

# Soft Actor Critic

Oliver Chmurzynski, Leon Bütinghaus, Thilo Röthemeyer

21. April 2021

# Contents

## 1 Soft Actor-Critic Grundlagen

- Grundlegender Aufbau
- Soft Policy Iteration

## 2 Soft Actor-Critic im kontinuierlichen Raum

- SAC Grundprinzip
- SAC Update Regeln

## 3 Ergebnisse

- Vergleich mit anderen Algorithmen
- Zusammenfassung

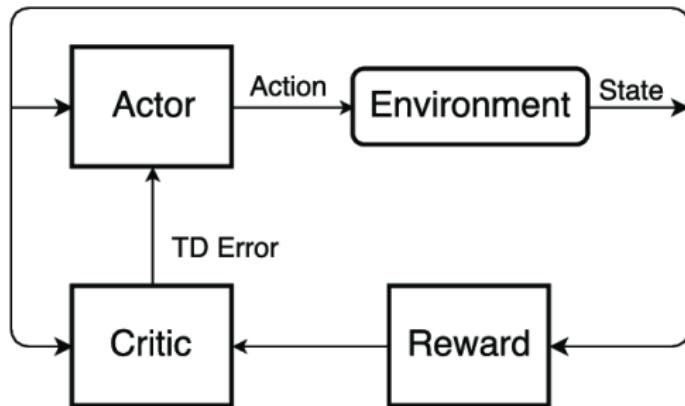
## 4 Literaturverzeichnis

## Probleme die bei RL Algorithmen auftreten

- On-Policy Algorithmen haben eine niedrige Sample Effizienz
- Off-Policy Algorithmen sind oft instabil und benötigen eine genaue Anpassung der Hyperparameter
- Soft Actor-Critic nutzt einen Off-Policy Ansatz und verbessert die Stabilität

# Actor-Critic

- Soft Actor Critic nutzt einen Actor-Critic Ansatz
- Actor lernt eine Policy
- Critic lernt eine Value Function



# Maximierung der Entropie

- Standard Reinforcement Learning maximiert die Belohnung

$$\sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t)]$$

- Soft Actor-Critic maximiert zusätzlich noch die Entropie

$$\sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))]$$

## Soft Policy Iteration

- Soft Policy Iteration ist die Basis für Soft Actor-Critic
- Benötigt Problem in tabellarischer Form
- Evaluiert und verbessert abwechselnd die Policy

# Policy Evaluation

- Q-Values werden iterativ durch die Anwendung eines modifizierten Bellman backup operators  $\mathcal{T}^\pi$  berechnet

$$\mathcal{T}^\pi Q(s_t, a_t) \triangleq r(s_T, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V(s_{t+1})]$$

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \log \pi(a_t | s_t)]$$

# Policy Improvement

- Policy wird an die neue Q-Funktion angepasst
- Mit Hilfe der Kullback-Leibler Divergenz wird die Policy auf eine gaußsche Verteilung projiziert

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left( \pi'(\cdot | s_t) \parallel \frac{\exp(Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right)$$

## Kontinuierlicher Aktionsraum

- große kontinuierliche Aktionsräume
  - ⇒ Approximation der Soft Policy Iteration
- Schritt von Tabellen zu DNNs
- Optimierung mittels gradient descent

# Funktionen und deren Netzwerke

- Q-Function:

$Q_\theta(s_t, a_t) \rightarrow$  Q-Value als Ausgabe

- Policy:

$\pi_\phi(s_t | a_t) \rightarrow$  Mittelwert und Kovarianz als Ausgabe  $\Rightarrow$  Gauss

Mit Parametervektoren  $\theta$  und  $\phi$

# Value Function

- benötigt für Q-Function
- eigenes Netzwerk für Value-Function kann
  - Training stabilisieren
  - simultanes Training aller Netzwerke möglich machen
- hier: kein eigenes Value Netzwerk
- Berechnung über Value-Function aus Soft Policy Iteration

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t | s_t)]$$

# Optimierung Q-Funktion

- Minimierung des Fehlers
- Fehlerfunktion: soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim D} \left[ \frac{1}{2} (Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_{\bar{\theta}}(s_{t+1})]))^2 \right]$$

mit parametrisierter Value-Function  $V_{\bar{\theta}}$

- Training über gradient descent

# Optimierung der Strategie

- Minimierung des Fehlers
- Fehlerfunktion: KL-Divergenz

$$J_{\pi}(\phi) = \mathbb{E}_{s_t \sim D} [\mathbb{E}_{a_t \sim \pi_{\phi}} [\alpha \log(\pi_{\phi}(a_t | s_t)) - Q_{\theta}(s_t, a_t)]]$$

- für vereinfachtes Training: reparameterization trick

# Reparameterization trick

- Ziel: statistisches Netzwerk  $\Rightarrow$  deterministisches Netzwerk
- $f_\phi$ : vektorwertige Abbildung auf Aktionsraum, mit Parameter  $\phi$
- $\epsilon$ : noise Vektor, aus fixer Verteilungsfunktion, z.B. Gauss

$$a_t = f_\phi(\epsilon_t; s_t)$$

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim N} [\alpha \log \pi_\phi(f_\phi(\epsilon_t; s_t) | s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))]$$

