(Deep) Reinforcement Learning

George Vouros

University of Piraeus, Greece

July 16, 2024

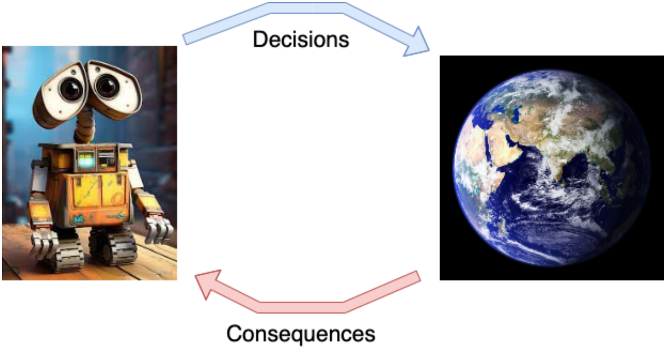
Outline

- Motivation & Introduction to Reinforcement Learning
- The Reinforcement Learning objective
- The Anatomy of Reinforcement Learning Algorithms
- Deep Reinforcement Learning Algorithms

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning provides a formalism for behaviour

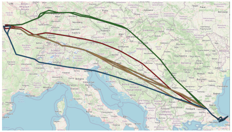
Basic Loop

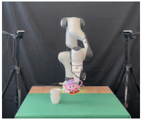


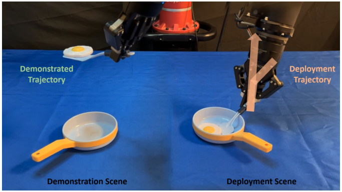
Introduction to (Deep) Reinforcement Learning

Reinforcement Learning provides a formalism for behaviour

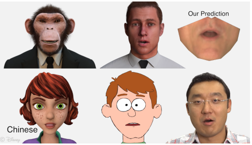
Examples (states, actions and trajectories)







R.L lab @ Imperial

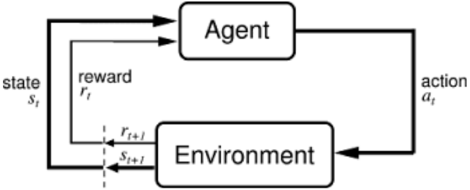


from Y. Yue

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning provides a formalism for behaviour

Basic Loop in a more rigorous way to introduce notation



Introduction to (Deep) Reinforcement Learning

What does the agent learns?

- A policy π: mapping from states S to actions $\mathcal{P}(A)$, based on past experience)
- Mind the dimensionality of state, action space

Introduction to (Deep) Reinforcement Learning

What does the agent learns?

- A policy π: mapping from states S to actions $\mathcal{P}(A)$, based on past experience)
- Mind the dimensionality of state, action space

curse of dimensionality

$$|S| = (255^3)^{200\times 200}$$

(more than atoms in the universe)

Figure from P.Abeel lectures on RL

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning involves

- Optimization
- Exploration
- Generalization
- Consequences and Rewards (sparse and/or delayed).

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning involves

- **Optimization:**
  - Find an optimal way to make decisions, yielding the best outcomes or at least very good outcomes.
    In other words: Find the optimal policy π^* that maximizes the sum of rewards that the agent gets while executing a task
- Exploration
- Generalization
- Consequences and Rewards (sparse and/or delayed).

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning involves

- Optimization
- **Exploration:**
  - Learn while interacting in the world (and failing)
  - Limited interaction means limited experience and knowledge (what would have happened if..?)
  - How much curiosity should be involved in the process? What if loosing everything while learning?
- Generalization
- Consequences and Rewards (sparse and/or delayed).

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning involves

- Optimization
- Exploration
- **Generalization:**
  - Is it possible to learn how to take optimal decisions at every possible state?
  - What about transferring decision-making knowledge between tasks?
- Consequences and Rewards (sparse and/or delayed).

Introduction to (Deep) Reinforcement Learning

Reinforcement Learning involves

- Optimization
- Exploration
- Generalization
- **Consequences and Rewards (sparse and/or delayed).**
    - Decisions at any particular state may have crucial impacts later on.
    - Temporal credit assignment when learning: what caused a very good or a very bad outcome?
    - Decisions when acting in the real world involve reasoning about long-term effects.

Introduction to (Deep) Reinforcement Learning

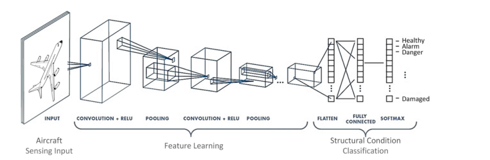
Why **Deep** Reinforcement Learning is important?

- Generalization abilities
- End-to-end training (what does it mean for RL)?

Introduction to (Deep) Reinforcement Learning

Why **Deep** Reinforcement Learning is important?

