Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review

Sergey Levine

Presented by Yuchen Lu

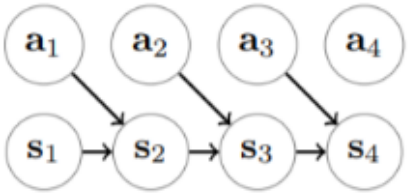
Table of contents

Table of contents

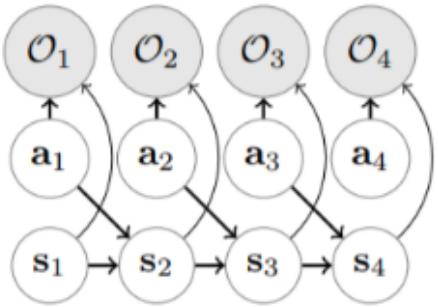
Transforming MDP to PGM

Construct a PGM from a MDP, such that doing inference in the resulting PGM is equivalent to find optimal policy in MDP.

Transforming MDP to PGM Cont.



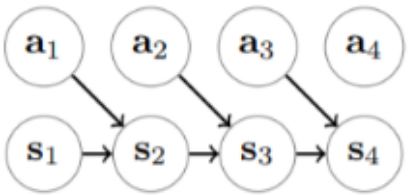
(a) graphical model with states and actions



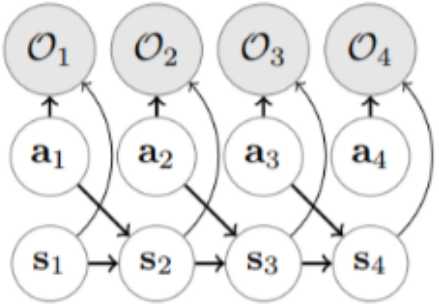
(b) graphical model with optimality variables

Figure 1: The graphical model for control as inference. We begin by laying out the states and actions, which form the backbone of the model (a). In order to embed a control problem into this model, we need to add nodes that depend on the reward (b). These "optimality variables" correspond to observations in a HMM-style framework: we condition on the optimality variables being true, and then infer the most probable action sequence or most probable action distributions.

Transforming MDP to PGM Cont.



(a) graphical model with states and actions



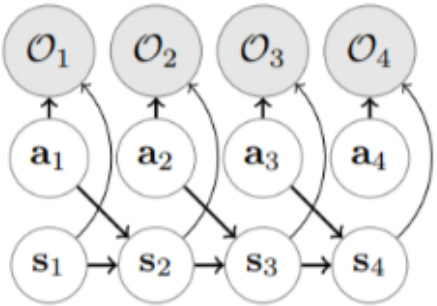
(b) graphical model with optimality variables

Figure 1: The graphical model for control as inference. We begin by laying out the states and actions, which form the backbone of the model (a). In order to embed a control problem into this model, we need to add nodes that depend on the reward (b). These "optimality variables" correspond to observations in a HMM-style framework: we condition on the optimality variables being true, and then infer the most probable action sequence or most probable action distributions.

Optimal control is equivalent to infer $p(a_t|s_t, O_{1:T})$

Transforming MDP to PGM Cont.

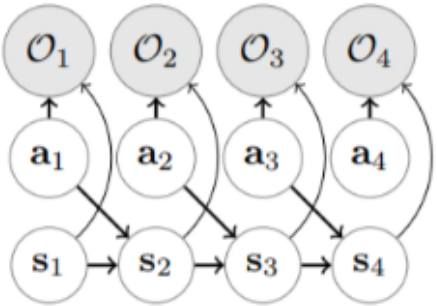
MDP	PGM
Optimal Control	Infer $p(a_t \mid s_t, O_{1:T})$
Model Dynamics	?
Reward	?



(b) graphical model with optimality variables

Transforming MDP to PGM Cont.

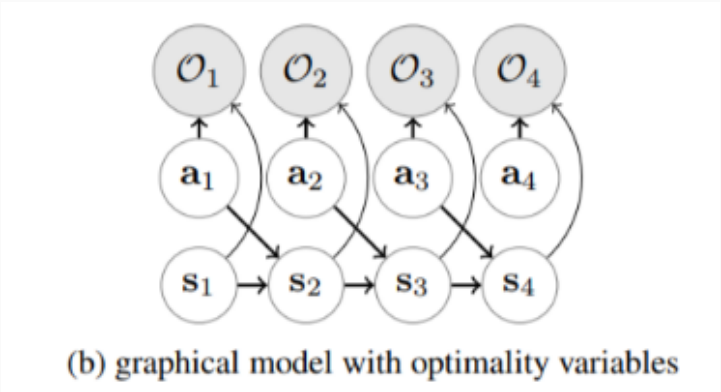
MDP	PGM
Optimal Control	Infer $p(a_t \mid s_t, O_{1:T})$
Model Dynamics	$p(s_{t+1} \mid s_t, a_t)$
Reward	?



