

Reinforcement Learning

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2026-01 Dozent: Karsten Flügge

Themen des Tages

Tag 10

Multi-Agent RL

S. 259–299



Multi-Agent RL

Multi-Agent Reinforcement Learning (MARL)

Szenarien mit mehreren **Agenten**, die in einer gemeinsamen Umgebung agieren.

Diese Agenten können **kooperieren**, **konkurrieren** oder eine **Mischung** aus beidem zeigen.

Jeder Agent optimiert seine **eigene Policy** unter Berücksichtigung anderer Agenten.

Die **Umgebung** ist **nicht mehr stationär**, da sich Policies anderer Agenten kontinuierlich ändern.

MARL ist besonders relevant in komplexen, dynamischen Systemen wie autonomen **Fahrzeugflotten**, Mehrspieler-**Spiele**n oder verteilten **Robotiksystemen**.

Es geht also nicht nur um das mathematisch/algorithmische Einfügen von mehreren Agenten sondern teilweise tatsächlich um **diskrete physikalische** Agenten.

Wie kann eine Policy Rückwirkung auf eine Umgebung haben?

Multi-Agent RL

- **Vorteile:**

- Modellierung komplexer, realitätsnaher Systeme.
- Ermöglicht **emergentes Verhalten** (z.B. Schwarmintelligenz).

- **Herausforderungen:**

- **Instabilität** durch sich verändernde Umgebung.
- **Skalierbarkeit**
- **Koordination**
- Teilweise Beobachtbarkeit und **Kommunikationsrestriktionen**.

- **Beispiele & Anwendungen**

- Kooperative Robotik (z.B. Drohnenschwärme)
- Strategische Spiele (z.B. StarCraft, Poker, ...) Puzzles
- Verkehrssteuerung (z.B. Ampeloptimierung mit mehreren Agenten?)
- Verteilte *Energiemanagementsysteme*

Multi-Agent RL

Bottom Up vs Top Down?

Mixture of Experts MoE (Börse / LLM)

verschiedene Agenten geben verschiedene "*Meinungen*" ab
am Ende wird eine gemeinsame Entscheidung getroffen,
Anhand von verschiedenen Kriterien.

https://github.com/Al4Finance-Foundation/FinRL-Trading/blob/master/rl_model.py

Beispiele & Anwendungen

- Kooperative **Robotik** (z.B. **Drohnenschwärme**)
 - **Industrie**/Reinigungsroboter (Kollisionsvermeidung, Kriegsdrohen)
 - **Robocup** Fußball WM (kleine *Hunde*)
- Strategische **Spiele** (z.B. **StarCraft**, Poker)
- Industrieroboter (Assemblieren Gabelstapler Transporter)
- **Verkehrssteuerung** (z.B. Ampeloptimierung mit mehreren Agenten?)
 - Lieferfahrzeuge Fahrer / Computer LIDAR 'verfügbar'
 - Flugtaxi Lilium
- Verteilte *Energiemanagementsysteme*
- **Leben** (Mensch und Tier und Pflanze und Maschine) **Soziale Insekten! Gruppenverhalten** (Ameisen, Bienen in Panik...)
 - Kolonie als Meta-Organism / 'Agent'
 - Pilze / Flechten / Korallen (Territorialkämpfe)
- **Börsen** (Trading bots stocks) Mixture of Experts
- CyberSecurity (Agents = Computers, Penetration Testing $\approx$ Friendly Hackers + External Evil Agents)
 - Anomaly / Thread-Detection-**Net** Mustererkennung

Beispiele & Anwendungen MARL

Synchron Turnen/Schwimmen/Fliegen/Tanzen

Dronen Flugshow (statt Böller!)

Selbstfahrende Autos Waymo Einparken (konkurierend statt kooperierend;)

Supervised / Fest einprogrammiert / RL

Medizinkapseln

Kapselendoskope(-) Gewebeproben entnehmen oder minimalinvasive Eingriffe

Sport

Boxroboter Wettbewerb Fußball Basketball ...

Perplexity? LLM Agenten MoE. MARL finetuning 'denkbar'

Baustelle ? Teuer MARL zu trainieren? Vorinvestition.

Humanoide? Wann? 2030 2040? Problem: Taktilität, Kraft, Preis

Erntehelfer: Sensibel genug?

künstliche Gliedmaßen optimieren (nicht MARL aber NN)

Reinigungsflotte wer war schon wo? (minimal MARL)

Industrie Lager

E-Gabelstapler

AMRs = **Autonomous Mobile Robots**

Aspekte von Multi-Agent Algorithmen

kooperativ / konkurrierend

kommunikativ / eigenständig

eigene / shared Policy?

eigene Hyperparameter?

eigenes Modell / Kopie von Umgebung?

(de)zentralisiert / autonom (top/bottom up)

(a)synchron

Kooperation

Pseudocode für **kooperatives Ensemble**:

for episode in episodes:

for agent in agents:

