Neural networks for Natural Language Processing

Jun.-Prof. Dr. Sophie Fellenz

Week 09 – Reinforcement Learning in NLP

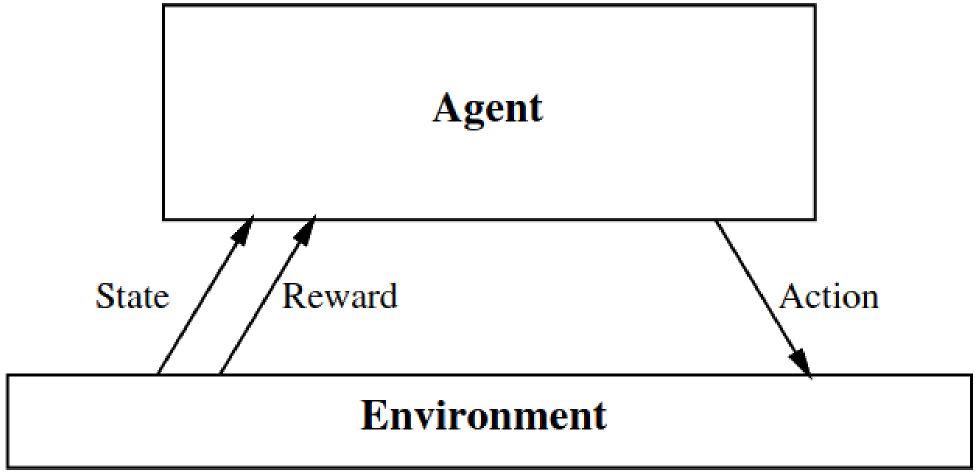
Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Books on RL

New book by Kevin Murphy: Reinforcement Learning: An Overview (Released on December 9th 2024 on arxiv)

Classic Intro by Sutton and Barto: Reinforcement Learning: An Introduction (1998)

Recap: Reinforcement Learning Problem



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{, where } 0 \leq \gamma < 1$$

Introduction to RL

‘Reinforcement learning is learning *what to do- how to map* situations to actions - so as to *maximize* a numerical *reward* signal.’

[Sutton&Barton 2018]

Introduction to RL

‘Reinforcement learning is learning *what to do- how to map* situations to actions - so as to *maximize* a numerical *reward* signal.’

[Sutton&Barton 2018]

What does this mean with respect to language?

-> language is now viewed as an act that is performed to influence the environment (John Searle, Speech Act Theory)

Timeline for RL in Games

- 1992: TD-Gammon, temporal difference learning, Backgammon
- 1997: Deep Blue defeats Kasparov in Chess
- 2013: Atari, deep RL
- 2016: AlphaGo, deep RL and MC-tree search defeats world champion in Go
- 2019: Dota 2 world champion defeated by OpenAI Five

RPTU

Agentic Workflows

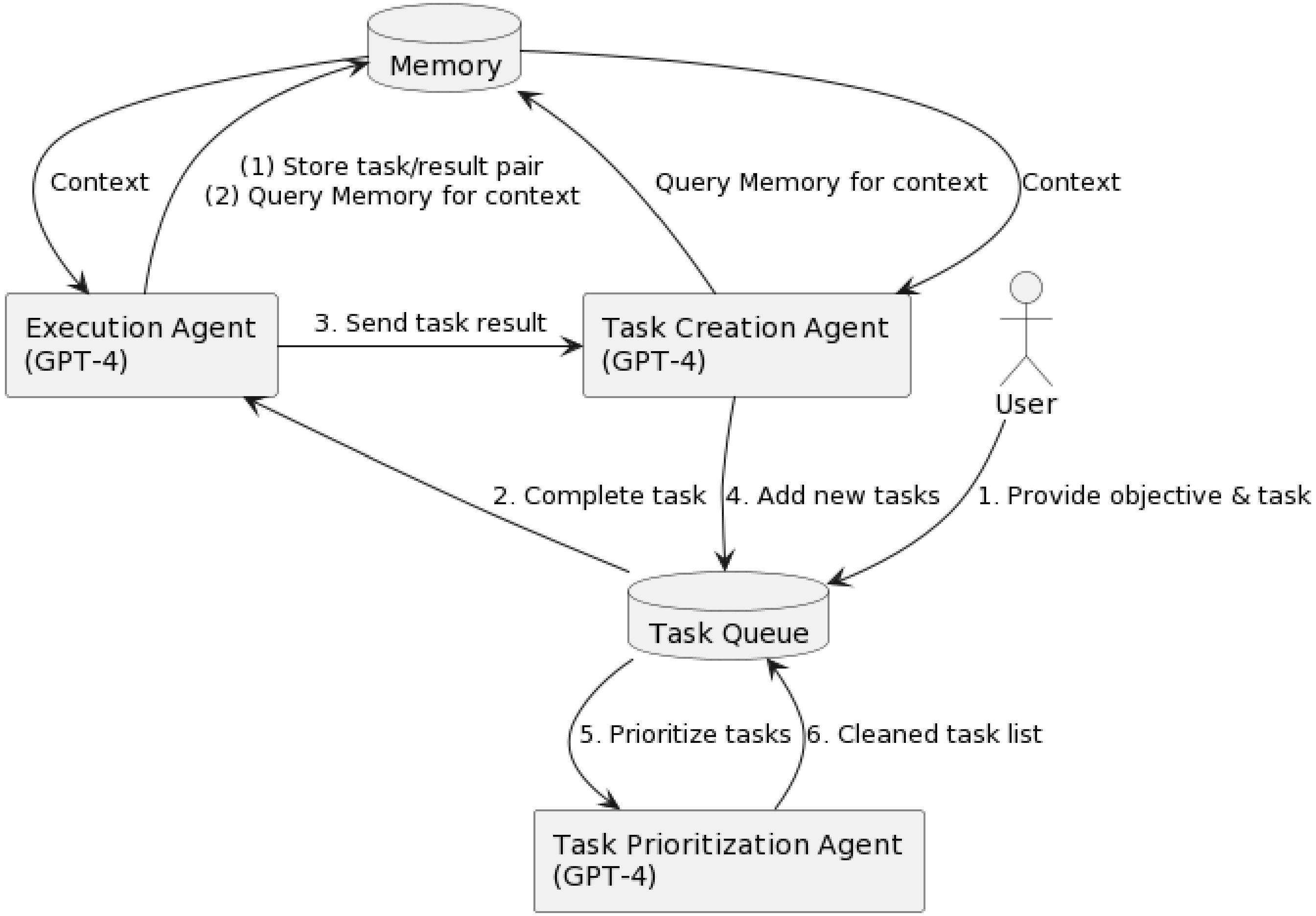
AI agentic workflows are structured processes that involve AI agents that operate autonomously to achieve specific goals within a defined environment.

