

(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 L^AT_EX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) *Subsubsection Heading Here*: Subsubsection text here.

II. ERGEBNIS

The conclusion goes here.

III. REINFORCEMENT LEARNING (RL)

Für ein besseres Verständnis des Policy Gradient Verfahrens wird nachfolgend ein Überblick der wichtigsten Grundlagen des Reinforcement Learnings (RL) gegeben.

Sutton und Barto [1] definieren ein Standard RL-Framework, in diesem interagiert ein Agent mit einer Umgebung. Der Agent lernt, in einer ihm unbekannten Umgebung, durch Versuch und Irrtum eine Strategie. 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 definieren die Umgebung als einen Markov Entscheidungsprozess (eng. Markov Decision Process). 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 $\dots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2} \dots$ beschreibt die Interaktion des Agenten mit dem MDP.

Ein Zustand ist eine Darstellung der Umwelt zu einem Zeitpunkt t . Die Zustände bestimmen, 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 . 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 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 bestimmen die Positionen auf die Spielsteine gesetzt werden können. 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.

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 $R(s, a)$ oder R_s^a . Diese

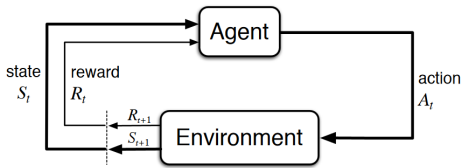


Fig. 1. Die Interaktion zwischen Agent und Umgebung nach [1].

Funktion ist eine Abbildung von Zustand-Aktionspaaren auf erwartete Belohnungen.

B. Die Strategie (Policy π)

Definiert das Verhalten des Agenten als Funktion. Maximierung der Summe der erwarteten Belohnungen entspricht der Findung einer optimalen Strategie.

Policy-Based Reinforcement Learning: π is life!

C. Die Bewertungsfunktion (Value Function)

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

Value-Based Reinforcement Learning:

This is Goodfellow: [5].

$$V_\theta(s) \approx V^\pi(s)$$

$$Q_\theta(s, a) \approx Q^\pi(s, a)$$

IV. FUNKTIONSAPPROXIMATION

-Kombination linearer Eigenschaften -Neuronale Netze

V. GRADIENTENBASIERTE OPTIMIERUNG

Der Agent soll die Policy π mittels Erfahrung und daraus resultierender Belohnung und Bestrafung optimieren.

Ein Methode die Funktion $\pi(s) = a$ zu minimieren, ist das Gradienten Abstiegsverfahren (Gradient Decent).

Definition eines Gradienten nach G. Hoever [3, vgl. S. 213]:

In einem Gradienten werden die verschiedenen partiellen Ableitungen einer Funktion $f(x)$ zusammengefasst.

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \quad (1)$$

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix} \quad (2)$$

VI. 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 $\rho(\pi)$:

$$\begin{aligned} \rho_{avR}(\pi) &= \lim_{n \rightarrow \infty} \frac{1}{n} E\{r_1 + r_2 + \dots + r_n | \pi\} \\ &= \sum_s d^\pi(s) \sum_a \pi(s, a) R_s^a, \end{aligned} \quad (3)$$

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 $d^\pi(s) = \lim_{t \rightarrow \infty} Pr\{s_t = s | s_0, \pi\}$ 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. Die zweite Formulierung der Zielfunktion $\rho(\pi)$ berücksichtigt einen genau festgelegten Startzustand s_0 und ausschließlich von diesem Startzustand ausgehende Langzeitbelohnungen:

$$\rho_{s_0}(\pi) = E\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0, \pi\right\}. \quad (4)$$

VII. POLICY GRADIENT METHODEN

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \quad (5)$$

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s, a)}{\partial \theta} Q^\pi(s, a) \quad (6)$$

$$Q^\pi(s, a) = \sum_{t=1}^{\infty} E\{r_t - \rho(\pi) | s_0 = s, a_0 = a, \pi\} \quad (7)$$

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*, 2nd 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: Google Deep Mind, 2015
- [5] I. Goodfellow and Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.