observe state s

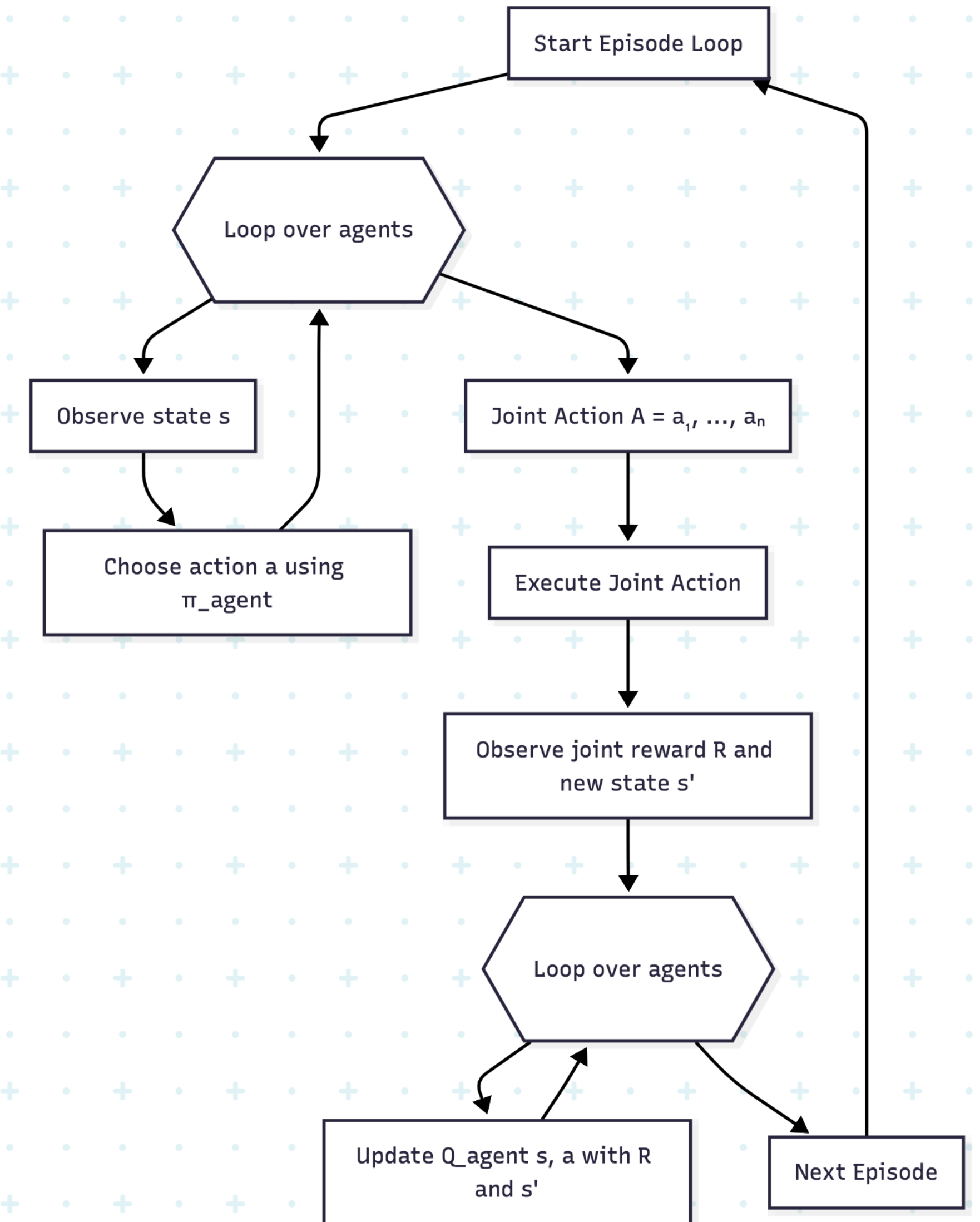
choose action a using **independent** policy π -agents

execute joint action $A = (a_1, \dots, a_n)$

observe **joint reward R** and new state s'

for agent in agents:

update $Q_{\text{agent}}(s, a)$ with R and s'



Advantage

a) TD Advantage "stepwise PG"

$$A(s,a) = \mathbf{Q}(s,a) - V(s)$$

b) Monte Carlo Advantage "episodisches PG"

$$A_t = \mathbf{G}_t - V(s_t) \text{ "baseline"}$$

$$\mathbf{G}_t = \sum_i \gamma^{t+i} r_i$$

Generalized Advantage Estimator (GAE)

$$\hat{A}_t^{(\lambda)} = \sum (\gamma\lambda)^l \delta_{t+l}$$

- Für $\lambda = 1$: volle Monte Carlo Advantage
- Für $\lambda = 0$: reiner TD-Fehler

PS: $\mathbb{E}_{a \sim \pi(s)} [A(s,a)] = 0$

GAE

Generalized Advantage Estimate

$$A_t := \sum (\gamma \lambda)^l \delta_{t+l}$$

δ_t ist unser delta von TD

$\delta_t = \underline{v}_t(S_t) - v_t(S_t)$ "Temporal Difference of Target v to real v"

$\delta_t = R_{t+1} + \gamma v_t(S_{t+1}) - v_t(S_t)$ "TD error of estimate"

$\lambda = 0 \rightarrow$ 1-step TD error $A_t = \delta_t$

$\lambda = 1 \rightarrow$ full GAE (infinite sum, Monte Carlo-like)

Asynchronous Kooperation

A3C Asynchronous Advantage Actor Critic

A3C Pseudocode für **kooperatives** hive-mind Ensemble:

for episode in episodes:

 get joint policy

 for agent in agents:

 observe state s

 choose action a using **own** policy π -agents

go execute joint action $A = (a_1, \dots, a_n)$

 observe **joint reward** R and new state s'

 for agent in agents:

update $Q_agent(s, a)$ with R and s'

Advantage A2C

A2C – Advantage Actor-Critic

Initialize policy $\pi(a \mid s)$, value function $V(s)$

for iteration = 1, 2, ... do:

Collect N trajectories of length T using π_θ

for each **trajectory** $\tau = (s_0, a_0, r_0, \dots, s_T)$:

for $t = 0$ to $T-1$:

$A_t = Q(s_t, a_t) - V(s_t)$, $Q(s_t, a_t) \approx r_t + \gamma * V(s_{t+1})$

$A_t = r_t + \gamma * V(s_{t+1}) - V(s_t)$ # Advantage estimate $\approx$ delta!

Loss_ π = $-E_t [\log \pi(a_t \mid s_t) * A_t]$ # policy loss $E_t \Rightarrow .mean()$!

Loss_ V = $E_t [(V(s_t) - (r_t + \gamma * V(s_{t+1})))^2]$ # critic/value MSE loss

H = $E_t [-\log \pi_\theta(a_t \mid s_t)]$ # Noise Compute entropy bonus (optional)

Update θ using $\nabla \theta$ (Loss_ π – $\beta * H$)

Update ϕ using $\nabla \phi$ Loss_ V

A3C – Asynchronous Advantage Actor-Critic. (Always Absolute Apsilon A8C)

A3C

Anpassen von **A2C – Asynchronous Actor-Critic** => **A3C**

Initialize global policy & value (parameters θ, ϕ)

for each **worker** thread do: # agents

Initialize **local** $\theta' \leftarrow \theta, \phi' \leftarrow \phi$

repeat:

Collect **trajectory** $\tau = (s_0, a_0, r_0, \dots, s_T)$ using $\pi_{\theta'}$

for $t = 0$ to $T-1$:

$$A_t = r_t + \gamma * V\phi'(s_{t+1}) - V\phi'(s_t)$$

$$\text{Loss}_{\pi'} = - E_t [\log \pi_{\theta'}(a_t | s_t) * A_t]$$

$$\text{Loss}_{V'} = E_t [(V_{\phi'}(s_t) - (r_t + \gamma * V_{\phi'}(s_{t+1})))^2]$$