Key features:

- Multiple agents
- Collaborative task execution
- Dividing tasks into steps that are executed individually
- Results and data are shared in a structured way

BabyAGI – AutoGPT (28.03.2023)

Image created by DALLE



https://yoheinakajima.com/task-driven-autonomous-agent-utilizing-gpt-4-pinecone-and-langchain-for-diverse-applications/

RPTU

Key Componentes of Agentic Workflows

- **Perception**: gather information about the environment
- **Decision-making**: decide on the next action based on pre-programmed goals and learned policies.
- **Action**: take action within the environment and monitor the effect of the action
- **Feedback and learning**: AI agents learn from their actions. Based on feedback and techniques such as RL, the agent adapts and improves over time
- **Collaboration**: coordinating tasks while communicating sharing information

Example Agentic Workflow

Conversational Agent:

- Perceive user‘s request (voice or text input)
- Make decision based on the intent of the query
- Take action (provide information or execute command)
- Learn from the interaction to improve future responses

Benefits of Agentic Workflows

Autonomy: no need for constant human input

Scalability: manage many tasks simultaneously, allows for systems that operate on large scalse

Adaptability: Can adapt to changes in dynamic environments (such as changes in inventory)

Outline

Today:

1. Introduction to Reinforcement Learning (RL)

2. Policy-Based RL

- **Policy Gradient methods, PPO**
- **ChatGPT**

3**. Value-based RL**

- Deep-Q Learning
- NLP applications

RPTU

Markov Decision Processes

Assume

- finite set of states S
- set of actions A
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
  - i.e., r_t and s_{t+1} depend only on *current* state and action
  - functions δ and r may be nondeterministic
  - functions δ and r not necessarily known to agent

Agent's Learning Task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

$$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

  from any starting state in S

- here $0 \leq \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \rightarrow A$
- but we have no training examples of form $\langle s, a \rangle$
- training examples are of form $\langle \langle s, a \rangle, r \rangle$

Value Function

To begin, consider deterministic worlds...

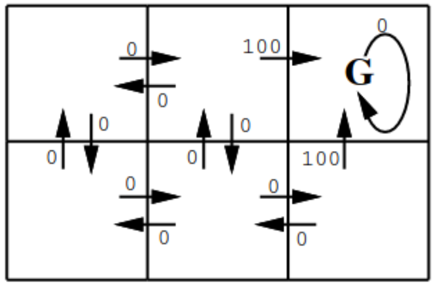
For each possible policy π the agent might adopt, we can define an evaluation function over states

$$
\begin{aligned}
V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \\
&\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}
\end{aligned}
$$

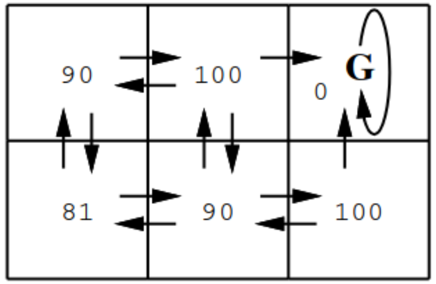
where $r_t, r_{t+1}, \ldots$ are generated by following policy π starting at state s

Restated, the task is to learn the optimal policy π^*

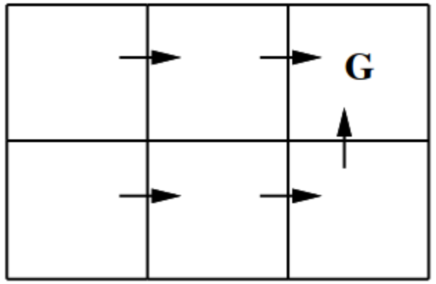
$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$



$r(s,a)$ (immediate reward) values



$V^*(s)$ values



One optimal policy

$\gamma = 0.9$

What to Learn

We might try to have agent learn the evaluation function V^{π^*} (which we write as V^*)

It could then do a lookahead search to choose best action from any state s because

$$\pi^*(s) = \operatorname*{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \Re$
- But when it doesn't, it can't choose actions this way

Q Function

Define new function very similar to V^*

$$Q(s, a) \equiv r(s, a) + \gamma V^*(\delta(s, a))$$

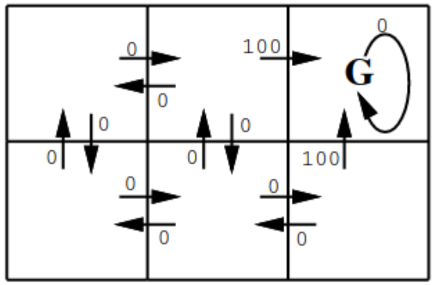
If agent learns Q, it can choose optimal action even without knowing δ!

