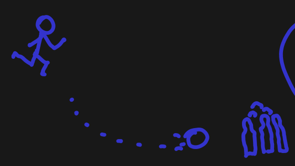
19/08/2020

Reinforcement Learning

- Sequential decision making.
  - Not needed when:
    - ↳ isolated decision (classification, regression)
    - ↳ that decision does not affect future inputs or decisions.
- Some applications can not ignore the dependence of the



Current decision on the Future

(non iid data)

The Plan:

1. Multi-task RL problem
2. Policy grads. & multi-task / meta counterparts/equivalents
3. Q-learning
4. Multi-task Q-learning.

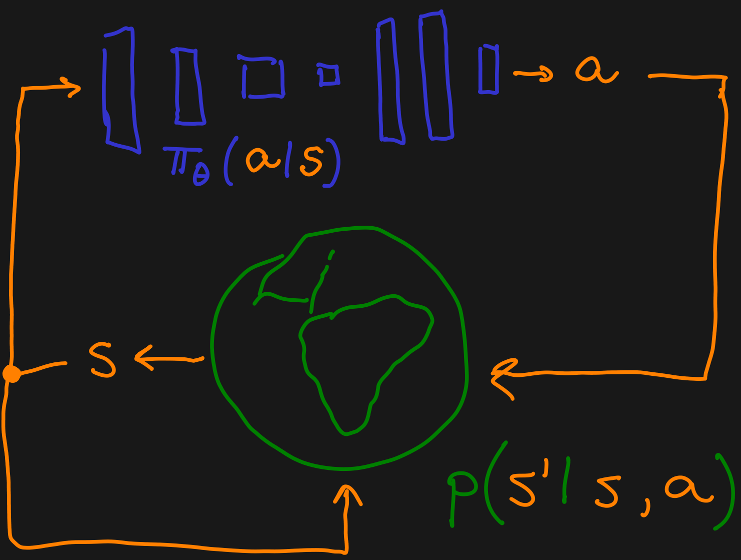
1.

Imitation Learning
- ↳ Lots of data
- ↳ Short horizon decision problems (compound of errors otherwise)

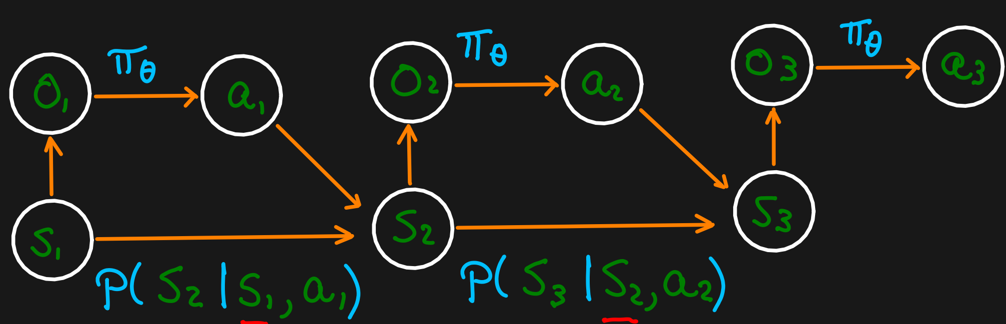
Reward Function.

MDP: defined by
- s
- a
- $r(s,a)$
- $p(s' \mid s,a)$

The goal of RL:



As Graphical Model:



$p(S_{t+1} \mid S_t, a_t)$: Markov Prop.

$$\theta^* = \arg\max \mathbb{E}_{(s,a)\sim p_\theta(s,a)}\left[r(s,a)\right]$$

$$\theta^* = \arg\max \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}\left[r(s_t,a_t)\right]$$

Finite horizon: Episode = T steps

RL Tasks

Supervised Learn.