$$H' = E_t [-\log \pi_{\theta'}(a_t | s_t)]$$

$$g_{\theta'} \leftarrow \nabla \theta' (\text{Loss}_{\pi'} - \beta * H')$$

$$g_{\phi'} \leftarrow \nabla \phi' \text{Loss}_{V'}$$

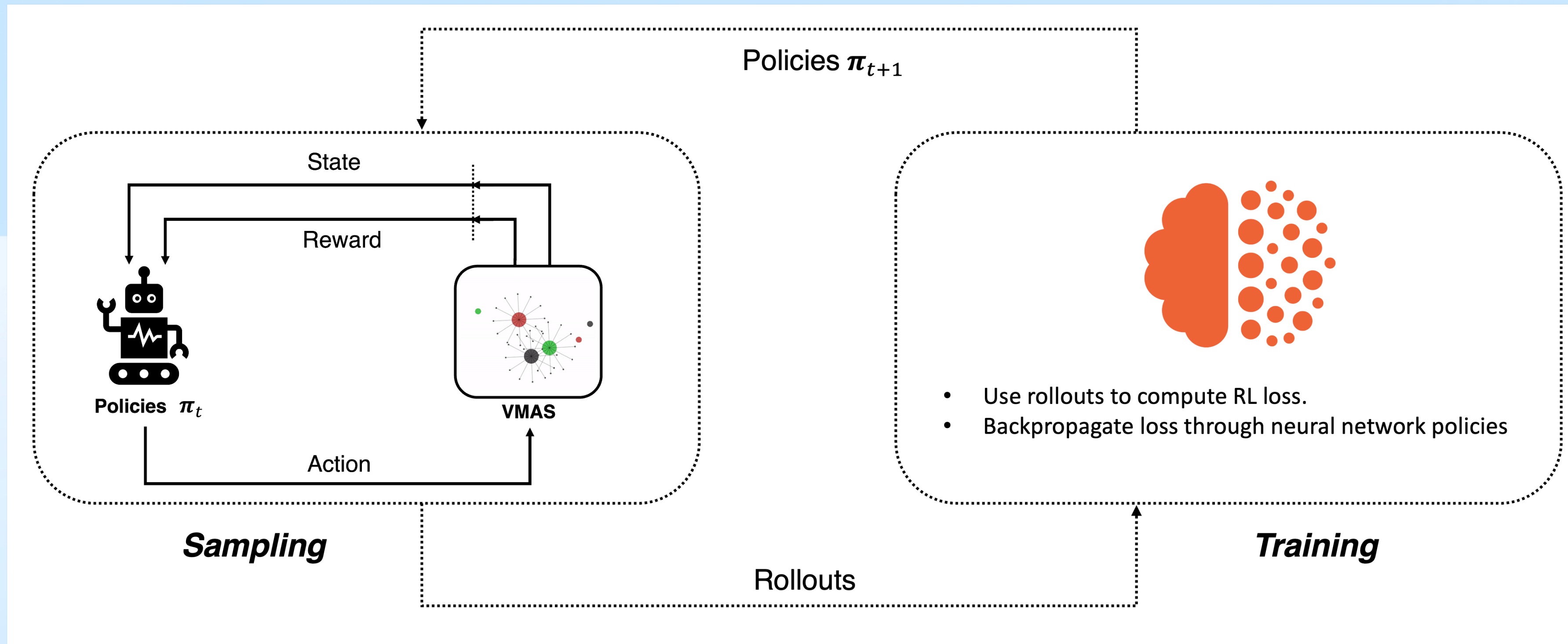
Asynchronously apply $g_{\theta'}, g_{\phi'}$ to global θ, ϕ

Sync local $\theta' \leftarrow \theta, \phi' \leftarrow \phi$

Aufgaben

Multi Agent **Library** / Example

Paper: Mastering diverse control tasks through world models



Libraries

<https://github.com/Replicable-MARL/MARLlib> **2023** :(

https://docs.pytorch.org/rl/stable/tutorials/multiagent_ppo.html **TorchRL**

<https://docs.ray.io/en/latest/> **Ray / RLLib**

<https://tianshou.org/en/stable/>

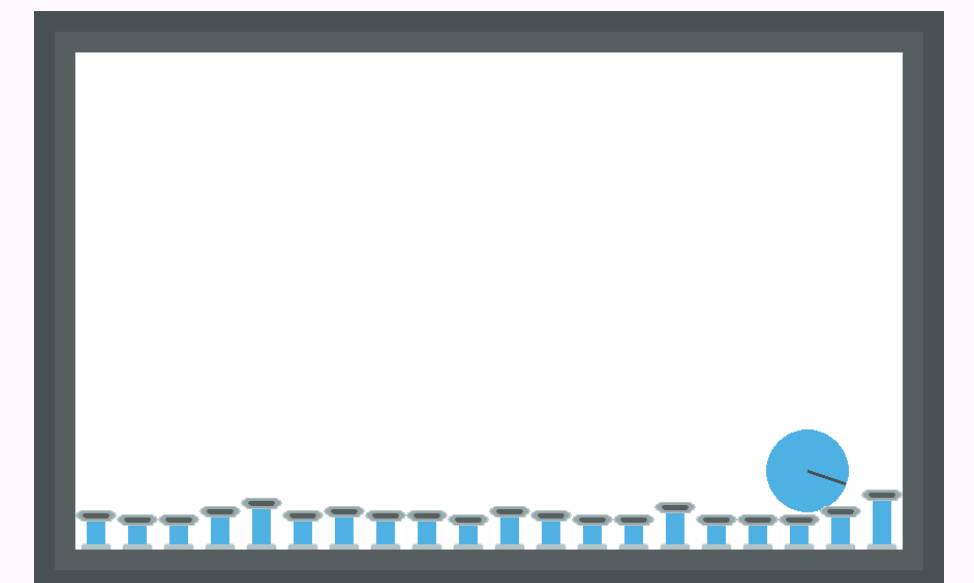
<https://github.com/Farama-Foundation/PettingZoo>

```
from pettingzoo.butterfly import pistonball_v6

env = pistonball_v6.parallel_env(render_mode="human")
observations, infos = env.reset()

while env.agents:
    # this is where you would insert your policy
    actions = {agent: env.action_space(agent).sample() for agent in env.agents}

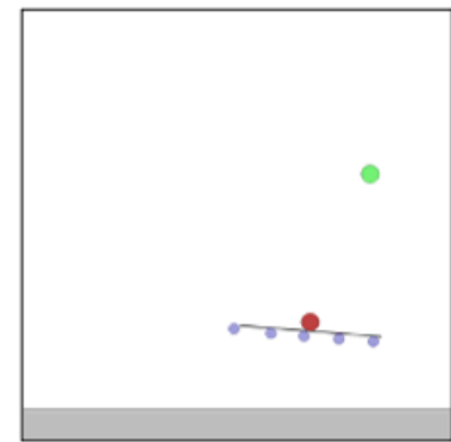
    observations, rewards, terminations, truncations, infos = env.step(actions)
env.close()
```