(b) graphical model with optimality variables

Transforming MDP to PGM Cont.

MDP	PGM
Optimal Control	Infer $p(a_t \mid s_t, O_{1:T})$
Model Dynamics	$p(s_{t+1} \mid s_t, a_t)$
Reward	$p(O_t \mid s_t, a_t) = exp(r(s_t, a_t))$



(b) graphical model with optimality variables

Approximate Inference

The paper gives derivation of a message passing algorithm for exact inference for $p(a_t|s_t, O_{1:T})$, but it's intractable and risking-seeking (details in the paper).

Approximate Inference

The paper gives derivation of a message passing algorithm for exact inference for $p(a_t|s_t, O_{1:T})$, but it requires a known world dynamic and risking-seeking (details in the paper).
Instead use a parameterized distribution $\pi(a_t|s_t)$ to approximate the true $p(a_t|s_t, O_{1:T})$.

Approximate Inference by V.I.

The paper gives derivation of a message passing algorithm for exact inference for $p(a_t|s_t, O_{1:T})$, but it requires a known world dynamic and risking-seeking (details in the paper).
Instead use a parameterized distribution $\pi(a_t|s_t)$ to approximate the true $p(a_t|s_t, O_{1:T})$.
Finding π by minimizing the KL divergence with the true posterior, which is equivalent of maximizing **the ELBO of** $\log p(O_{1:T})$ **(classic variational inference result)**.

Approximate Inference by V.I.

$$
\begin{aligned}
\log p(O_{1:T}) &= \log \int p(O_{1:T}, s_{1:T}, a_{1:T}) ds_{1:T} da_{1:T} \\
&= \log \int p(O_{1:T}, s_{1:T}, a_{1:T}) \frac{q(s_{1:T}, a_{1:T})}{q(s_{1:T}, a_{1:T})} ds_{1:T} da_{1:T} \\
&= \log \underset{s_{1:T}, a_{1:T} \sim q}{E} \left[\frac{p(O_{1:T}, s_{1:T}, a_{1:T})}{q(s_{1:T}, a_{1:T})} \right] \\
&\geq \underset{s_{1:T}, a_{1:T} \sim q}{E} [\log p(O_{1:T}, s_{1:T}, a_{1:T}) - \log q(s_{1:T}, a_{1:T})]
\end{aligned}
$$

Approximate Inference by V.I.

Our ELBO:

$$\underset{s_{1:T}, a_{1:T} \sim q}{E} [\log p(O_{1:T}, s_{1:T}, a_{1:T}) - \log q(s_{1:T}, a_{1:T})]$$

Accorinding to our PGM,

$$p(O_{1:T}, s_{1:T}, a_{1:T}) = p(s_1) \Pi_{t=1}^{T-1} p(s_{t+1}|s_t, a_t) p(a_t) \exp(\sum_{t=1}^{T} r(s_t, a_t))$$

Also choose q to be of the form so that we can easily sample,

$$q(s_{1:T}, a_{1:T}) = p(s_1) \Pi_{t=1}^{T-1} p(s_{t+1}|s_t, a_t) \pi(a_t|s_t)$$

Approximate Inference by V.I.

Our ELBO becomes:

$$\underset{s_{1:T}, a_{1:T} \sim q}{E} \left[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t | s_t)) \right]$$

If we zoom in per time step, then maximizing this ELBO is equivalent to maximizing

$$\underset{s_t \sim q(s_t), a_t \sim \pi(a_t | s_t)}{E} \left[\sum_{t'=t}^{T} (r(s'_t, a'_t) \right] + \underset{s_t \sim q(s_t)}{E} [H(\pi(\cdot | s_t))]$$

which is the objective of **maximum entropy RL**

Table of contents

Maximizing ELBO Recursively (DP)

ELBO looks like:

$$\underset{s_{1:T}, a_{1:T} \sim q(s_{1:T}, a_{1:T})}{E} \left[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t | s_t)) \right]$$

$$= \sum_{t=1}^{T} \underset{s_t, a_t \sim q(s_t, a_t)}{E} [r(s_t, a_t) - \log \pi(a_t | s_t)]$$

Maximizing ELBO via DP

ELBO looks like:

$$\underset{s_{1:T}, a_{1:T} \sim q(s_{1:T}, a_{1:T})}{E} \left[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t | s_t)) \right]$$

$$= \sum_{t=1}^{T} \underset{s_t, a_t \sim q(s_t, a_t)}{E} [r(s_t, a_t) - \log \pi(a_t | s_t)]$$

- Let $a_t = \underset{s_t, a_t \sim q(s_t, a_t)}{E} [r(s_t, a_t) - \log \pi(a_t | s_t)]$. Let $b_t = \sum_{t'=t}^{T} a_{t'}$. We want to maximize b_1.

