technische universität dortmund

fakultät für informatik

Soft Actor Critic

Oliver Chmurzynski, Leon Büttinghaus, Thilo Röthemeyer

20. April 2021

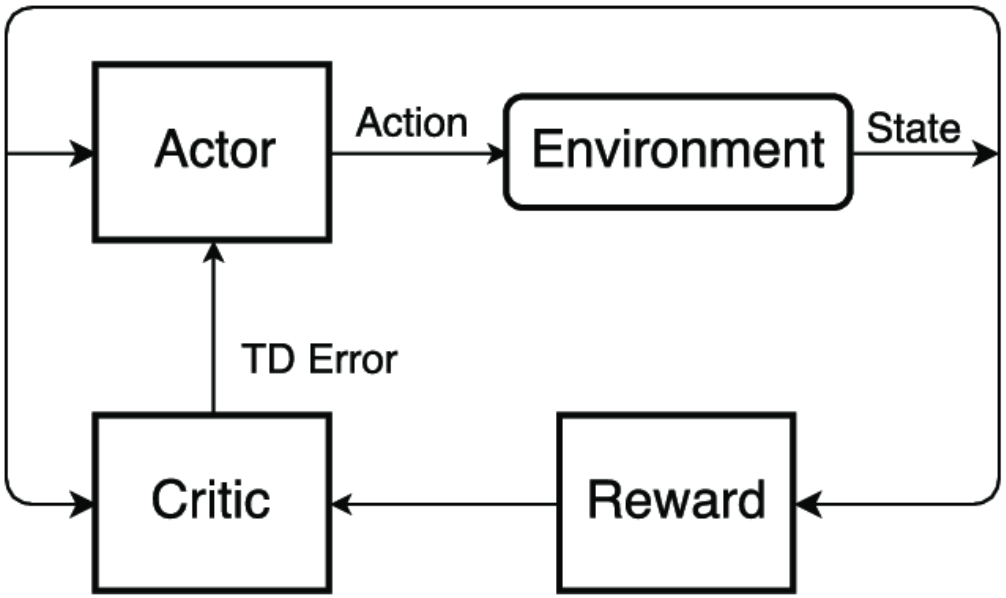
Contents

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 Reinforcment Lerning 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

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

Kontinuierlicher Aktionsraum

- kontinuierliche Aktionsräume benötigen

  $\Rightarrow$ Approximation für Q-Funktion

  $\Rightarrow$ Approximation für Strategie

- Schritt von Tabellen zu DNNs
- Optimierung mittels gradient descent

Funktionen und deren Netzwerke

- State Value Funktion:

  $V_\psi(s_t)$ $\rightarrow$ Skalar als Ausgabe

- Q-Funktion:

  $Q_\theta(s_t, a_t)$ $\rightarrow$ Skalar als Ausgabe

- Strategie:

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

Mit Parametervektoren ψ, θ und ϕ

State Value Funktion

- eigenes Netzwetk nicht notwendig, aber
  - stabilisiert Training
  - macht simultanes Training aller Netzwerke möglich
- somit: Berechnung des state values über eigenes Netzwerk

Optimierung State Value Funktion

- Minimierung des Fehlers
- Fehler: Residuenquadratsumme aus state-value und Erwartungswert

$$J_V(\psi) = \mathbb{E}_{s_t \sim D}\left[\frac{1}{2}(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi}[Q_\theta(s_t, a_t) - log\pi_\phi(s_t|a_t)])^2\right]$$

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - Q_\theta(s_t, a_t) + log\pi_\phi(s_t|a_t))$$

Optimierung Q-Funktion

- Minimierung des Fehlers
- Fehler: soft Bellman Restwert

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}(Q_\theta(s_t,a_t) - \hat{Q}_\theta(s_t,a_t))^2\right]$$

mit $\hat{Q}(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\psi}}(s_{t+1})]$

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t,s_t)(Q_\theta(s_t,a_t) - r(s_t,a_t) - \gamma V_{\overline{\psi}}(s_{t+1}))$$

Optimierung der Strategie

- Minimierung des Fehlers
- Fehler: KL-Divergenz

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D}\left[D_{KL}\left(\pi_\phi(\cdot|s_t)\middle\|\frac{exp(Q_\theta(s_t,\cdot))}{Z_\theta(s_t)}\right)\right]$$

- zur Berechnung des Gradienten: reparameterization trick

Reparameterization trick (1/2)

- Ziel: neue Parameter des Erwartungswertes

  $\Rightarrow$ umschreiben der Zielfunktion und des Gradienten

- f_ϕ: vektorwertige Abbildung auf Aktionsraum, mit Parameter ϕ
- ϵ: Vektor, aus fixer Vertilungsfunktion, z.B. Gauss

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

neuer Parameter für Erwartungswert: $\mathbb{E}_{s_t \sim D} \Rightarrow \mathbb{E}_{s_t \sim D, \epsilon_t \sim N}$

Reparameterization trick (2/2)

- Zielfunktion kann umgeschrieben werden:

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

- Gradient kann geschätzt werden:

$$\hat{\nabla}_\phi J_\pi(\phi) =$$

$$\nabla_\phi log\pi_\phi(a_t|s_t) + (\nabla_{a_t} log\pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t))\nabla_\phi f_\phi(\epsilon_t; s_t)$$