- Training über gradient descent

## $\alpha$ -Training

- passenden temperature Wert  $\alpha$  zu finden ist schwer
- Entropie verändert sich, je genauer die policy wird
  - ⇒ optimale Aktion unklar: mehr exploration
  - ⇒ optimale Aktion klar: mehr exploitation
- Lösung:  $\alpha$  wird über gradient descent trainiert
- Fehlerfunktion für  $\alpha$ :

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t} [-\alpha \log \pi_t(a_t | s_t) - \alpha \mathcal{H}]$$

# Algorithmus (1/2)

- Benutzung zweier Q-Funktionen  
⇒ verschnellert Training bei komplizierten Problemen
- Training zweier Parameter  $\theta_1$  und  $\theta_2$
- Gradientenberechnung mit  $\min\{Q_{\theta_1}, Q_{\theta_2}\}$
- target Q-Netzwerk wird geupdatet

## Algorithmus (2/2)

### Algorithm 1: Soft Actor-Critic

Input  $\theta_1, \theta_2, \phi$

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2, \mathcal{D} \leftarrow \emptyset$

**for each iteration do**

**for each environment step do**

$a_t \sim \pi_\phi(a_t | s_t), s_{t+1} \sim p(s_{t+1} | s_t, a_t)$   
 $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$

**end**

**for each gradient step do**

$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i) \text{ for } i \in \{1, 2\}$   
 $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$   
 $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$   
 $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \bar{\theta}_i \text{ for } i \in \{1, 2\}$

**end**

**end**

## Ziel der Experimente

- Stabilität und Sample Komplexität im Vergleich zu anderen Algorithmen
  - Kontinuierliche Aufgaben
  - Verschiedene Schwierigkeitgrade
- OpenAI gym und rllab

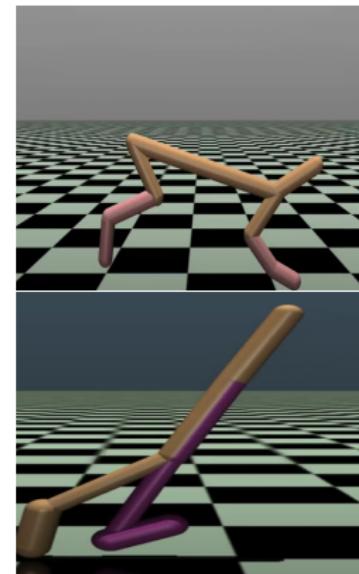
# Vergleich zu anderen Algorithmen

- SAC
  - Durchschnittswert (mean action)
  - Feste und variable Temperatur (Anpassung im neuen Paper)
- PPO, DDPG
  - Kein Exploration noise
- TD3
- SQL mit zwei Q Funktionen
  - Evaluation mit Exploration noise

## Vergleich zu anderen Algorithmen

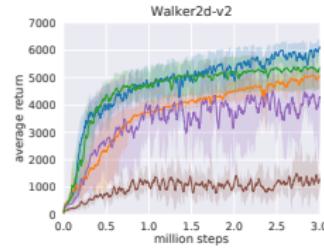
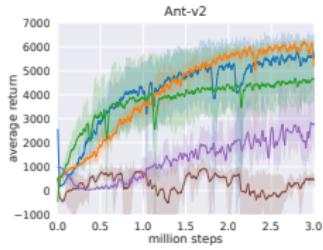
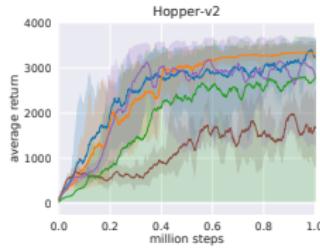
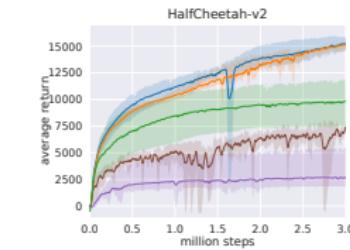
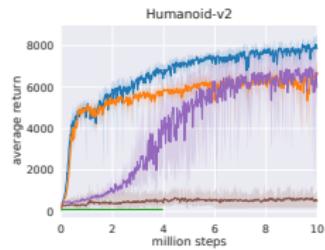
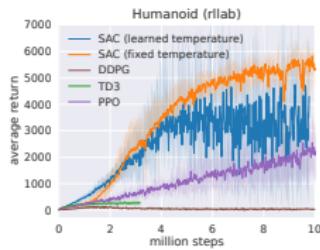
- 5 Instanzen mit einer Evaluation alle 1000 Schritte
- Schattierter Verlauf zeigt min und max der fünf Durchläufe

# Ergebnisse



[Kir18]

# Ergebnisse



[?]

## Zusammenfassung

- Soft actor critic vorgestellt
  - Off policy Algorithmus
  - Entropiemaximierung verbessert Stabilität
  - Besser als state-of-the-art Algorithmen
  - Gradientenbasiertes Temperatur Tuning



**Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine.**

Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

*CoRR*, abs/1801.01290, 2018.



**Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine.**

Soft actor-critic algorithms and applications.

*CoRR*, abs/1812.05905, 2018.



**Sam Kirkles.**

Halfcheetah.

Youtube, 2018.



**Diederik P Kingma and Max Welling.**

Auto-encoding variational bayes, 2014.



**Andrew Trask.**

*Grokkking Deep Learning.*

Manning Publications Co., USA, 1st edition, 2019.