Maximizing ELBO via DP

ELBO looks like:

$$\underset{s_{1:T}, a_{1:T} \sim q(s_{1:T}, a_{1:T})}{E} \left[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t | s_t)) \right]$$

$$= \sum_{t=1}^{T} \underset{s_t, a_t \sim q(s_t, a_t)}{E} [r(s_t, a_t) - \log \pi(a_t | s_t)]$$

- Let $A_t = \underset{s_t, a_t \sim q(s_t, a_t)}{E} [r(s_t, a_t) - \log \pi(a_t | s_t)]$. Let $B_t = \sum_{t'=t}^{T} A_{t'}$. We want to maximize b_1.
- Maximize B_T first (base case). Then help us solve larger problem $B_{T-1}, b_{T-2}, \ldots$ until b_1.

DP: Base Case. B_T

Some derivation shows

$$
\begin{aligned}
B_T &= \underset{s_T, a_T \sim q(s_T, a_T)}{E} [r(s_t, a_t) - \log \pi(a_t|s_t)] \\
&= \underset{s_T \sim q(s_T)}{E} \left[-KL\left(\pi(a_T|s_T) || \frac{\exp(r(s_T, a_T))}{\exp(V(s_T))} \right) + V(s_T) \right]
\end{aligned}
$$

where $V(s_T) = \log \int \exp(r(s_T, a_T)) da_T$

DP: Base Case. B_T

Some derivation shows

$$\begin{aligned}\max B_T &= \max \underset{s_T, a_T \sim q(s_T, a_T)}{E} [r(s_T, a_T) - \log \pi(a_T | s_T)] \\ &= \max \underset{s_T \sim q(s_T)}{E} \left[-KL \left(\pi(a_T | s_T) || \frac{\exp(r(s_T, a_T))}{\exp(V(s_T))} \right) + V(s_T) \right]\end{aligned}$$

where $V(s_T) = \log \int \exp(r(s_T, a_T)) da_T$.
Maximize B_T by choosing $\pi(\cdots | s_T)$ such that $KL = 0$, the result is

$$\max B_T = \underset{s_T \sim q(s_T)}{E} [V(s_T)]$$

DP: One step further. B_{T-1}

$$\max B_{T-1} =$$

$$\max \underset{s_{T-1},a_{T-1}\sim q(s_{T-1},a_{T-1})}{E} \left[A_{T-1} + \max \underset{s_T\sim p(s_T|s_{T-1},a_{T-1}),a_T\sim\pi(a_T|s_T)}{E} [A_T] \right]$$

$$= \max \underset{s_{T-1},a_{T-1}\sim q(s_{T-1},a_{T-1})}{E} [r(s_{T-1},a_{T-1}) - \log \pi(a_{T-1}|s_{T-1})] + \underset{s_{T-1},a_{T-1}\sim q(s_{T-1},a_{T-1}),s_T\sim p(s_T|s_{T-1},a_{T-1})}{E} [V(s_T)]$$

DP: One step further. B_{T-1}

$$\max B_{T-1} = \max \underset{s_{T-1}, a_{T-1} \sim q(s_{T-1}, a_{T-1})}{E} [r(s_{T-1}, a_{T-1}) - \log \pi(a_{T-1}|s_{T-1})] + \underset{s_{T-1}, a_{T-1} \sim q(s_{T-1}, a_{T-1}), s_T \sim p(s_T|s_{T-1}, a_{T-1})}{E} [V(s_T)]$$

Let $Q(s_{T-1}, a_{T-1}) = r(s_{T-1}, a_{T-1}) + \underset{s_T \sim p(s_T|s_{T-1}, a_{T-1})}{E} [V(s_T)]$, we further have

$$max \underset{s_{T-1} \sim q(s_{T-1})}{E} \left[-KL \left(\pi(a_{T-1}|s_{T-1}) || \frac{\exp Q(s_{T-1}, a_{T-1})}{\exp V(s_{T-1})} \right) + V(s_{T-1}) \right]$$

where $V(s_{T-1}) = \log \int \exp Q(s_{T-1}, a) da$ is the log normalization constant. Then the maximization is similar to base case.