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s, a) + \gamma V^*(\delta(s, a))]$$

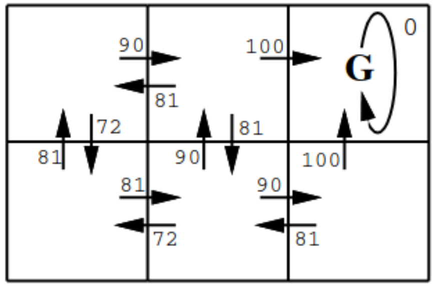
$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Q is the evaluation function the agent will learn

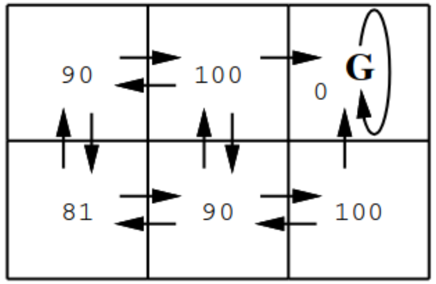
RPTU



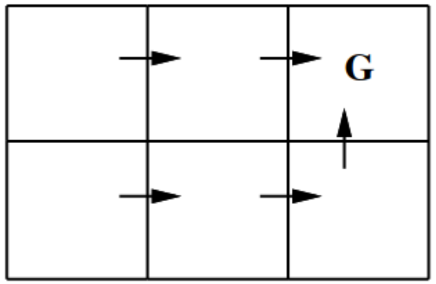
$r(s, a)$ (immediate reward) values



$Q(s, a)$ values



$V^*(s)$ values



One optimal policy

$\gamma = 0.9$

Training Rule to Learn Q

Note Q and V^* closely related:

$$V^*(s) = \max_{a'} Q(s, a')$$

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Nice! Let $\hat{Q}$ denote learner's current approximation to Q. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

where s' is the state resulting from applying action a in state s

RPTU

Q-Learning for Deterministic Worlds

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state s

Do forever:

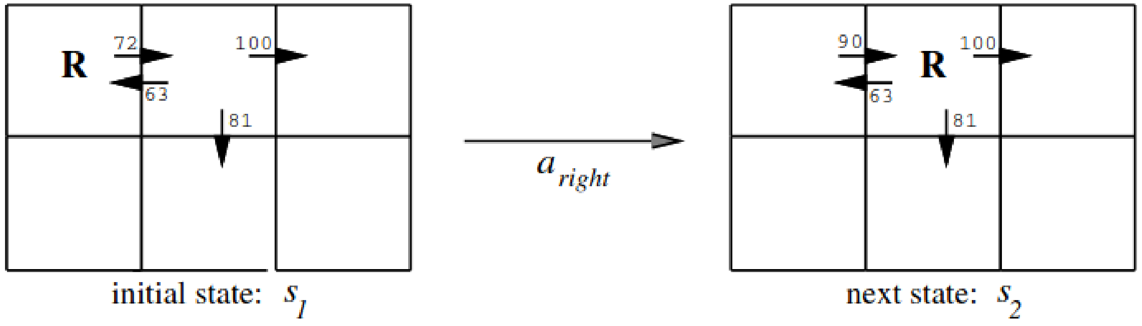
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

RPTU

Updating $\hat{Q}$



$$
\begin{aligned}
\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\
&\leftarrow 0 + 0.9 \max\{63, 81, 100\} \\
&\leftarrow 90
\end{aligned}
$$

notice if rewards non-negative, then

$$(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

RPTU

$\hat{Q}$ converges to Q. Consider case of deterministic world where see each $\langle s, a\rangle$ visited infinitely often.

Proof: Define a full interval to be an interval during which each $\langle s, a\rangle$ is visited. During each full interval the largest error in $\hat{Q}$ table is reduced by factor of γ

Let $\hat{Q}_n$ be table after n updates, and Δ_n be the maximum error in $\hat{Q}_n$; that is

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s,a) - Q(s,a)|$$

For any table entry $\hat{Q}_n(s,a)$ updated on iteration $n+1$, the error in the revised estimate $\hat{Q}_{n+1}(s,a)$ is

$$\begin{aligned} |\hat{Q}_{n+1}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s',a')) \\ &\quad -(r + \gamma \max_{a'} Q(s',a'))| \\ &= \gamma |\max_{a'} \hat{Q}_n(s',a') - \max_{a'} Q(s',a')| \\ &\leq \gamma \max_{a'} |\hat{Q}_n(s',a') - Q(s',a')| \\ &\leq \gamma \max_{s'',a'} |\hat{Q}_n(s'',a') - Q(s'',a')| \\ |\hat{Q}_{n+1}(s,a) - Q(s,a)| &\leq \gamma \Delta_n \end{aligned}$$

Note we used general fact that

$$|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$$

RPTU

Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values

$$\begin{aligned} V^{\pi}(s) &\equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] \\ &\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}] \end{aligned}$$

$$Q(s,a) \equiv E[r(s,a) + \gamma V^*(\delta(s,a))]$$

RPTU

Nondeterministic Case

Q learning generalizes to nondeterministic worlds

Alter training rule to

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a'} \hat{Q}_{n-1}(s',a')]$$

where

$$\alpha_n = \frac{1}{1 + visits_n(s,a)}$$

Can still prove convergence of $\hat{Q}$ to Q [Watkins and Dayan, 1992]

RPTU

Temporal Difference Learning

Q learning: reduce discrepancy between successive
Q estimates

One step time difference:

$$Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

Why not two steps?