Task:

$$\underbrace{\mathcal{T}_i \triangleq \{ \overset{\text{prior}}{p_i(x)}, \overset{\text{likelihood}}{p_i(y|x)}}_{\text{data generating distrib.}}, \overset{\text{loss}}{\mathcal{L}_i} \}$$

Reinforcement Learning

Task:

$$\mathcal{T}_i \triangleq \{ \underset{\substack{\text{state}\\\text{space}}}{S_i}, \underset{\substack{\text{action}\\\text{space}}}{A_i}, \underset{\substack{\text{initial}\\\text{state}\\\text{distribution}}}{p_i(s_1)}, \underset{\text{dynamics}}{p_i(s'|s,a)}, \underset{\text{reward}}{r_i(s,a)} \}$$

Recommendation System as RL Tasks

each people = different task.

↳ personalized

$$\left.\begin{matrix} p_i(s'|s,a) \\ r_i(s,a) \end{matrix}\right\} \text{Vary across tasks}$$

Character animation

- $r_i(s,a)$ vary across maneuvers

- $\left.\begin{matrix} p_i(s_1) \\ p_i(s'|s,a) \end{matrix}\right\}$ vary across garment and initial state

- # Multi-Robot RL

  S_i, A_i, $p_i(s_1)$, $p_i(s'|s,a)$ vary across robots

Alternative view:

A task identifier z_i is part of the state s

$s = (\bar{s}, z_i)$

↑ original state

Going back to the first definition

$$\mathcal{T}_i \triangleq \{ S_i, A_i, p_i(s_1), p_i(s'|s,a), r_i(s,a) \}$$

We can see how as the task identifier is now part of s, $\begin{cases} p_i(s'|s,a), \\ r_i(s,a) \end{cases}$ becomes $\begin{cases} p(s'|s,a), \\ r(s,a) \end{cases}$

20/08/2020

$$\mathcal{T}_i \triangleq \{ S_i, A_i, p_i(s_1), p(s'|s,a), r(s,a) \}$$

- And now it can be cast as an MDP

$$\{\mathcal{T}_i\} = \{ \cup S_i, \cup A_i, \frac{1}{N}\sum_i p_i(s_1), p(s'|s,a), r(s,a) \}$$

This says: we can apply std. single task RL to the multi-task problem with this view of multi-task RL

The Goal of Multi-task

Multi-task

$S = (\bar{s}, z_i)$ task identifier

z_i could be: 1-hot task ID

lang. description

- Desired Goal State : $z_i = s_g$

this is called

"Goal-Conditioned RL"

where the reward is some distance to s_g

$$r(s) = r(\bar{s}, s_g) = -d(\bar{s}, s_g)$$

eg: Euclidean ℓ_2

Sparse 0/1 : $\mathbb{1}\{\bar{s} = s_g\}$

RL Algorithm anatomy.

Generate samples (run policy)

Fit model to estimate return

- Compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} \cdot r_{t'}$ (Monte Carlo Policy Grad.)
- Fit $Q_\phi(s,a)$ (Actor Critic Q-learning)
- Estimate $p(s' \mid s, a)$ (model based)

Improve the Policy

- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (Policy Grad)
- $\pi(s) = \arg\max Q_\phi(s,a)$ (Q-learning)
- Optimize $\pi_\theta(a \mid s)$ (model based)

We focus to model-free for now.

Evaluating the objective

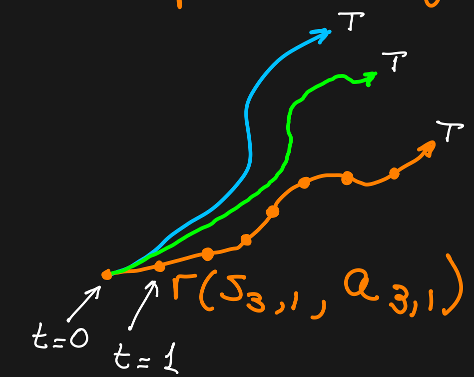
sum over time period : $r(\tau)$

$$\theta^* = \arg\max_{\theta} \underbrace{\mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right]}_{J(\theta)}$$

$$J(\theta) \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

sum over samples from π_θ

Samples from π_θ



Direct Policy Differentiation

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[r(\tau)\right] = \int \pi_\theta(\tau)\, r(\tau)\, d\tau$$

integrates over trajectories

dif:

$$\nabla_\theta J(\theta) = \int \underbrace{\nabla_\theta \pi_\theta(\tau)}_{= \pi_\theta(\tau) \cdot \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)}} r(\tau)\, d\tau$$

$$= \int \pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau) \cdot r(\tau) \cdot d\tau$$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) \cdot r(\tau)\right]$$