Algorithmus

Algorithm 1: Soft Actor-Critic

Initialize parameter vectors $\psi, \overline{\psi}, \theta, \phi$

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**

    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$

    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$

    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$

    $\overline{\psi} \leftarrow \tau\psi + (1 - \tau)\overline{\psi}$

  end

end

Algorithmus

Algorithm 2: Soft Actor-Critic

Input θ_1, θ_2, ϕ

$\overline{\theta_1} \leftarrow \theta_1$, $\overline{\theta_2} \leftarrow \theta_2$, $\mathcal{D} \leftarrow \emptyset$

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**

    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$

    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$

    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$

    $\overline{\theta_i} \leftarrow \tau\theta_i + (1 - \tau)\overline{\theta_i}$ 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

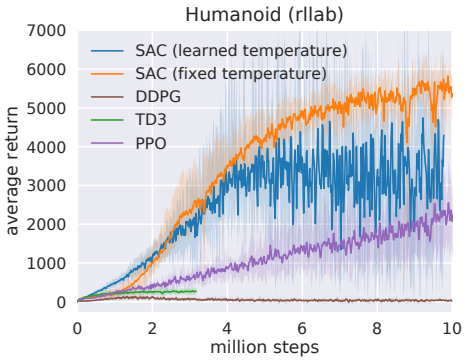
Vergleich zu anderen Algorithmen

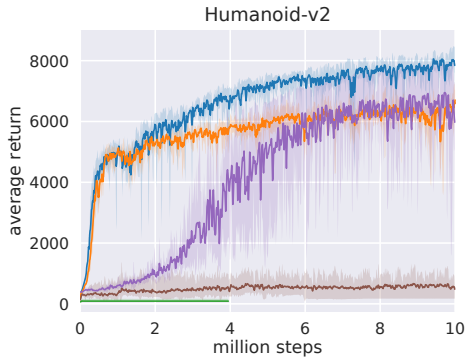
- SAC
  - Durchschnittswert (mean action)
  - Feste und variable Temperatur (Anpassung im neuen Paper)
- PPO, DDPG
  - Kein Exploration noise
- TD3
- SQL mit zwei Q Funktionen
  - Evaluation mit Exploration noise

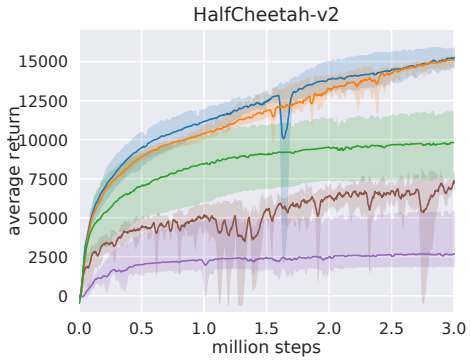
Vergleich zu anderen Algorithmen

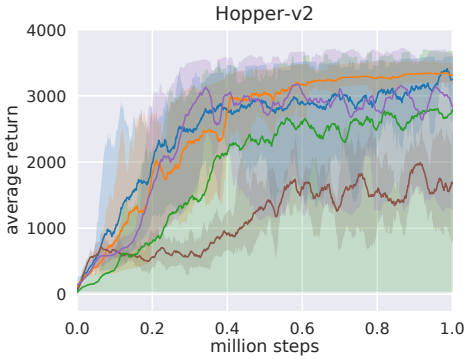
- 5 Instanzen mit einer Evaluation alle 1000 Schritte
- Schattierter Verlauf zeigt min und max der fünf Durchläufe

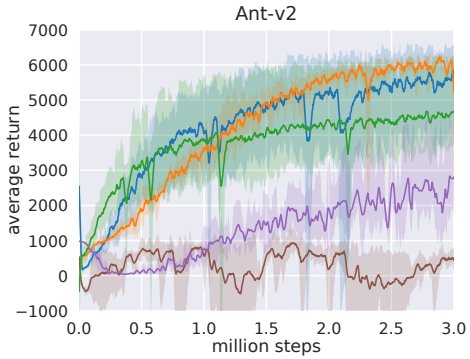
Ergebnisse

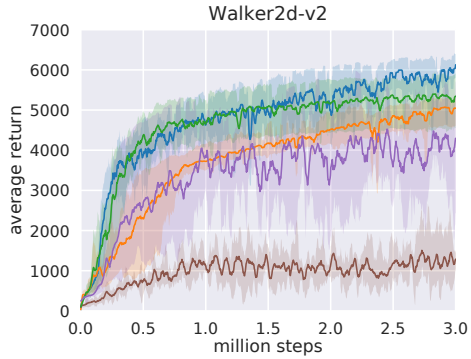

Humanoid (rllab)
SAC (learned temperature)
SAC (fixed temperature)
DDPG
TD3
PPO
average return
million steps


Humanoid-v2
average return
million steps


HalfCheetah-v2
average return
million steps


Hopper-v2
average return
million steps


Ant-v2
average return
million steps


Walker2d-v2
average return
million steps

[?]

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.

Diederik P Kingma and Max Welling.

Auto-encoding variational bayes, 2014.

Andrew Trask.

Grokking Deep Learning.

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