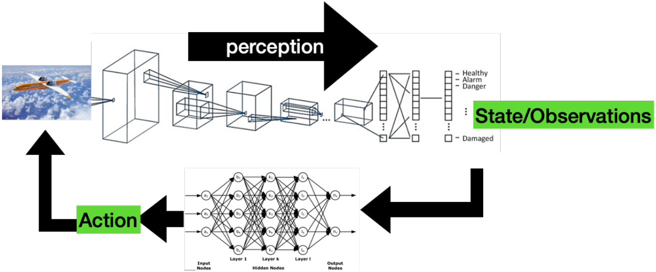
- Generalization abilities
- End-to-end training (what does it mean for RL)?



Introduction to (Deep) Reinforcement Learning

Why **Deep** Reinforcement Learning is important?

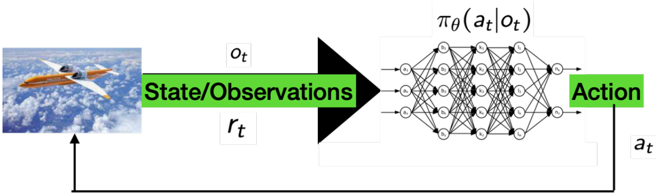
- Generalization abilities $\pi_\theta(a_t|o_t)$, a_t, r_t
- End-to-end training (what does it mean for RL)?



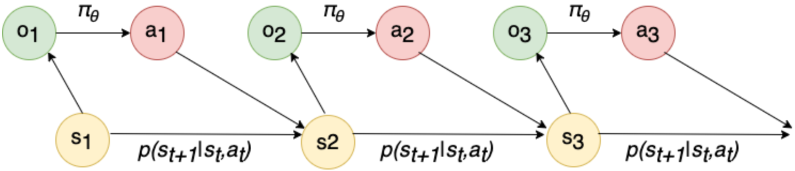
- Advances in DRL go in par with advances in DL.

Introduction to (Deep) Reinforcement Learning

Notation

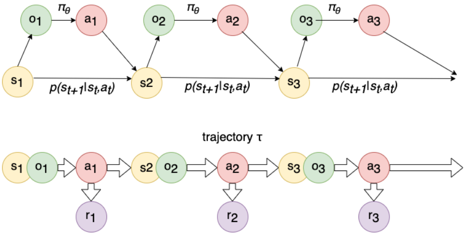


s_t state	$\pi_\theta(a_t\|s_t)$ fully observable
o_t observation	$\pi_\theta(a_t\|o_t)$ partially observable
a_t action	$r_t(s_t, a_t)$ reward



Introduction to (Deep) Reinforcement Learning

The objective given a POMDP $(S, A, \mathcal{O}, \mathcal{E}, \mathcal{T}, r)$, is to learn a policy that generates the best trajectories with high probability



Probability of τ given a policy π_θ

$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

Objective: tune θ to get

$\theta^* = argmax_\theta \mathbb{E}_{\tau \sim p_\theta}[\sum_t r_t], r_t = r(s_t, a_t)$

Introduction to (Deep) Reinforcement Learning

Probability of τ given a policy π_θ

$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

Objective: tune θ to get

$\theta^* = argmax_\theta \mathbb{E}_{\tau \sim p_\theta}[\sum_t r_t], r_t = r(s_t, a_t)$

Objective for finite time horizons

$\theta^* = argmax_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)}[r_t]$, where $p_\theta(s_t, a_t)$ the state, action marginal

Objective for infinite time horizons

$\theta^* = argmax_\theta \mathbb{E}_{(s,a) \sim \mu}[r(s, a)]$, where $\mu = p_\theta(s, a)$ the stationary distribution of states, actions

Introduction to (Deep) Reinforcement Learning

Definitions

- **Q**uality of action at state
  $Q^\pi(s_t, a_t) = \sum_t^T \mathbb{E}_{\pi_\theta}[r(s_t, a_t)|s_t, a_t]$
  Given a policy π and $Q^\pi(s, a)$, then we can improve π, by choosing $a = argmax_a Q^\pi(s, a)$
- **V**alue of state
  $V^\pi(s_t) = \sum_t^T \mathbb{E}_{\pi_\theta}[r(s_t, a_t)|s_t] = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$
  In case $Q^\pi(s, a) > V^\pi(s)$ then π can be modified by increasing the probability of a.
- **A**dvantage
  $A^\pi(s_t, a_t) = [Q^\pi(s_t, a_t) - V^\pi(s_t)]$

Bellman backup

$$Q^\pi(s, a) = r(s_t, a_t) + \mathbb{E}[(V^\pi_{t+1}(s_{t+1}))]$$

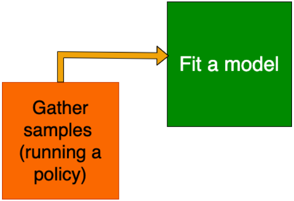
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms[1]

[1] adapted from the one proposed by S.Levine

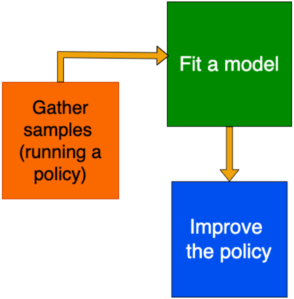
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms



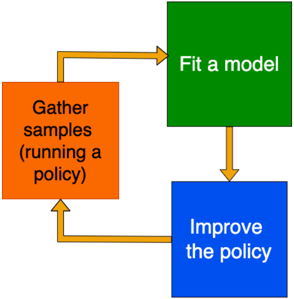
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms



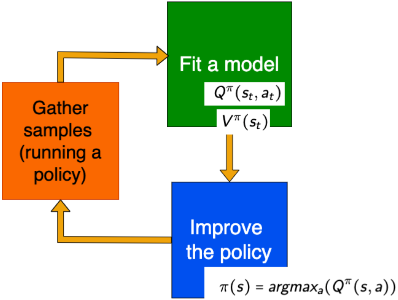
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms



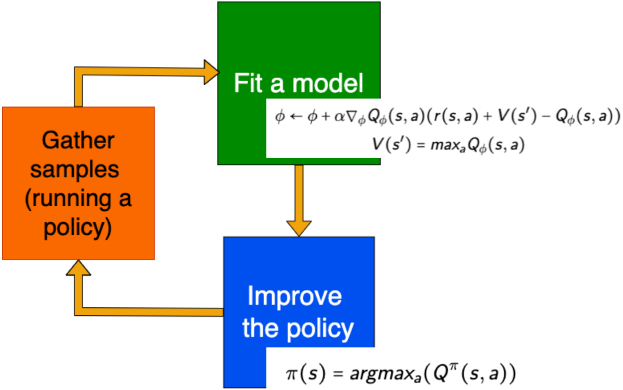
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms: Value based



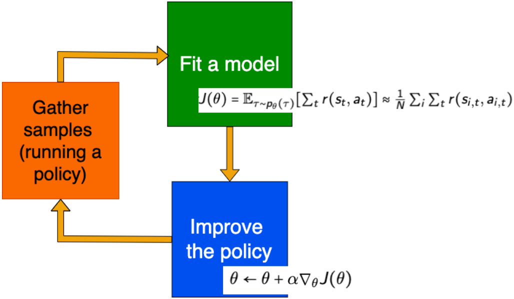
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms: Q-Learning



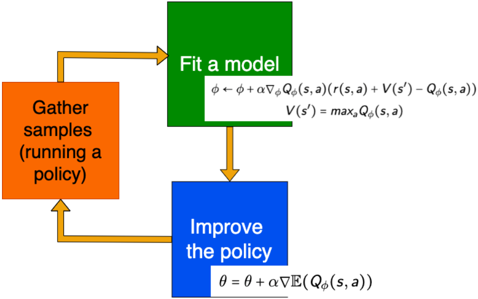
Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms: Direct policy gradient



Introduction to (Deep) Reinforcement Learning

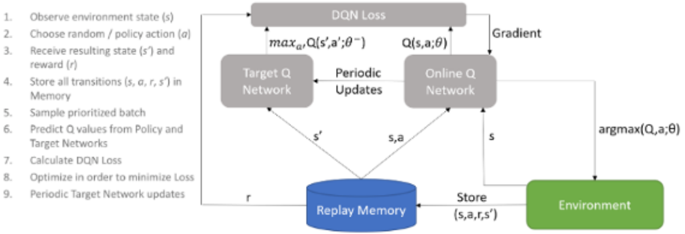
The anatomy of DRL algorithms: Actor Critic



Introduction to (Deep) Reinforcement Learning

The anatomy of Q-Learning algorithms

With a target network.

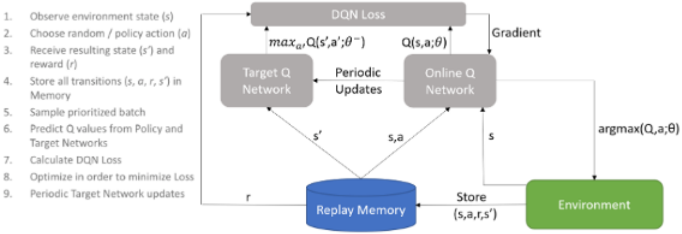


$$L_i(\theta_i) = \mathbb{E}_{(s,a,Rwd,s')\sim U(D)}[(Rwd + \gamma max_{a'} Q(s', a';\ \theta_i^-) - Q(s, a;\ \theta_i))^2]$$

Introduction to (Deep) Reinforcement Learning

The anatomy of Q-Learning algorithms

Double Q Learning with target.



$$L_i(\theta_i) = \mathbb{E}_{(s,a,Rwd,s')\sim U(D)}[(Rwd + \gamma max_{a'} Q(s', a';\ \theta_i^-) - Q(s, a;\ \theta_i))^2]$$

$$\boxed{Q^A(s,a) = Q^A(s,a) + \alpha(Rwd + \gamma Q^B(s', a^*) - Q^A(s,a))}$$

$$\boxed{Y_t^{QDouble} = Rwd_{t+1} + \gamma Q(s_{t+1}, argmax_a Q(s_{t+1}, a;\ \theta_t);\ \theta_t^-)}$$

Introduction to (Deep) Reinforcement Learning

So far...