Probability of trajectory under policy π

$$\underbrace{\pi_\theta(s_1, a_1, \dots, s_T, a_T)}_{\pi_\theta(\tau)} = \underbrace{p(s_1)}_{\text{initial state density}} \cdot \prod_{t=1}^{T} \underbrace{\pi_\theta(a_t \mid s_t)}_{\text{policy prob.}} \cdot \underbrace{p(s_{t+1} \mid s_t, a_t)}_{\text{dynamics probability}}$$

$\prod_{t=1}^{T}$ ← over all trajectory time-steps

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)$$

$$\nabla_\theta \log \pi_\theta(\tau) = \underbrace{\emptyset}_{\text{does not depends on } \theta} + \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) + \underbrace{\emptyset}_{\text{does not depends on } \theta}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \underbrace{\sum_{i=1}^{N}}_{\text{generate samples}} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\right) \underbrace{\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)}_{\text{estimate return}}$$

and then

$$\underbrace{\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)}_{\text{improve the policy}}$$

REINFORCE

1. Sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$

2. $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(a_t^i, s_t^i)\right)\left(\sum_t r(s_t^i, a_t^i)\right)$

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

Imitation Learning

↳ Max. Likelihood of Expert actions

- Policy gradient

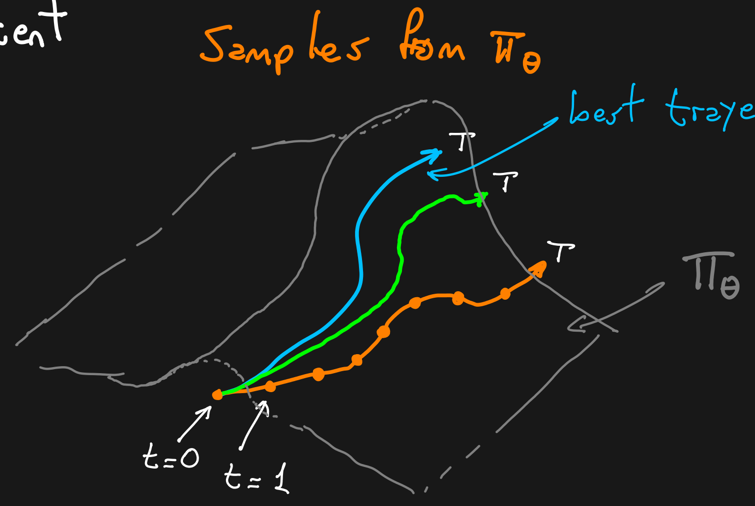
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(a_t^i, s_t^i)\right)\underbrace{\left(\sum_t r(s_t^i, a_t^i)\right)}_{\text{Maximizes the probability of actions that have high reward}}$$

- Maximum Likelihood

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N}\sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(a_t^i, s_t^i)\right)$$