$$Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a \hat{Q}(s_{t+2}, a)$$

Or n?

$$Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \cdots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

Blend all of these:

$$Q^{\lambda}(s_t, a_t) \equiv (1-\lambda) \left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \ldots \right]$$

RPTU

Temporal Difference Learning

$$Q^{\lambda}(s_t, a_t) \equiv (1-\lambda) \Big[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t)$$

Equivalent expression:

$$\begin{aligned} Q^{\lambda}(s_t, a_t) \; &= r_t + \gamma[\; (1-\lambda) \max_a \hat{Q}(s_t, a_t) \\ &\qquad + \lambda \; Q^{\lambda}(s_{t+1}, a_{t+1})] \end{aligned}$$

TD(λ) algorithm uses above training rule

- Sometimes converges faster than Q learning
- converges for learning V^* for any $0 \leq \lambda \leq 1$ (Dayan, 1992)
- Tesauro's TD-Gammon uses this algorithm

Deep Reinforcement Learning

Deep RL = Deep learning + Reinforcement learning

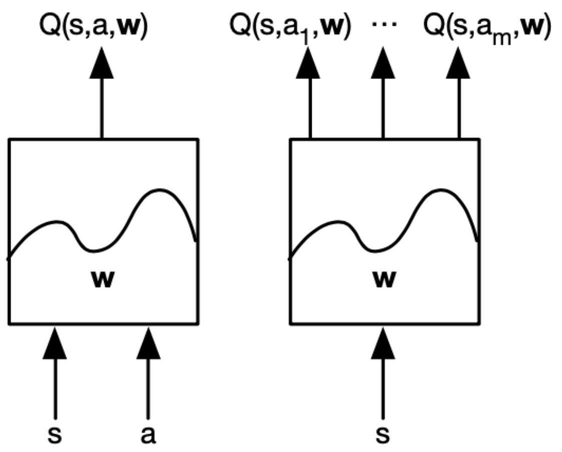
Use the *deep neural network* to *approximate*:

- Policy
- Value Function
- Model

Deep Q-networks

Represent Q-value function by Q-network with weights w

$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$



source:https://icml.cc/2016/tutorials/deep_rl_tutorial.pdf

Deep Q-Networks with Experience Replay

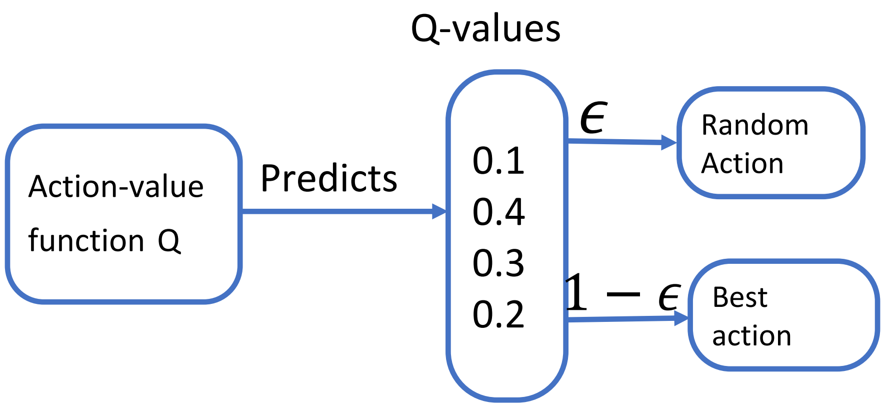
- An action-value function NN with parameters w
- A target action value function NN with parameters w^{-}
- (Epsilon-greedy) policy π selects action a_t
- A replay buffer to store transitions:$(s_t, r_{t+1}, a_t, s_{t+1})$
- Randomly sample mini-batch from replay Buffer

- Minimise MSE loss by SGD:

$$L = (r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s, a, w))^2$$

- Update $w^{-} \leftarrow w$ every C steps

ϵ-greedy action selection methods



Exploration policy

Usually, set the parameter of $\epsilon = 0.1$

Pseudo Code

p = random()
if p < ϵ:
    random action
else:
    best action

https://livebook.manning.com/book/deep-reinforcement-learning-in-action/chapter-3/55

RPTU

RL for Text-Based Adventure Games

Text-based Adventure Games:

- Language-based interactions are part of our everyday life.
- An ideal testbed for language-based autonomous agents.

Text-based Adventure Games

At time step t :

Current State:

- **Observation**: Secret Entrance. […] Then the door swings open before you, opening into the abandoned city of Deephome.
- **Inventory:** You are carrying: King's Order, a lantern(providing light)
- **Look**: This is a rather dark and small room, having only two exits, […]. It has been three hundred years since your people lived here.

Total score: 6

RPTU

Text-based Adventure Games

At time step t :

Current State:

- **Observation**: Secret Entrance. […] Then the door swings open before you, opening into the abandoned city of Deephome.
- **Inventory:** You are carrying: King's Order, a lantern(providing light)
- **Look**: This is a rather dark and small room, having only two exits, […]. It has been three hundred years since your people lived here.

Total score: 6

Valid Action Space:

[say manaz, push mountain, close door, get in door, pull order down, put light down, pull all down]

RPTU

Text-based Adventure Games

Action: get in door
Score: +1
Reward: 1

Next State:

- **Observation**: Northern Guard Post [...] Your score has just gone up one point.
- **Inventory:** You are carrying: King's Order, a lantern(providing light)
- **Look**: This guard post is small and inconspicuous, [...]. To the southwest is a small door that leads out to the Main Hall and there is a tiny table in the middle of the room.