- Motivation for DRL
- Notation and Definitions
- Specification of the DRL objective
- Anatomy of any DRL algorithm

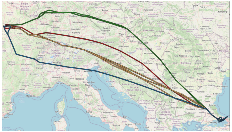
Stochastic and sub-optimal behaviour

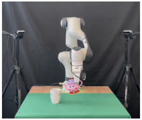
Important questions related to (D)RL

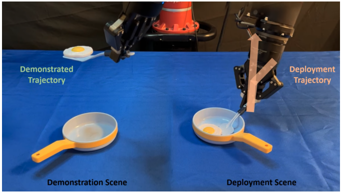
- Does (D)RL provide a reasonable model of human behaviour?
- Can we derive optimality and planning as probabilistic inference?

Introduction to (Deep) Reinforcement Learning

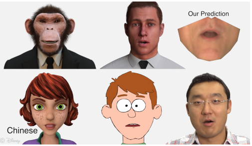
We need to take into account stochastic and sub-optimal behaviour







R.L lab @ Imperial



from Y. Yue

Introduction to (Deep) Reinforcement Learning

“Bounded” rationality

In any fully observed setting we can prove that there exist deterministic optimal policies, given that the objective is linear in the state, action marginals.
Recall that

- Objective for finite time horizons:

$$\theta^* = argmax_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r_t]$$

  , where $p_\theta(s_t, a_t)$ the state, action marginal

- Objective for infinite time horizons:

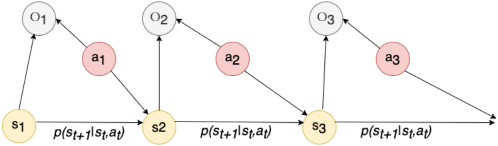
$$\theta^* = argmax_\theta \mathbb{E}_{(s,a)\sim\mu}[r(s,a)]$$

  , where $\mu = p_\theta(s, a)$ the stationary distribution of states, actions

So we need to recover rationality to take into account randomness

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[2]



Let $p(\mathcal{O}_t|s_t, a_t) = exp(r(s_t, a_t))$, then

$$
\begin{aligned}
p(\tau|\mathcal{O}_{1:T}) &= \frac{p(\tau, \mathcal{O}_{1:T})}{p(\mathcal{O}_{1:T})} \\
&\propto p(\tau)\prod_t exp(r(s_t, a_t)) \\
&= p(\tau)exp(\sum_t r(s_t, a_t))
\end{aligned}
$$

<u>Any case with low reward </u>is exponentially less likely to be chosen.

[2] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[3]

$$p(\tau|\mathcal{O}_{1:T}) = p(\tau)exp(\sum_t r(s_t, a_t))$$

Any case with low reward is exponentially less likely to be chosen.

- So we can model suboptimal behaviour - e.g. given demonstrations of near optimal choices while performing a task (inverse and imitation learning)
- Formulates stochastic behaviour - useful for exploration, generalization and transfer learning.
- We can apply inference algorithms to solve control and planning problems (under specific conditions)

[3] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for near-optimal behaviour[4]

Then we can compute the near optimal policy

$$\pi(a_t|s_t) = p(a_t|s_t, \mathcal{O}_{1:T}) = p(a_t|s_t, \mathcal{O}_{t:T}) = \frac{p(\mathcal{O}_{t:T}|s_t, a_t)}{p(\mathcal{O}_{t:T}|s_t)} p(a_t|s_t)$$

$$= \frac{\beta(s_t, a_t)}{\beta(s_t)} c$$

where, c is the action prior which is constant, assuming a uniform distribution, and β are backward messages computed recursively from $t = T$ to $t = 1$, assuming knowledge of transition probabilities.

[4] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Recovering rationality using probabilistic graphical models for sub-optimal behaviour

Given,

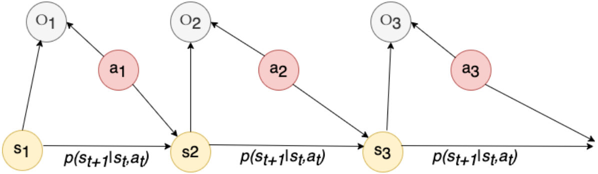
$$p(\mathcal{O}_t|s_t, a_t) \propto exp(r(s_t, a_t))$$
$$p(s_{t+1}|s_t, a_t)$$

Then we can compute backward messages recursively

$$\text{for } t = T - 1 \text{ to } 1:$$
$$\beta(s_t, a_t) = p(\mathcal{O}_t|s_t, a_t)\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}[\beta_{t+1}(s_{t+1})]$$
$$\beta(s_t) = \mathbb{E}_{a_t\sim p(a_t|s_t)}[\beta(s_t, a_t)]$$

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[5]



We can also compute forward messages (useful for inverse reinforcement learning)

$$a_t(s_t) = p(s_t|\mathcal{O}_{1:t-1})$$

recursively, starting from the usually known $a_1(s_1)$, as well as the marginal probabilities

$$p(s_t|\mathcal{O}_{1:T}) \propto \beta_t(s_t)a_t(s_t)$$

[5] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[6]

$$\beta(s_t, a_t) = p(\mathcal{O}_t | s_t, a_t) \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\beta_{t+1}(s_{t+1})]$$

$$\beta(s_t) = \mathbb{E}_{a_t \sim p(a_t|s_t)}[\beta(s_t, a_t)]$$

let $Q_t(s_t, a_t) = log \beta_t(s_t, a_t) = r(s_t, a_t) + log \mathbb{E}[exp(V_{t+1}(s_{t+1}))]$
let $V_t(s_t) = log \beta_t(s_t) = log \int exp(Q_t(s_t, a_t))\, da_t$

Notice:
1. The **optimistic transition implied by** Q_t and
2. The **softmax in the definition of** $V_t(s_t)$, as $Q_t(s_t, a_t)$ gets bigger.