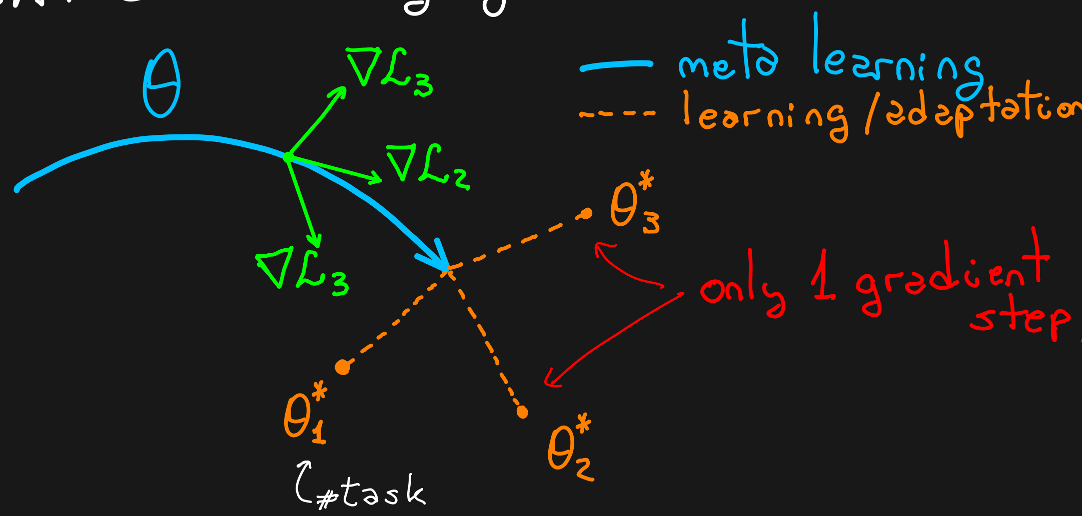
NOT weighted by reward

Policy gradient



We can directly apply RL + Multi-task

Example: MAML + Policy gradient



- Black Box meta-learning + Policy gradients

  3D Mazes - test on solve unseen mazes

- Combination with
  - Q-learning
  - Actor Critic

  } More difficult as are not gradient based algorithms: Bootstraping is Dynamic Programming

- Variance on Policy Grad.
  - can be mitigated with baselines, trust regions
    - see Hado van Hasselt video on baselines.

Importance weights can help to rehuse data.

Value-Based RL

Value function

- $V^{\pi}(s_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(s_{t'}, a_{t'} | s_t) \right]$

Q function

- $Q^{\pi}(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(s_{t'}, a_{t'} | s_t, a_t) \right]$

equiv.

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

Bellman eq:

- For the optimal policy π^*

$$Q^*(s,a) = \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[r(s,a) + \gamma \max Q^*(s',a') \right]$$

Best reward in the future

Fitted Q-iteration: DP algorithm that leads to Bellman eq.

1. collect dataset $\{(s_i, a_i, s_i', r_i)\}$
2. set $y_i \leftarrow r(s_i, a_i) + \gamma \cdot \max_{a_i'} Q_\phi(s_i', a_i')$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \| Q_\phi(s_i, a_i) - y_i \|^2$

K iters

- Now we have $Q_\phi(s,a)$, get policy $\pi(a|s)$ from $\arg\max_a Q_\phi(s,a)$

↳ Off policy algo.

↳ can use replay buffer

- Not grad. descent algorithm: is a DP algo. ⇒ tricky to combine with MAML or Black Box approach.
- Can be readily extended to multi-task / goal-conditioned RL

Multi-task RL Algorithms

- Policy

  condition also on task z_i

  $$\pi_\theta(a|\bar{s}) \to \pi_\theta(a|\bar{s}, z_i)$$

- Q-function

  $$Q_\phi(\bar{s}, a) \to Q_\phi(\bar{s}, a, z_i)$$

Analogous as Multi task S.Learn., but

- Data distribution is controlled by the agent
- Data sharing across tasks?
- If known dynamics of MDP while changing across tasks.
  ↳ how can we leverage this knowledge?

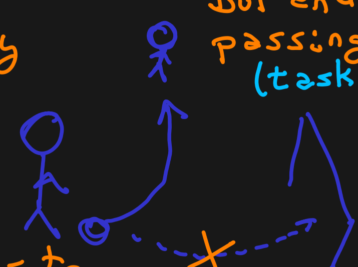
(Spa.: "Comprensión Retrospectiva")

Hindsight relabeling / aka Hindsight Experience Replay "HER"

Relabel experience with other task id and store both

task 1 : {same experience}

task 2 : {same experience}



Goal-conditioned RL with hindsight relabeling.