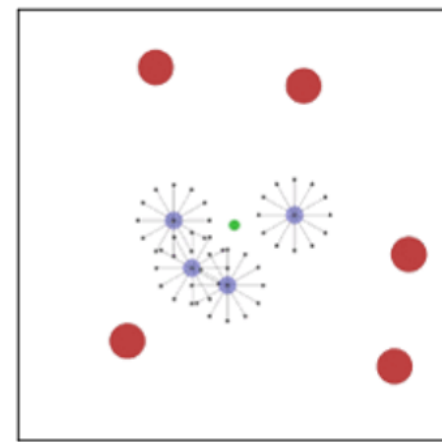
TorchRL 'gym'

https://docs.pytorch.org/rl/stable/tutorials/multiagent_ppo.html

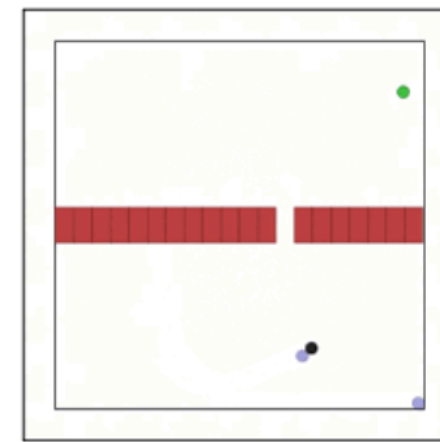
Docs > Multi-Agent Reinforcement Learning (PPO) with TorchRL Tutorial