[6] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[7]

$Q_t(s_t, a_t) = log \beta_t(s_t, a_t)$
$V_t(s_t) = log \beta_t(s_t)$

$$\text{for } t = T-1 \text{ to } 1:$$
$$Q_t(s_t, a_t) = r(s_t, a_t) + log \mathbb{E}[exp(V_{t+1}(s_{t+1}))]$$
$$V_t(s_t) = log \int exp(Q_t(s_t, a_t))\, da_t$$

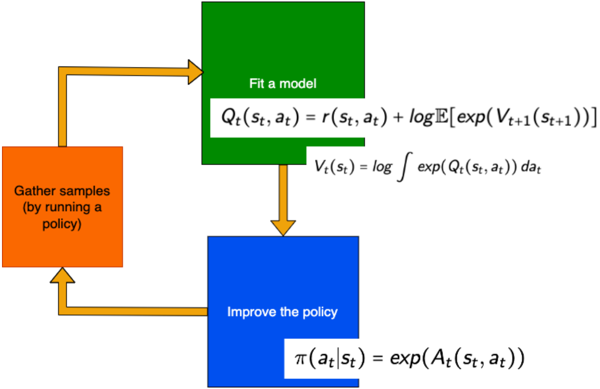
$\pi(a_t|s_t) = \frac{\beta(s_t, a_t)}{\beta(s_t)} = exp(Q_t(s_t, a_t) - V_t(s_t)) = exp(A_t(s_t, a_t))$
adding temperature we can balance between deterministic ($\alpha \to 0$) and stochastic (soft) ($\alpha \to \infty$) policy:

$$\pi(a_t|s_t) = \frac{\beta(s_t, a_t)}{\beta(s_t)} = exp(\frac{1}{\alpha} Q_t(s_t, a_t) - \frac{1}{\alpha} V_t(s_t)) = exp(\frac{1}{\alpha} A_t(s_t, a_t))$$

[7] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

Introduction to (Deep) Reinforcement Learning

The anatomy of DRL algorithms: Soft Q-Learning with optimality bias



Introduction to (Deep) Reinforcement Learning

Variational inference[8]

$= \exp(Q_t(s_t, a_t) - V_t(s_t)) = exp(A_t(s_t, a_t))$

To
avoid the optimistic bias of increasing the probabilities of actions that result into high rewards in very infrequent cases, we need to consider how to act near optimally given the "original" [9] transition probabilities.

Variational inference leads to obtaining an approximation $\hat{p}(s_{1:T}, a_{1:T})$ of $p(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T})$ with dynamics $p(s_{t+1}|s_t, a_t)$

[8] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

[9] i.e. Those not affected by our optimized decisions, $p(s_{t+1}|s_t, a_t, \mathcal{O}_{1:T})$

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour

Variational inference leads to obtaining an approximation $\hat{p}(s_{1:T}, a_{1:T})$ of $p(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T})$ with dynamics $p(s_{t+1}|s_t, a_t)$. Let

$$\hat{p}(s_{1:T}, a_{1:T}) = p(s_1) \prod_t p(s_{t+1}|s_t, a_t)\hat{p}(a_t|s_t)$$

It is proved that by setting the variational lower bound

$$logp(\mathcal{O}_{1:T}) \geq \mathbb{E}_{(s_{1:T}, a_{1:T})\sim\hat{p}}[\sum_t r(s_t, a_t) - log\,\hat{p}(a_t|s_t)]$$

this **translates to maximize the reward and action entropy:**

$$logp(\mathcal{O}_{1:T}) \geq \sum_t \mathbb{E}_{(s_t, a_t)\sim\hat{p}}[r(s_t, a_t) + \mathcal{H}(\hat{p}(a_t|s_t)]$$

Introduction to (Deep) Reinforcement Learning

Recovering rationality using probabilistic graphical models for sub-optimal behaviour[10]

$$logp(\mathcal{O}_{1:T}) \geq \sum_t \mathbb{E}_{(s_t,a_t)\sim\hat{p}}[r(s_t,a_t) + \mathcal{H}(\hat{p}(a_t|s_t)]$$