Total score: 7

RPTU

Text-based Adventure Games

At time step t :

Current State:

- **Observation**: Secret Entrance. […] Then the door swings open before you, opening into the abandoned city of Deephome.
- **Inventory:** You are carrying: King's Order, a lantern(providing light)
- **Look**: This is a rather dark and small room, having only two exits, […]. It has been three hundred years since your people lived here.

Total score: 6

Valid Action Space: [say manaz, push mountain, close door, get in door, pull order down, put light down, pull all down]

get in door

Reward: +1

Next State:

- **Observation**: Northern Guard Post […] Your score has just gone up one point.
- **Inventory:** You are carrying: King's Order, a lantern(providing light)
- **Look**: This guard post is small and inconspicuous, […]. To the southwest is a small door that leads out to the Main Hall and there is a tiny table in the middle of the room.

Total score: 7

Valid Action Space: [say manaz, get in northeast, get in southwest, put light down, put order down, put all down, open cabinet, push letter to ground]

RPTU

Text-based Adventure Games

Challenges:

- Combinatorial Action Space: Large and not fixed
- Commonsense Reasoning
- Knowledge Representation
- Sparse Reward Signals

Deep Reinforcement Learning

Deep RL = Deep learning + Reinforcement learning

Use the deep neural network to approximate Policy π or Value Function $v_{\pi}(s)$.

Action: a

$a \sim \pi(a|s)$

Agent

(Policy: SAC)

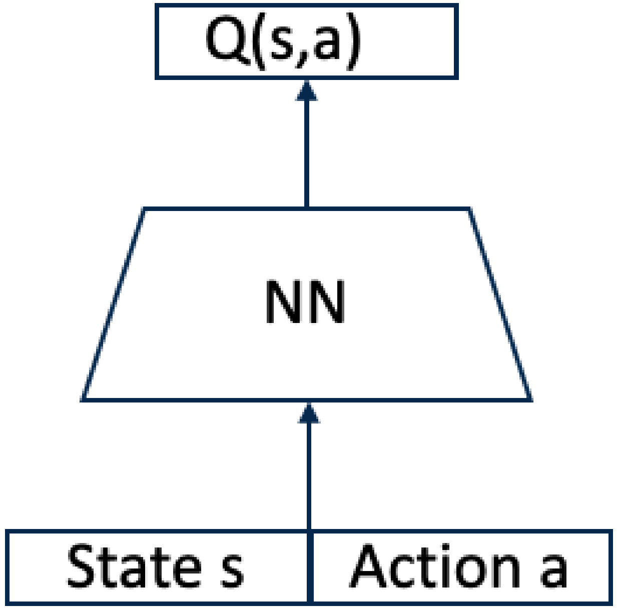
State: s

Environment

(Text-based games)

Reward: r

Deep Q-Learning



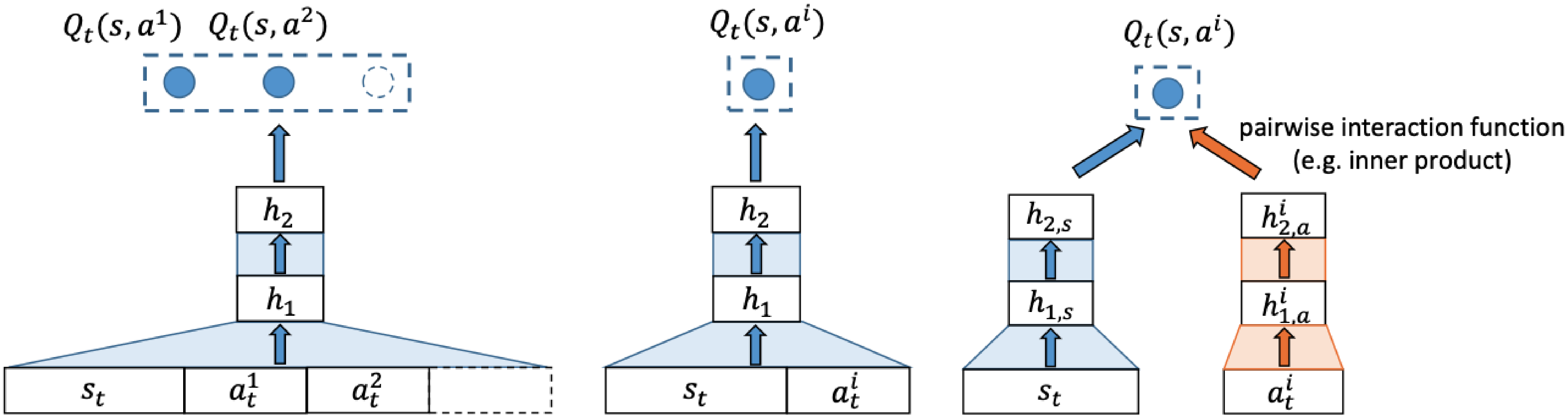
- A R**eplay Buffer** to store transitions:$(s_t, r_{t+1}, a_t, s_{t+1})$
- Randomly sample mini-batch from Replay Buffer
- Minimise MSE loss by SGD:

$$L = (r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w))^2$$

RPTU

NLP application using Q-learning: Text-based adventure games

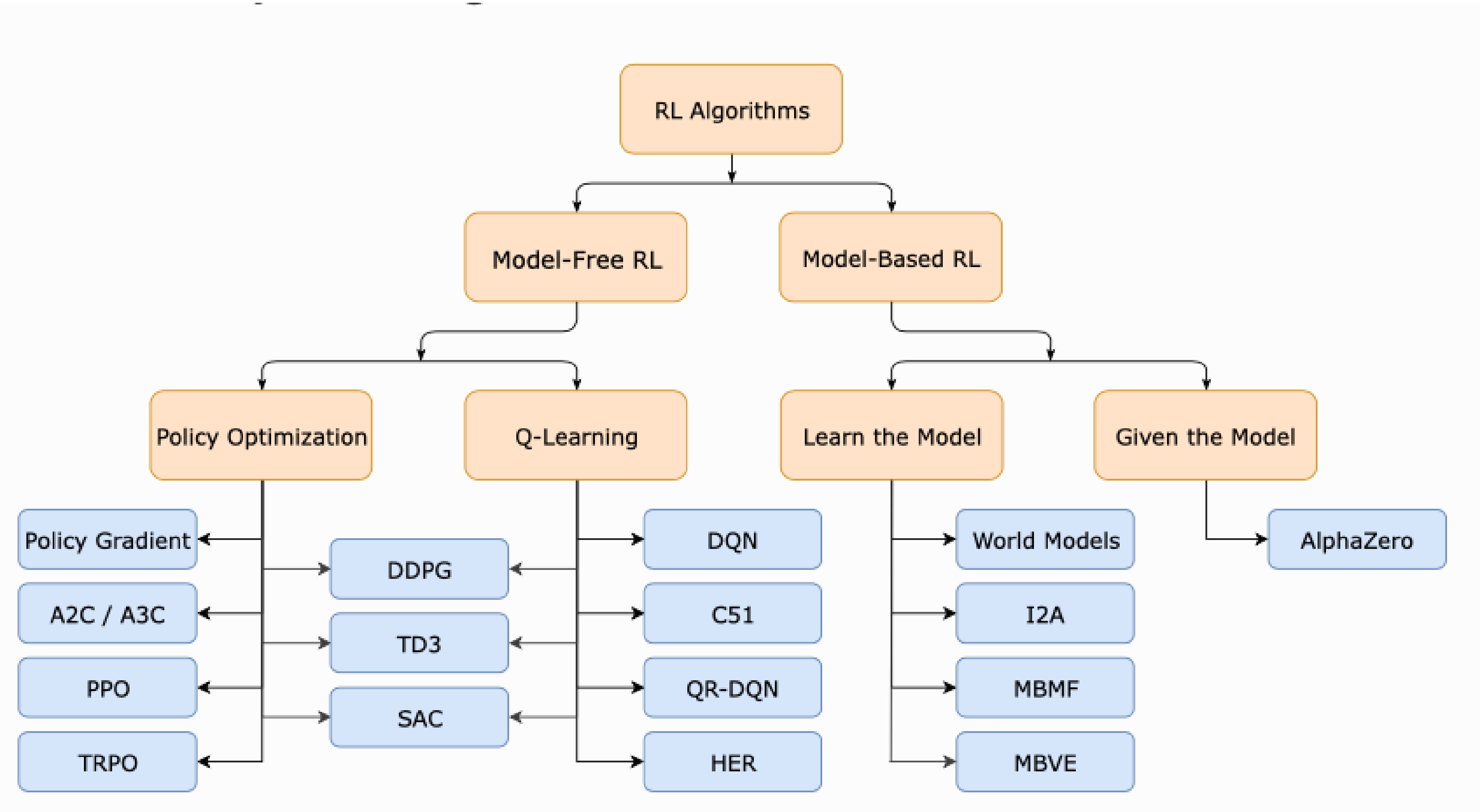
- Different deep Q-learning models:



(a) Max-action DQN (b) Per-action DQN (c) DRRN

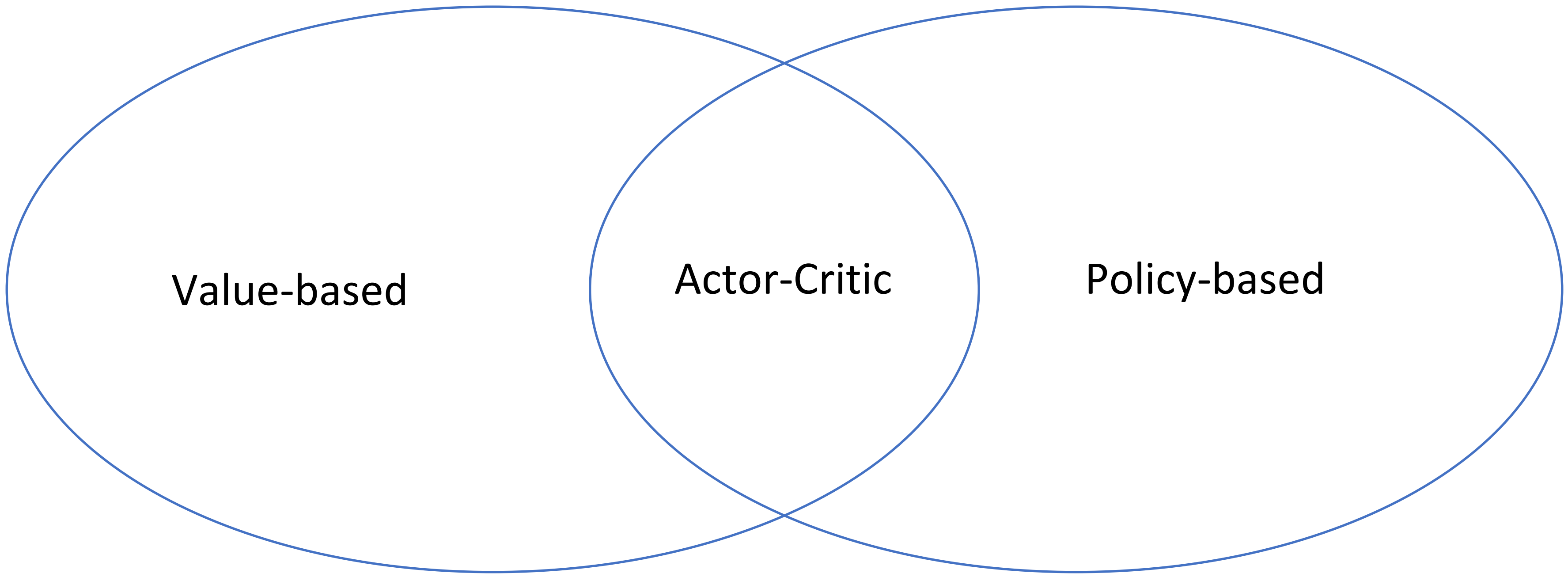
[Ji He,et.2016] Deep reinforcement learning with a natural language action space.