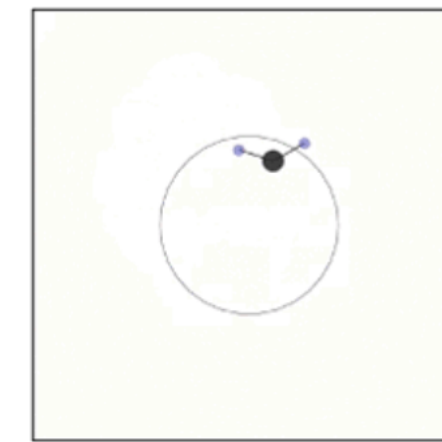
Balance



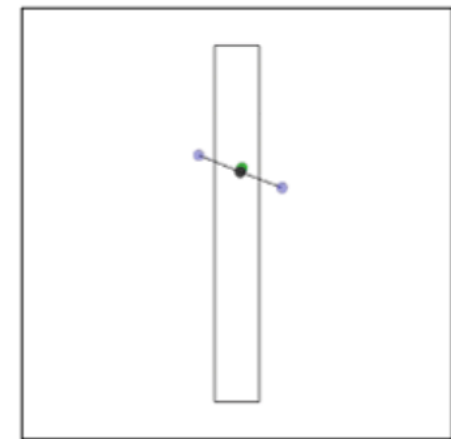
Flocking



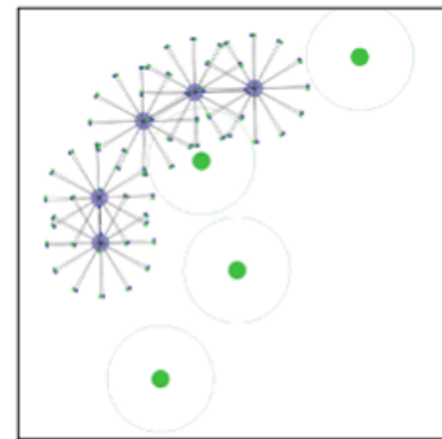
Ball passage



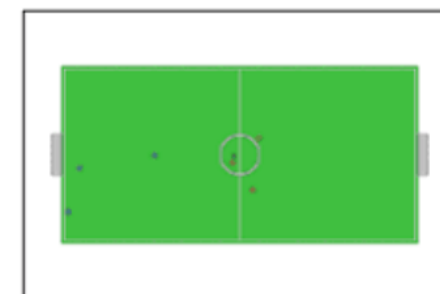
Ball trajectory



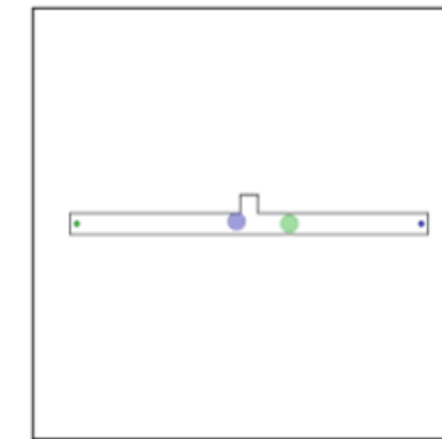
Buzz wire



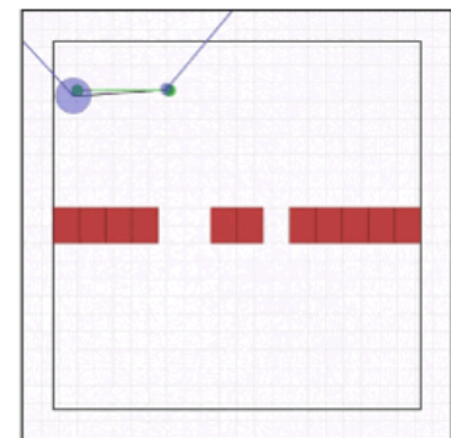
Discovery



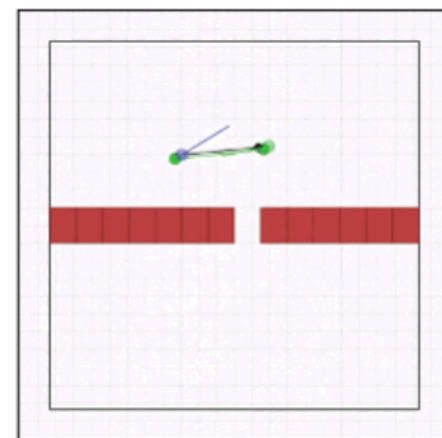
Football



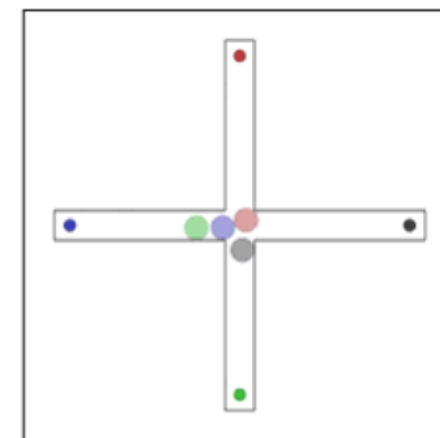
Give way



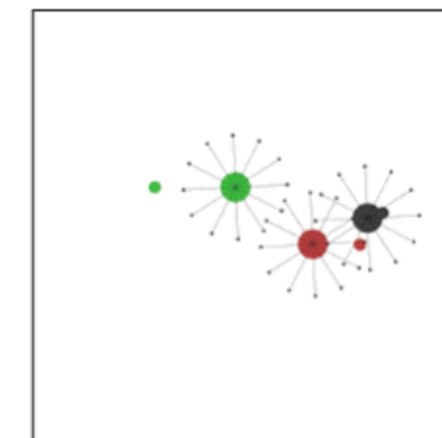
Joint passage size



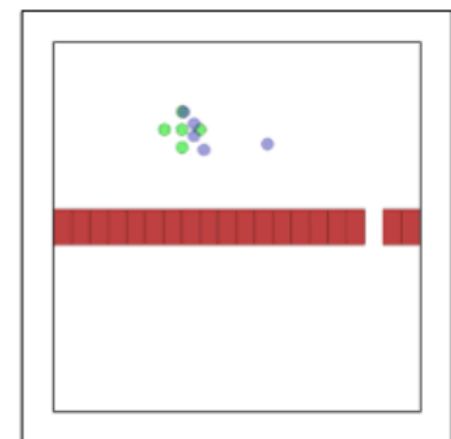
Joint passage



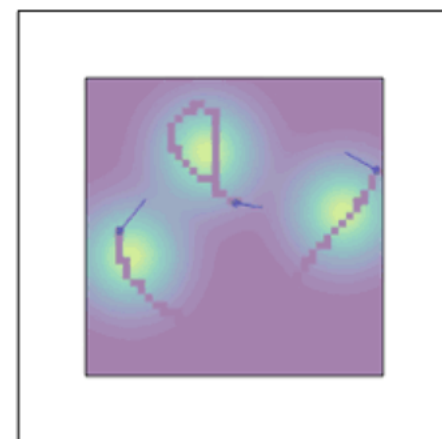
Multi give way



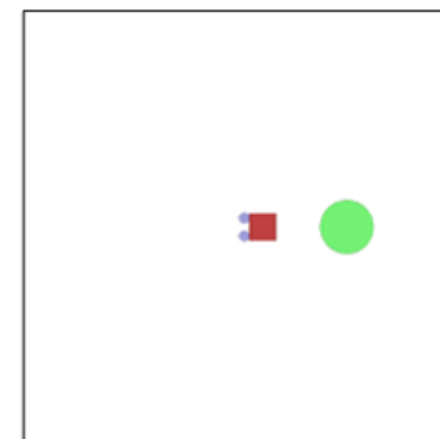
Navigation



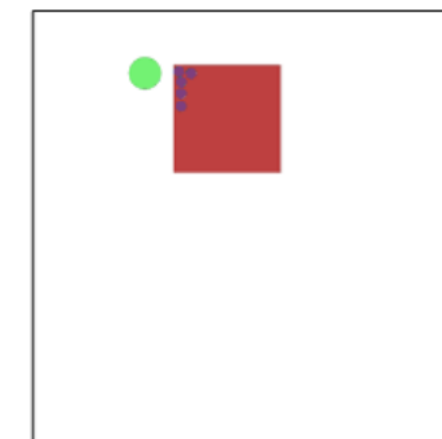
Passage



Sampling



Transport



Reverse transport

Aufgabe

Welche Aspekte von Multi-Agent Algorithmen sind in *multiagent_ppo.py* umgesetzt?

a / b

kooperativ / konkurrierend ? a (*emergent!*) / b (ignorierend bei give way)

kommunikativ / eigenständig? Kommunikation über collector? wissen wir nicht. eher nicht / indirekt b

eigene / shared Policy? **shared** (wenn eingeschaltet!)

eigene Hyperparameter? **nein**

je eigenes Modell / Kopie von Umgebung? **nein**

autonom / zentralisiert? autonom (weil centralised=False, controller nur zum Organisieren)

(a)synchron? **synchrone Schleife**

Aufgabe

Welche Aspekte von Multi-Agent Algorithmen sind in *multiagent_ppo.py* umgesetzt?

a / b

kooperativ / konkurrierend? a

kommunikativ / eigenständig? b

eigene / shared Policy? b (einstellbar!)

eigene Hyperparameter? ja/nein

eigene Umgebung? eigenes Modell / Kopie von Umgebung? ja/**nein**

autonom / zentralisiert? a/b?

(**a**)synchron? s!

PPO

Proximal Policy Optimization

Vielleicht der wichtigste **RL Regularisierungsansatz** der Dekade.

Stelle sicher dass die updated **policy** *nahe* (***proximal***) zur alten policy ist, also ohne katastrophale Sprünge: Kontinuität beim lernen.

Bei RL ist der "Pfad zum Erfolg" oft ein **schmaler Grat**.
Zu kleine Schritte führen zum **fast Stillstand**,
Zu große Schritte zu **katastrophalen Fehlschritten**.



PPO

Proximal Policy Optimization

proxima ist lateinisch für "nahe"

proximal kennen wir von ...

approximiert "angenähert"

Proxima Centauri 3-Sterne 4.25 light-years 'nahe' 3 body problem!

vibe proofing. coding

Terence Tao best mathematician

PPO

Proximal Policy Optimization

Wie stellen wir sicher, dass die updated **policy** nahe (*proximal*) zur alten policy ist?

$\pi \approx \pi'$ $\Leftrightarrow \pi - \pi' \approx 0$ oder besser ...

PPO

Proximal Policy Optimization

Wie stellen wir sicher, dass die updated **policy** nahe (***proximal***) zur alten policy ist?

$$\pi \approx \pi' \Leftrightarrow \pi / \pi' \approx 1$$

this quotient **ratio** is often called

$$r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}$$

PPO

Proximal Policy Optimization

Wie stellen wir sicher, dass die updated **policy** nahe (*proximal*) zur alten policy ist?

$$\pi \approx \pi' \Leftrightarrow \pi / \pi' \approx 1$$

this quotient **ratio** is often called

$$r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}$$

Sota : **GRPO** group relative policy optimization, **DeepSeek** LLM

PPO

Proximal Policy Optimization

Wie stellen wir sicher, dass die updated **policy** nahe (*proximal*) zur alten policy ist?

$$\pi \approx \pi' \Leftrightarrow \pi / \pi' \approx 1$$

this quotient **ratio** is often called

$$r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}$$

Naiv: **loss += MSE(π / π' , 1)**

PPO

Proximal Policy Optimization

updated **policy** nahe (*proximal*) zur alten policy ist?