is optimized when $\hat{p}(a_t|s_t) \propto exp(Q(s_t,a_t))$ resulting into

$$\pi(a_t|s_t) = \hat{p}(a_t|s_t) = exp(Q(s_t,a_t) - V(s_t))$$

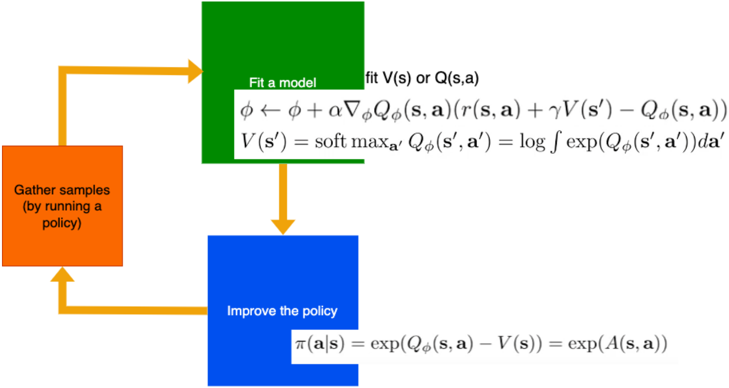
$$V(s_t) = log \int exp(Q_t(s_t,a_t))\, da_t$$

with **the regular (unbiased) Bellman backup**

$$Q_t(s,a) = r(s_t,a_t) + \mathbb{E}[V_{t+1}(s_{t+1})]$$

[10] Levine (2018) "RL and control as probabilistic inference: Tutorial and review"

The anatomy of DRL algorithms: Soft Q-Learning



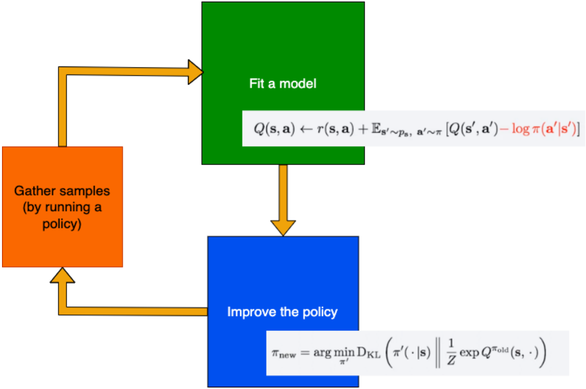
$$\phi \leftarrow \phi + \alpha \nabla_\phi Q_\phi(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\phi(\mathbf{s}, \mathbf{a}))$$

$$V(\mathbf{s}') = \text{soft}\max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}') = \log \int \exp(Q_\phi(\mathbf{s}', \mathbf{a}'))d\mathbf{a}'$$

$$\pi(\mathbf{a}|\mathbf{s}) = \exp(Q_\phi(\mathbf{s}, \mathbf{a}) - V(\mathbf{s})) = \exp(A(\mathbf{s}, \mathbf{a}))$$

Introduction to (Deep) Reinforcement Learning

Soft Actor Critic (T.Haarnooja et al., 2018)



$$D_{KL}(\pi_\theta(a|s)||\frac{1}{Z}exp(Q_\phi(s,a))) = \mathbb{E}_s[\mathbb{E}_{a\sim\pi_\theta(s)}[log\pi_\theta(a|s) - Q_\phi(s,a)]]$$

Introduction to (Deep) Reinforcement Learning

So far...

- Recovering rationality considering sub-optimal behaviour
- Incorporating MaxEnt terms in the RL objective
- Q-Learning and Soft Q-learning
- Soft Actor Critic

Introduction to (Deep) Reinforcement Learning

Policy Gradient

Goal: $maxJ(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t r(s_t, a_t)] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[(\sum_t \nabla_\theta log \pi_\theta(a_t|s_t))(\sum_t r(s_t, a_t))]$$

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (\sum_t \nabla_\theta log \pi_\theta(a_t|s_t))(\sum_t r(s_t, a_t))$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

REINFORCE Algorithm:

1. sample τ_i from $\pi_\theta(a_t|s_t)$
2. $\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (\sum_t \nabla_\theta log \pi_\theta(a_t|s_t))(\sum_t r(s_t, a_t))$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go to 1.

Introduction to (Deep) Reinforcement Learning

Policy Gradient with Causality and baselines

1. sample τ_i from $\pi_\theta(a_t|s_t)$
2. $\nabla J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(\sum_t \nabla_\theta log \pi_\theta(a_t|s_t))(\sum_t r(s_t,a_t))$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go to 1.

where

$$\text{Reward to go: } \hat{Q}(s_t,a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'},a_{t'}|s_t,a_t)]$$

$$\text{It can simply be: } \hat{Q}(s_t,a_t) = \sum_{t'=t}^{T} r(s_t,a_t)$$

and

$$\text{Baseline: } b = V(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q(s_t,a_t)]$$

$$\text{It can simply be: } b = \frac{1}{N}\sum_i \hat{Q}(s_t^i,a_t^i)$$

Introduction to (Deep) Reinforcement Learning

Policy Gradient with Causality and baselines

$$\text{Reward to go: } Q(s_t, a_t) \approx \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'} | s_t, a_t)]$$

$$\text{Baseline: } b = V(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q(s_t, a_t)]$$

$$\text{Advantage: } A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

Usually, a simple fit $\hat{A}(s_t, a_t) suffices$.
So,

1. sample τ_i from $\pi_\theta(a_t|s_t)$
2. $\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (\sum_t \nabla_\theta log \pi_\theta(a_t^i | s_t^i)) \hat{A}(s_t^i, a_t^i)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go to 1.