A Taxonomy of RL Algorithms



source:[spinning-up]

Overview RL Paradigms



RPTU

Recap: REINFORCE: Monte Carlo Policy Gradient

Pseudocode:

for each episode do:

Generate a trajectory Rollout $(S_0, A_0, R_1, \ldots S_{T-1}, A_{T-1}, R_T$ (using the current policy $\pi(.|., \theta)$

For each step of the episode $t = 0,1, \ldots, T-1$:

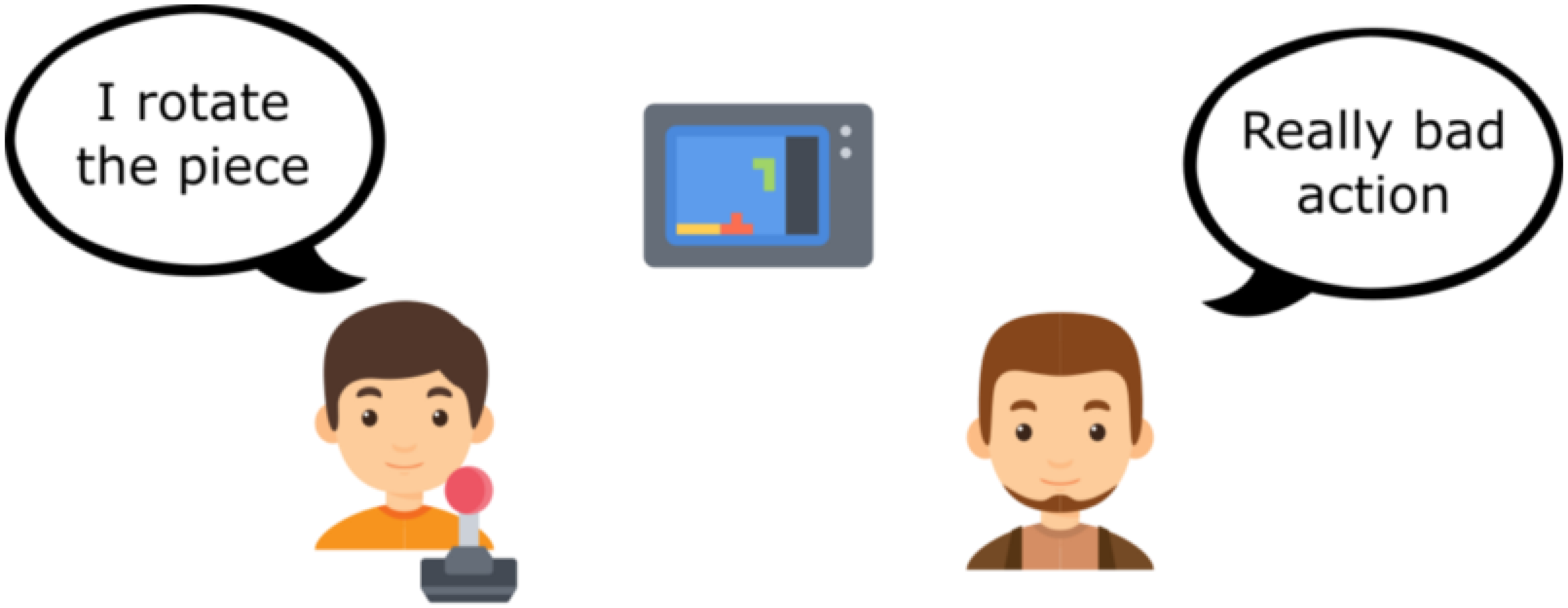
Compute the discounted cumulative future reward: $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

Update the policy parameter θ:

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_s | S_t, \theta)$$

[Sutton&Barton 2018] RPTU

Actor-Critic



Actor

Critic

Policy

Value-function

Actor-Critic

Critic Update parameters $\boldsymbol{w}$ of $v_{\boldsymbol{w}}$ by TD (e.g., one-step) or MC

Actor Update $\boldsymbol{\theta}$ by policy gradient

function One-step Actor Critic

Initialise s, $\boldsymbol{\theta}, w$

for t = 0, 1, 2, … **do**

Sample $A_t \sim \pi_{\boldsymbol{\theta}}(S_t)$

Sample R_{t+1} and S_{t+1}

$\delta_t = R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t)$ [one-step TD-error, or **advantage**]

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta\, \delta_t\, \nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ [TD(0)]

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, \delta_t\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t \mid S_t)$ [Policy gradient update (ignoring γ^t term)]

RPTU

Summary

- Value-based vs Policy-based RL
  - Q-learning learns the value of each state-action pair
  - Policy-based RL learns the policy directly
- Agentic workflows are enabled by combining multiple agents and improving their performance at specific tasks using RL