Wir stellen sicher dass die **ratio nahe bei 1** ist:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Via **clipping**, oder andere **losses**

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta))\hat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t)]$$

Trusted Regions: erlaube nur r im Bereich **[1- ε , 1+ ε]**

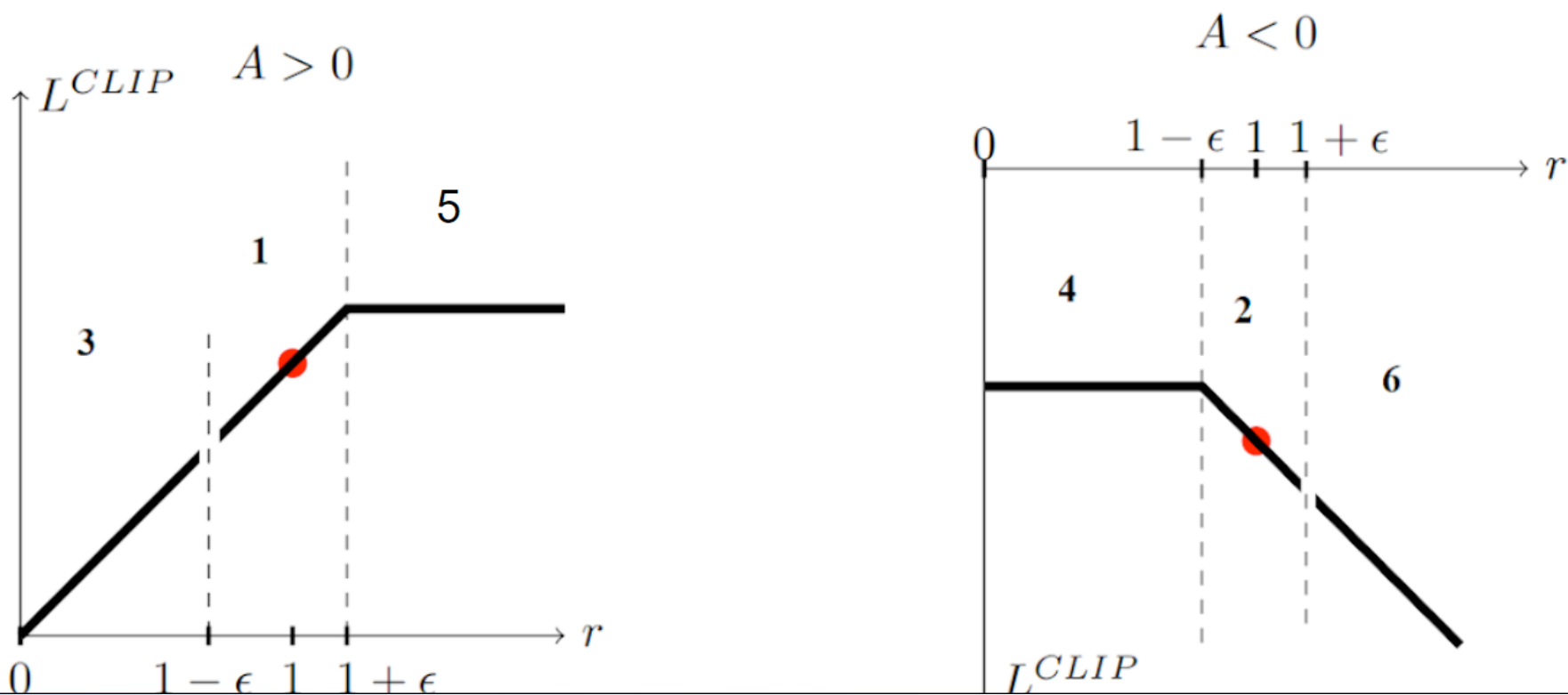
PPO

normale Formel $L = -\sum \log(\pi) * A$ wird ge-clip'ed und mit r ersetzt:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right]$$

<https://huggingface.co/blog/deep-rl-ppo#visualize-the-clipped-surrogate-objective>

	$p_t(\theta) > 0$	A_t	Return Value of $\min$	Objective is Clipped	Sign of Objective	Gradient
1	$p_t(\theta) \in [1 - \epsilon, 1 + \epsilon]$	+	$p_t(\theta) A_t$	no	+	✓
2	$p_t(\theta) \in [1 - \epsilon, 1 + \epsilon]$	-	$p_t(\theta) A_t$	no	-	✓
3	$p_t(\theta) < 1 - \epsilon$	+	$p_t(\theta) A_t$	no	+	✓
4	$p_t(\theta) < 1 - \epsilon$	-	$(1 - \epsilon) A_t$	yes	-	0
5	$p_t(\theta) > 1 + \epsilon$	+	$(1 + \epsilon) A_t$	yes	+	0
6	$p_t(\theta) > 1 + \epsilon$	-	$p_t(\theta) A_t$	no	-	✓



PPO

Proximal Policy Optimization

updated **policy** nahe (*proximal*) zur alten policy ist?

Wir stellen sicher dass die **ratio nahe bei 1** ist:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Via **clipping**, oder andere **losses**:

custom_loss = (1-r)^2 *punish* $r \gg 1$

loss += custom_loss **prinzipiell** einfach **bestrafen**

schlechtes toy example, geht nur um das Prinzip in Wirklichkeit **KL**

Vorschau: Information und Entropie

Shannon-Information $:= -\log(p)$ **self Information** ($\neq$ Fischer Information)

$p=1 \Rightarrow -\log(1) = 0$ keine 'Information' wenn ein sicheres Ereignis eintritt

$p \approx 0 \Rightarrow -\log(0) \approx \infty$ maximale 'Information' wenn ein seltenes Ereignis doch eintritt

"punktweise Überraschung"

Entropie:

Entropie (mittlere Shannon-Information):