Introduction to (Deep) Reinforcement Learning

Policy Gradient with Causality and baselines and Importance sampling

Making the algorithm off-policy (i.e. exploit samples from previous iteration):

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_t \nabla_\theta log \pi_\theta(a_t|s_t)(\sum_t r(s_t, a_t))] =$$

$$\mathbb{E}_{\tau \sim \pi_{\theta'}(\tau)}[\sum_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta log \pi_\theta(a_t|s_t)(\sum_t r(s_t, a_t)]$$

The Algorithm:

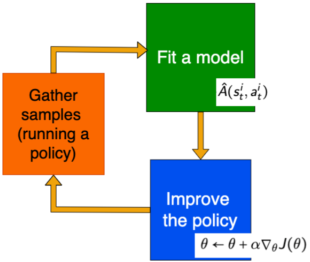
1. sample τ_i from $\pi_{\theta'}(a_t|s_t)$
2. $\nabla J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(\sum_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta log \pi_\theta(a_t^i|s_t^i))\hat{A}(s_t^i, a_t^i)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go to 1.

Introduction to (Deep) Reinforcement Learning

Policy Gradient with Causality and baselines and Importance sampling

The Algorithm:

1. sample τ_i from $\pi_{\theta'}(a_t|s_t)$
2. $\nabla J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(\sum_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta log \pi_\theta(a_t^i|s_t^i))\hat{A}(s_t^i, a_t^i)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go to 1.



Introduction to (Deep) Reinforcement Learning

Trust Region Policy Optimization (TRPO)

J.Schulman et al., "Trust Region Policy Optimization", 2015

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

- Also worth considering using a penalty instead of a constraint

$$\underset{\theta}{\text{maximize}} \qquad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] - \beta \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

Introduction to (Deep) Reinforcement Learning

Proximal Policy Optimization (PPO)

J.Schulman et al.,"Proximal Policy Optimization", 2017

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$
- Clipping prevents policy from having incentive to go far away from θ_{k+1}
- Clipping seems to work at least as well as PPO with KL penalty, but is simpler to implement

Introduction to (Deep) Reinforcement Learning

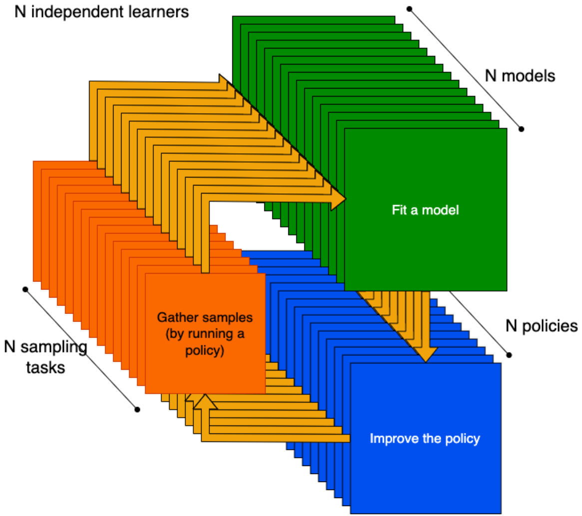
So far...

- Policy Gradient (addressing variance and bias)
- Importance sampling for sample efficiency
- Natural Policy Gradient (TRPO)
- Proximal Policy Optimization (PPO)

Multi-agent Reinforcement Learning (MARL)

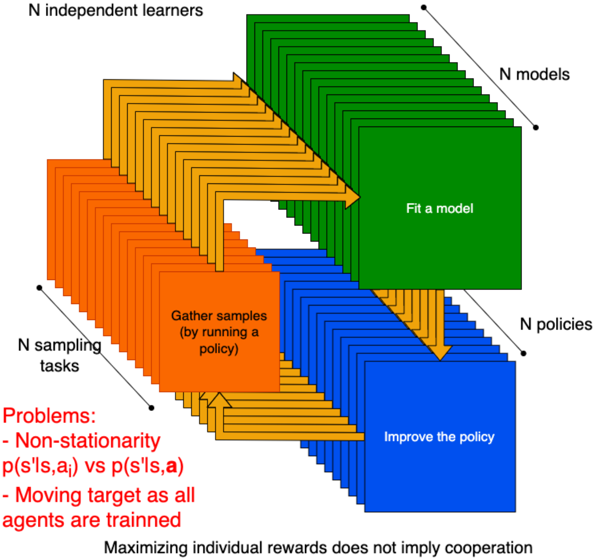
Introduction to (Deep) Reinforcement Learning

Multi-agent Reinforcement Learning (MARL)