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, s_{goal}, r_{1:T})\}$ using some policy.
2. Store data in replay buffer

$$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$$

3. Perform hindsight relabeling
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal.

$$\mathcal{D}_k' = \{(s_{1:T}, a_{1:T}, s_T, r_{1:T}')\}$$

   where $r_t' = -d(s_t, s_T)$ ← distance to (new) goal

   b. Store $\mathcal{D}_k'$ in Replay Buffer

$$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k'$$

4. Update Policy using replay buffer $\mathcal{D}$

Repeat ↑ 1.

Other relabeling in a.? "any state"

Result:

↳ Exploration gets more efficient
(exploring for 1 task helps exploration in other tasks)

Multi-task RL with relabeling

Very similar structure:

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, z_i, r_{1:T})\}$ using some policy.
2. Store data in replay buffer

$$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$$

3. Perform hindsight relabeling

Choose
- Randomly
- task(s) in which the trajectory gets high reward.

a. Relabel experience in $\mathcal{D}_k$ for task $\mathcal{T}_j$

$$\mathcal{D}'_k = \{(s_{1:T}, a_{1:T}, z_j, r'_{1:T})\}$$

where $r'_t = -r_j(s_t)$ ← return of state s_t "following" task $\mathcal{T}_i$?

b. Store $\mathcal{D}'_k$ in Replay Buffer

$$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$$

Requires

- Reward function form is known, evaluatable
- Dynamics consistent across goals / task
  - ↳ Relabeling is not that direct.
- Using off policy algorithm

Robot example

much better with HER

Image observations

we need a distance function d between

$$r'_t = -d(s_t, s_T)$$

current state

goal state

Binary reward: $\begin{cases} 1 & \text{is same} \\ 0 & \text{if not} \end{cases}$

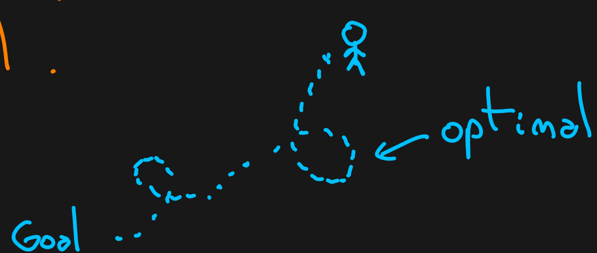
- Sparce
- Accurate

Random, unlabeled interaction

↳ optimal under the 0/1 reward of reaching last state.

- We don't care what happens before the terminal state, all trajectories are optimal if they reach the desired image-goal.

"we have an optimal sample for reaching the goal"



If data is optimal

↳ use supervised imitation learning.

Similar structure again:

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, s_T, r_{1:T})\}$ using some policy.

   ~~Store data in replay buffer~~

   $\cancel{\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k}$

2. Perform hindsight relabeling

   a. Relabel experience in $\mathcal{D}_k$ using last state s_T as goal

   $$\mathcal{D}'_k = \{(s_{1:T}, a_{1:T}, s_T, r'_{1:T})\} \leftarrow \text{relabeled data}$$

   where $r'_t = -d(s_t, s_T)$

   b. Store $\mathcal{D}'_k$ in Replay Buffer

   $$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$$

3. Update policy using <u>Supervised Imitation Learning</u> on Replay Buffer $\mathcal{D}$

Example of robot w/data from human: "Learn to reach image of a goal."

↳ Goal Image vs Current Image

↳ it works ˥'') ʃ

24/08/2020

Use insight to learn a better goal representation

1. Collect random, unlabeled iteraction data $\{(s_1, a_1, \dots, a_{t-1}, s_t)\}$
2. Train a latent state representation

$$s \to x$$

and latent state model

$$f(x' | x, a)$$

st. if we plan a sequence of actions wrt goal state s_t

we recover the observed action sequence.

"This correspond to embedding a planer in latent space into a goal condition policy"

3. Throw away latent space model

return goal representation x

("Distributional Planning Networks":

Accurate and shapped Reward Function

↳ not sparse as with $\mathbb{1}\{\text{same image}\}$