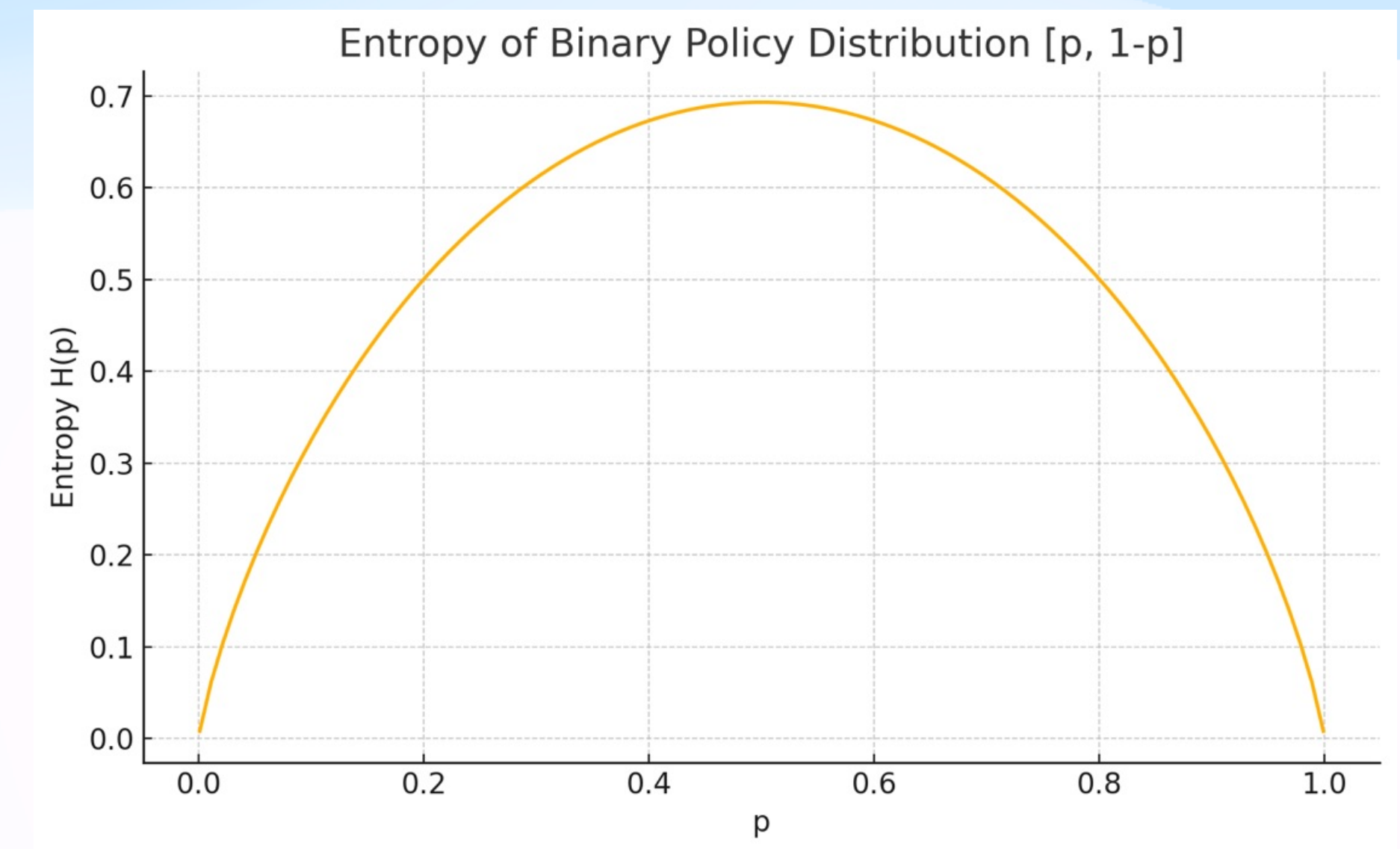
durchschnittliche Überraschung / Unordnung (einer Verteilung)

$$H(p) = -\sum p_i \cdot \log(p_i) \quad \sum p_i = 1$$

Bei Unfairer Münze die immer Kopf/Zahl ergibt, gibt es keine Überraschungen $\Rightarrow H=0$

Bei Fairer Münze die mal Kopf/Zahl ergibt, gibt ist Ergebnis nicht vorhersehbar "Überraschungsfaktor $H=.7$ "

Wird verwendet z.B. in **SAC, PPO, MaxEntropy IRL**



KL Divergenz

KL Kullback-Leibler-Divergenz misst wie **nah** unsere neue policy(?) an der alten ist (so wie **PPO** mit $\log$, mit clipping $1 \approx 1.44$ bits?)

Je kleiner die KL-Divergenz, desto ähnlicher sind die Policies:

$$DKL = \sum \pi(a) \cdot \log(\pi(a) / \pi'(a))$$

$$D_{KL}(\pi_{\text{alt}} \parallel \pi_{\text{neu}}) = \sum_a \pi_{\text{alt}}(a \mid s) \log \left(\frac{\pi_{\text{alt}}(a \mid s)}{\pi_{\text{neu}}(a \mid s)} \right)$$

This disables them while they are already minimized well to focus learning on the prediction loss

Sum is always **positive!**

Some terms **might be negative** but since π is probability $\pi(i)$ in $[0,1]$ and $\sum \pi(i) = 1 \Rightarrow$ cancel out (Gibbs inequality)

Fantastic KL Divergence and How to (Actually) Compute It

<https://youtu.be/tXE23653JrU>

KL Divergenz

KL Kullback-Leibler-Divergenz

Ist eng mit dem Konzept von **Cross Entropie** verwandt:

Die **Cross-Entropie** zwischen P und Q ist definiert als:

$$H(P,Q) = - \sum P(x) \log(Q(x))$$

Die Self-**Entropie** von P ist definiert als:

$$H(P) = - \sum P(x) \log(P(x))$$

Zeige: **KL Divergenz**:

$$D_{kl} = H(P,Q) - H(P)$$

KL = Cross-Entropie - Entropie

$$D_{kl} = - \sum P(x) \log(Q(x)) - - \sum P(x) \log(P(x)) = \sum P(x) (\log(P(x)) - \log(Q(x))) = \sum P(x) (\log(P(x)/Q(x)))$$

$$D_{kl} = \sum \pi(a) \cdot \log(\pi(a)/\pi'(a)) = \dots \quad \text{💡 Nutze } \log(a/b) = \log(a) - \log(b)! \quad (Q=\text{neue policy} \quad P=\text{alte})$$

$$D_{KL}(\pi_{\text{alt}} \parallel \pi_{\text{neu}}) = \sum_a \pi_{\text{alt}}(a \mid s) \log \left(\frac{\pi_{\text{alt}}(a \mid s)}{\pi_{\text{neu}}(a \mid s)} \right)$$

Für One Hot Encoded Target fallen die beiden Konzepte tatsächlich zusammen weil die Entropie von One Hot and Coding null ist!

KL Divergenz

KL Kullback-Leibler-Divergenz

Zum Messen der 'Nähe' von policies.

Anwendungen:

Maschinelles Lernen / Deep Learning:

- Verlustfunktionen in Klassifikationsaufgaben (z. B. Softmax + cross-entropy ist KL).
- Variational Inference: Approximation von Posteriors (z. B. in Variational Autoencoders).
- Generative Modelle (GAN-Varianten, Diffusion Models).

• Statistik:

- Modellselektion (AIC, BIC basieren auf KL).
- Hypothesentests und Likelihood-Ratios.