References

- [Sutton & Barto, 2018]Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
- [Li.et.al, 2016] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., & Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.
- [Bosselut et.al 2018] Bosselut, A., Celikyilmaz, A., He, X., Gao, J., Huang, P. S., & Choi, Y. (2018). Discourse-aware neural rewards for coherent text generation. *arXiv preprint arXiv:1805.03766*.
- [Hausknecht et.al, 2020] Hausknecht, M., Ammanabrolu, P., Côté, M. A., & Yuan, X. (2020, April). Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [Deepmind RL2021] https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021
- https://youtu.be/y3oqOjHilio?si=XZQI4bplaBFB7BPC
- [UCL course on RL]:https://www.davidsilver.uk/teaching/
- [Das, Rajarshi, et al., 2017] Das, Rajarshi, et al.(2017) "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning." *arXiv preprint arXiv:1711.05851*.
- [spinning-up]: https://spinningup.openai.com/en/latest/index.html
- https://huggingface.co/learn/deep-rl-course/unit8/clipped-surrogate-objective

RPTU

Policy objective

- Goal: given policy $\pi_\theta(s, a)$, find the best parameters θ
- How to measure the quality of a policy?
- In episodic environments: use the start value
  - $J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$
- In continuing environments: average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or average reward per time step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) R_s^a$$

Where $d^{\pi_\theta}(s)$ is the stationary distribution of Markov chain for π_θ

RPTU

Policy Gradient

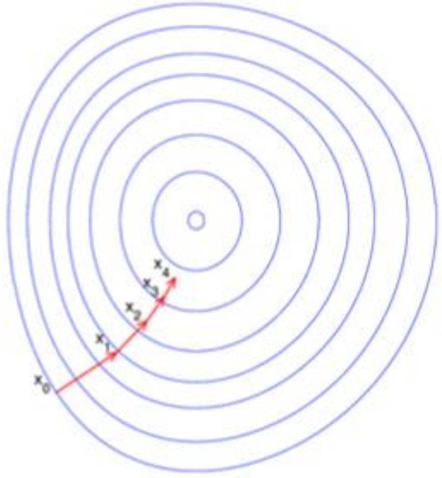
- Idea: ascent the gradient of the objective $J(\boldsymbol{\theta})$

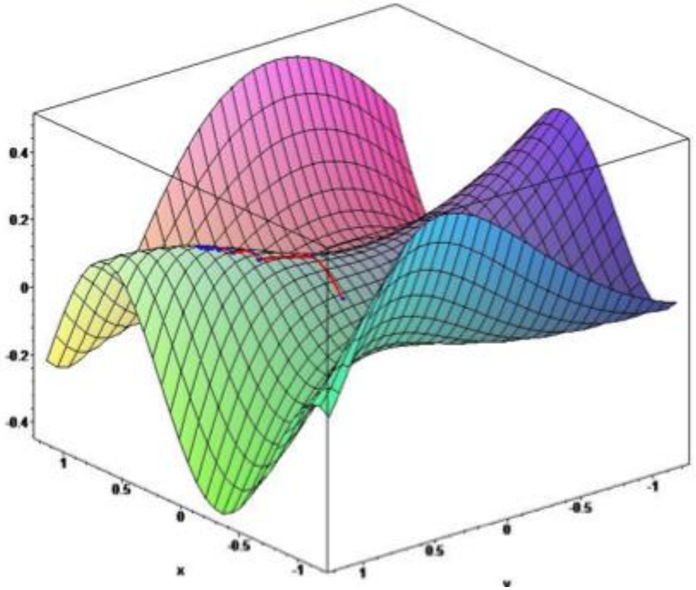
$$\Delta\boldsymbol{\theta} = \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the **policy gradient**

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_n} \end{pmatrix}$$

- and α is a step-size parameter
- Stochastic policies help ensure $J(\boldsymbol{\theta})$ is smooth (typically/mostly)





Source: [Deepmind RL2021]

Contextual Bandits Policy Gradient

- Consider a one-step case (a contextual bandit) such that $J(\theta) = E_{\pi_\theta}[R(S, A)]$.
- (Expectation is over d (states) and π (actions))
- (For now, d does not depend on π)
- We cannot sample R_{t+1} and then take a gradient:
- R_{t+1} is just a number and does not depend on θ!
- Instead, we use the identity:

$$\nabla_\theta E_{\pi_\theta}[R(S, A)] = E_{\pi_\theta}[R(S, A)\nabla_\theta \log \pi(A|S)]$$

(Proof on next slide)

- The right-hand side gives an expected gradient that can be sampled
- Also known as REINFORCE (Williams, 1992)

RPTU

Score function trick

Let $r_{sa} = \mathbb{E}\,[R(S, A) \mid S = s, A = a\,]$

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S, A)] &= \nabla_{\boldsymbol{\theta}} \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a|s)\, r_{sa} \\
&= \sum_s d(s) \sum_a r_{sa}\, \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) \\
&= \sum_s d(s) \sum_a r_{sa}\, \pi_{\boldsymbol{\theta}}(a|s) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}}(a|s)} \\
&= \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a|s)\, r_{sa}\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\
&= \mathbb{E}_{d,\pi_{\boldsymbol{\theta}}}[R(S, A)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S)]
\end{aligned}$$

Policy gradient theorem (episodic)

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}(s,a)$, let d_0 be the starting distribution over states in which we begin an episode. Then, the policy gradient of $J(\boldsymbol{\theta}) = \mathbb{E}\left[G_0 \mid S_0 \sim d_0\right]$ is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T} \gamma^t q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \mid S_0 \sim d_0\right]$$

where

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \end{aligned}$$

RPTU

source:[UCL course on RL, lecture 7]