochastische) Policy legt jeweils exakt eine Aktion für einen Zustand fest. Die Policy ist angenähert (approximiert), aus Gründen der Rechenzeitverbesserung. θ beschreibt einen Parametervektor durch den die Funktionsapproximation realisiert werden kann. Das optimieren dieses Parametervektors θ ist Ziel des Policy Gradient Verfahrens. Warum wird die Policy Approximiert? Der Grund dafür ist das Dimensionsproblem. Bei tabellarischen Aktion-Wert Methoden steigt die Größe der Tabelle exponentiell mit der Größe der Zustands- und Aktionsräume. Über die Tabelle muss annähernd unendlich oft iteriert werden, um eine optimale Strategie zu erlernen. Lernen mit tabellarischer Wertefunktionsdarstellung ist ausschließlich für sehr einfache niedrigdimensionale Probleme sinnvoll. Um auch komplexere Probleme lösen zu können wird auf eine exakte Berechnung der optimalen Strategie verzichtet und eine annähernd optimale Strategie wird, aus gründen enormer Rechenzeiteinsparungen, bevorzugt. Eine Strategie ist optimal, wenn die Summe der erwarteten Belohnungen maximal ist, gegenüber allen anderen möglichen Strategien. Es können auch mehrere Strategien optimal sein, dann ist der aufsummierte Wert der erwarteten Belohnungen, für jede dieser Strategien, gleich hoch.

III. Policy Gradient Theorem

Die nachfolgenden Erläuterungen basieren auf der wissenschaftlichen Arbeit unter anderem von R. S. Sutton [2]. Der Agent soll eine annähernd optimale Policy mittels Funktionsapproximation und Policy Gradienten Verfahren erlernen.

A. Gradientenbasierte lokale Optimierung

Eine Methode dies zu erreichen ist die Maximierung einer Zielfunktion $\rho(\pi)$, mittels eines Gradienten Anstiegsverfahrens. Eine Zielfunktion berechnet die Qualität einer Policy. Definition eines Gradienten nach G. Hoever [3, vgl. S. 213]: In einem Gradienten werden die verschiedenen partiellen Ableitungen einer Funktion $f(x)$ zusammengefasst. Zu einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ heißt (falls die partiellen Ableitungen existieren)

$$\nabla f(x) := grad\, f(x) := \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right)$$

Gradient von f im Punkt x. (∇f wird "nabla f" 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 $x^{(0)}$ eine Maximalstelle (Gradient Ascent), so geht man ein Stück in die Richtung des Gradienten, also

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

allgemein:

$$x^{(i+1)} = x^{(i)} + \alpha_i \cdot \nabla\, f(x^{(i)}).$$

Dabei beschreibt α_i die Schrittweite. Die genaue Wahl dieser Schrittweite ist oft nicht ganz einfach, wird an dieser Stelle jedoch nicht näher erläutert. Sucht man eine Minimalstelle (Gradient Descent), so setzt man entsprechend

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

Bei der univariaten und multivariaten Regression (siehe maschinelles Lernen: vorhersage von kontinuierlichen Werten [5]) wird das Gradienten Abstiegsverfahren verwendet, um eine Kostenfunktion zu minimieren. Diese Kostenfunktion berechnet die Qualität einer Hypothese (z.B. mittels dem Fehler der kleinsten Quadrate, eng. least-squared-errors). Bei Problemen des Reinforcement Learnings ist die Kostenfunktion definiert durch eine Zielfunktion $\rho(\pi)$ und der Agent versucht eine Policy zu erlernen für die, die Zielfunktion maximal wird. Der Agenten verändert die Policy mittels des Parametervektors θ unter Verwendung eines Gradienten Anstiegsverfahrens. Vielmehr ist die Kostenfunktion bei RL Problemen eher als Gewinnfunktion oder Belohnungsfunktion zu bezeichnen, da diese maximiert und nicht minimiert werden soll. Wie genau sieht die mit dem Gradienten Anstiegsverfahren zu maximierende Funktion $\rho(\pi)$ aus?

B. Zielfunktion

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 $\rho(\pi)$:

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

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:

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

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:

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

und

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

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. Policy Gradient

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \tag{1}$$

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) \tag{2}$$

D. Policy Gradient mit Funktionsapproximation

-Kombination linearer Eigenschaften -Neuronale Netze
$V_\theta(s) \approx V^\pi(s)$
$Q_\theta(s,a) \approx Q^\pi(s,a)$
$\pi_\theta(s,a) = P[a|s,\theta]$

IV. Policy Gradient Methoden

A. Finite Difference Policy Gradient

B. Monte-Carlo Policy Gradient (REINFORCE)

C. Actor-Critic Policy Gradient

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