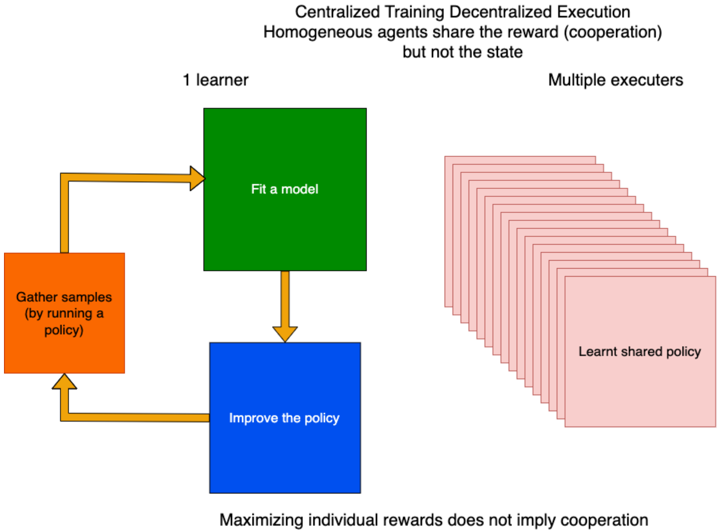
Introduction to (Deep) Reinforcement Learning

Multi-agent Reinforcement Learning (MARL)



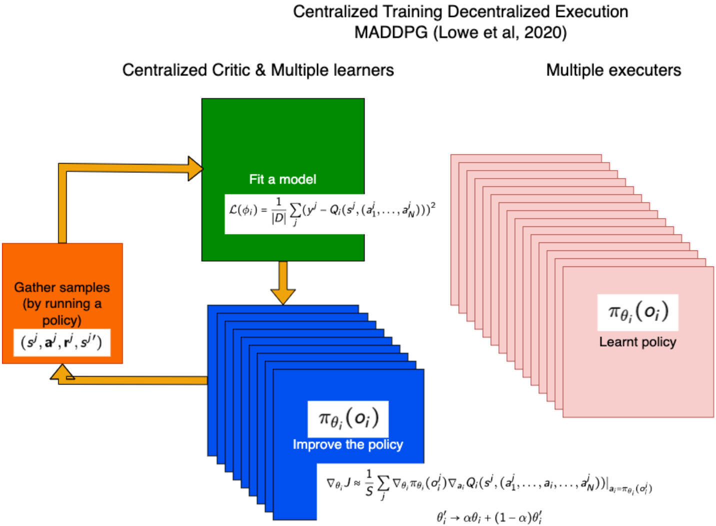
Introduction to (Deep) Reinforcement Learning

Multi-agent Reinforcement Learning (MARL)



Introduction to (Deep) Reinforcement Learning

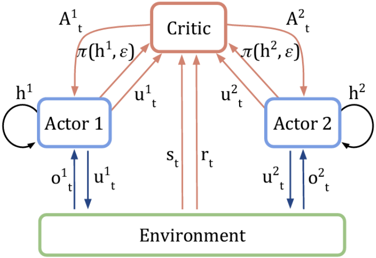
Multi-agent Reinforcement Learning (MARL)



Introduction to (Deep) Reinforcement Learning

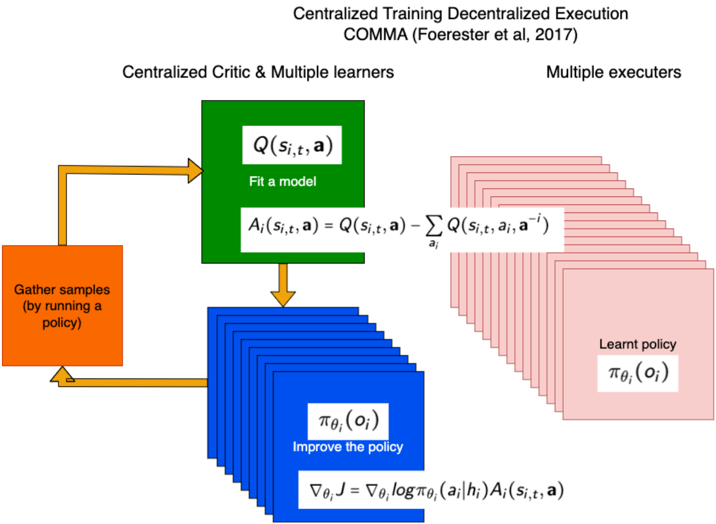
Multi-agent Reinforcement Learning (MARL)

COMMA (Foerester et al, 2017)



Introduction to (Deep) Reinforcement Learning

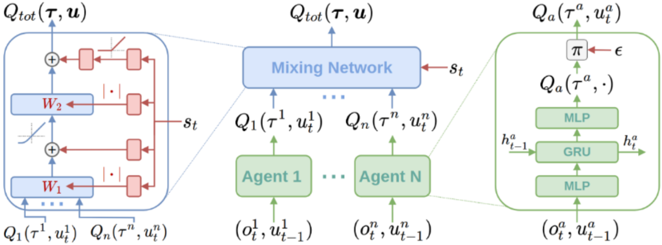
Multi-agent Reinforcement Learning (MARL)



Introduction to (Deep) Reinforcement Learning

Multi-agent Reinforcement Learning (MARL)

Between Independent Learners and Centralized Training
QMIX (Rashid et al, 2018)



$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) \right)^2 \right]$$

Introduction to (Deep) Reinforcement Learning

Multi-agent Reinforcement Learning (MARL)

Centralized Training Decentralized Execution
DGN (Jiang et al, 2017)(Papadopoulos et al, 2024)