• Signalverarbeitung & Informationstheorie:

- Quantifizierung von Redundanz, Kanalloptimierung.
- Sprach- und Bildverarbeitung (z. B. Speaker Verification, Texture Analysis).

• Naturwissenschaften:

- Bioinformatik (Vergleich von Sequenzmotiven, Populationsverteilungen).
- Neurowissenschaften (Vergleich von Feuerratenverteilungen).

• Ökonomie / Sozialwissenschaften:

- Messen von Ungleichheiten in Wahrscheinlichkeitsverteilungen (z. B. Marktanteile).

KL Divergenz

KL Kullback-Leibler-Divergenz

Anwendung:

Extra Loss term kann einfach mit eingebaut werden!

$$D_{\text{KL}}(\pi_{\text{alt}} \parallel \pi_{\text{neu}}) = \sum_a \pi_{\text{alt}}(a \mid s) \log \left(\frac{\pi_{\text{alt}}(a \mid s)}{\pi_{\text{neu}}(a \mid s)} \right)$$

KL-Divergenz ein mathematischer Anker für **Stabilität, Exploration und Approximation** in praktisch allen state-of-the-art RL-Algorithmen.

"Erwartungen / Funktionen"

Sind unseren Größen Q , V , G stets einheitlich definiert?

Alle Werte/Funktionen/Formeln gibt es in drei Formen:

- empirisch (konkret)
- geschätzt (modellbasiert)
- theoretisch (Erwartungswert unter Verteilung)

G kann ein **konkreter Wert** sein, **berechnet** aus **gesampelten** r

G kann ein **gemittelter Wert** sein, **berechnet** aus **batch**

G kann ein **erwarteter Wert** sein, **geschätzt** durch ein **Q-Netzwerk**

G kann ein **Erwarteter Wert** $\mathbb{E}$ sein, im Sinne im Kontext einer **Verteilung θ unseres Netzwerks**

G kann ein **Erwarteter Wert** $\mathbb{E}$ sein, im Sinne im Kontext einer theoretischen **Verteilung p unserer Umgebung**

$$G_t = \sum_{i=0, i < T-t} \gamma^i r_{t+i}$$

$$G_t = \mathbb{E} \sum_{i=0, i < T-t} \gamma^i (r_{t+i} \mid \pi)$$

Q intuitiv als Qualität einer Aktion

$Q = \mathbb{E}(G_t)$ abstrakte / theoretische Definition von Quality vs MDP

$q(s, a) = \sum p^* [r + \gamma \sum_{a'} \pi(a' \mid s') q^p(s', a')]$ exakte **Schätzung** mit Bellman Gleichung (im limit konvergiert Schätzung gegen "wahren Wert")

$Q \leftarrow Q + \alpha (r + \gamma Q(s', a') - Q)$ pragmatischer **Schätz-Ansatz** mit **updates** z.B. bootstrapping, **DQN**, ...

Nachtrag: *Return*

G Gain expected cumulative reward $\approx$ Return **R** function **synonym**? JA! Oder Nicht ganz!?

⚠ Vermeide *Return* **R** wegen Verwechslungsgefahr mit *reward* **r**!

$G = \sum r$ "Gewinn ist Summe der rewards"

$G = \sum \gamma^i * r$ G_t Expected return starting from time t

$V = E(G \mid s)$ local!

$Q = E(G \mid s, a)$

$\hat{R} = J(\pi) := E(s \sim \mu, a \sim \pi)[Q]$ 💡 overall global expected return

$\hat{R} = \sum \mu(s) \sum_a \pi^* Q(s, a)$ # μ state distribution 💡

$\mu(s) = \sum_t \gamma^t * p(s_t \mid \pi)$ # the expected frequency of visiting state **s** under policy 💡

eigentlich $E[R]$ so **Q** so bellman?

Custom Loss

Loss = ... beliebig setzbar ...

Loss = **mathematisch motiviert** :

Loss = classical Policy Gradient :

Loss = $-\log_prob * \Phi$ 'surrogate'

letzteres hat immer selben Erwartungswert! nur andere Varianz bei anderem Φ $\approx$ Streuung

Sei

$$g(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a | s) G], \quad L_\Phi(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a | s) [G - \Phi(s)]].$$

Da $\sum_a \pi_\theta(a | s) = 1$ gilt

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a | s)] = \nabla_\theta \sum_a \pi_\theta(a | s) = \nabla_\theta 1 = 0,$$

folgt unmittelbar

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a | s) \Phi(s)] = 0 \quad \implies \quad L_\Phi(\theta) = g(\theta).$$

- Erwartungswert: unverändert (unbiased), unabhängig von der Wahl von Φ , solange Φ nicht von a abhängt.

- Varianz: $\text{Var}[\nabla_\theta \log \pi_\theta(a | s) (G - \Phi(s))]$ hängt stark von Φ .

Die optimale (minimal-Varianz)-Wahl ist

$$\Phi^*(s) = \mathbb{E}_{\pi_\theta}[G | s] = V^{\pi_\theta}(s),$$

RL bei LLM

a) Finetuning

b) Environment python evaluation reward = 1 bei richtiger Lösung

Man nimmt Foundation Model (GPT-3)

Gib 100000 Mathe Aufgaben und löse sie (geleitet von PPO/GRPO)

<think>

mach was du willst

<think>

<solution>

wert

</solution>

Referate / Extra Wunsch Themen

Inverse Reinforcement
Meta Reinforcement Learning

Novelty Exploration **Surprise**

Hierarchisches Reinforcement Learning (arc!)

RL + **LLM** (DeepSeek, ...)

Dreamer

Alpha**Zero**, μ **Zero**? AbsoluteZero

Praxis: **Eigene Umgebungen**

FactoryIO.com / NVIDIA / Siemens **Digital Twin**

bullet.py robot arm

Referate : Extra Wunsch Themen

Wiederholung

RL in Sprachmodellen **LLM** (RLHF / **reasoning** / pokemon)

Aufmerksamkeitsmodelle

priorisierte Wiederholung

RL in Spielen?

Algorithmen: **SAC, TD3**

virtual twin / **factory.io** / **Kuka?** Enes

Sparmodell? **BudgetEnv?** Custom Environment.
mit baseline evaluieren?

reward shaping?

<https://gibberblot.github.io/rl-notes/single-agent/reward-shaping.html>

<https://arxiv.org/abs/2011.02669>

einfachster State of the Art Algorithmus **SoTa** $\pm$?

most **impressive example** a) running locally b) trainable locally c) tuneable locally

RL in Spielen welche Spiele wurden (noch nicht) gelöst.

Vokabular

PPO

MARL

lambda in GAE

Emergent

PPO

KL

(Cross/Self)Entropy

centralized