DP: General Case

In general,

$$\max B_t = max \underset{s_t \sim q(s_t)}{E} \left[-KL \left(\pi(a_t|s_t) || \frac{\exp Q(s_t, a_t)}{\exp V(s_t)} \right) + V(s_t) \right]$$

where

$$Q(s_t, a_t) = \begin{cases} r(s_t, a_t) + \underset{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}{E} [V(s_{t+1})] & t < T \\ r(s_t, a_t) & t = T \end{cases}$$

$$V(s_t) = \log \int \exp Q(s_t, a) da$$

Another Bellman Equation

Standard (Hard) Bellman equation:

$$Q(s_t, a_t) = r(s_t, a_t) + \underset{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}{E} \left[\max_a Q(s_{t+1}, a) \right]$$

Another Bellman Equation

Standard Bellman equation (Hard max op):

$$Q(s_t, a_t) = r(s_t, a_t) + \underset{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}{E} \left[\max_a Q(s_{t+1}, a) \right]$$

Soft Bellman equation (Soft max op):

$$Q(s_t, a_t) = r(s_t, a_t) + \underset{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}{E} \left[\log \int \exp Q(s_{t+1}, a) da \right]$$

Soft Q-Learning

Define loss for parameterized $Q_\phi(s, a)$ to be

$$L = \underset{s_t, a_t, r_t}{E}\left[(Q_\phi(s, a) - r_t - \log \int \exp Q_{\phi'}(s_t, a) da)^2\right]$$

where ϕ' is a delayed version. Then use same algorithm as deep-Q Learning.

Soft Q-Learning

Define loss for parameterized $Q_\phi(s, a)$ to be

$$L = \underset{s_t, a_t, r_t}{E}\left[(Q_\phi(s, a) - r_t - \log \int \exp Q_{\phi'}(s_t, a) da)^2\right]$$

where ϕ' is a delayed version. Then use same algorithm as deep-Q Learning.

- In the discrete space, the algorithm is trivial to implement.
- For the continuous case, both approximating integral and sampling from implicit policy is not trivial.
- More details in paper **Reinforcement Learning with Deep Energy-Based Policies**

Table of contents

Intuition: Hierarchical RL as Hierarchical PGM

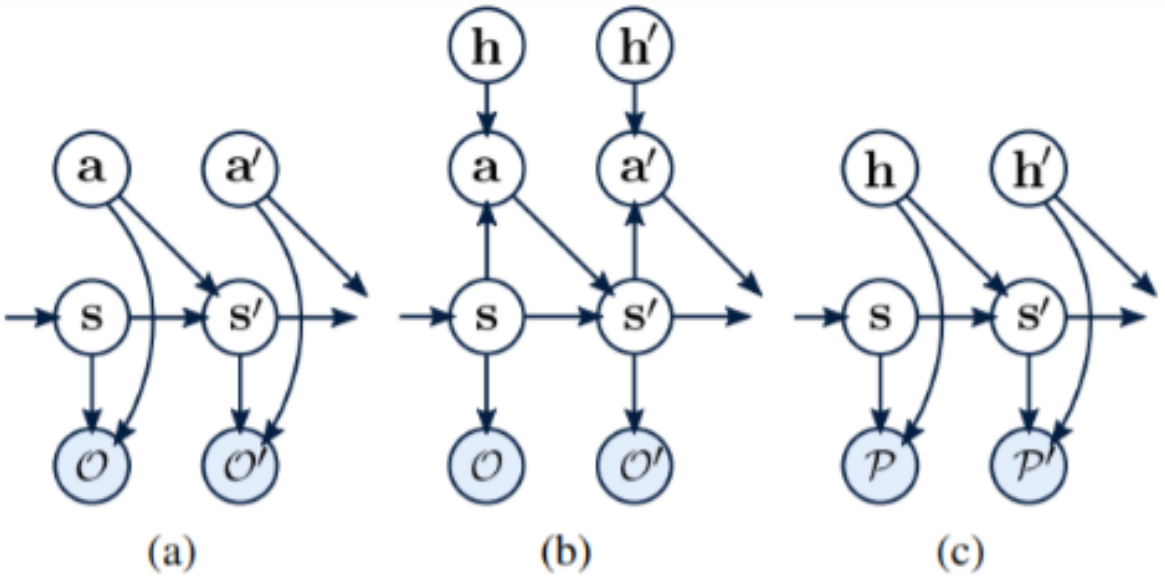


Figure 1. (a) The optimal control problem can be cast as an inference problem by considering a graphical model that consist of transition probabilities, action priors, and optimality variables. We can infer the optimal actions by conditioning on the optimality variables. (b) We can then train a latent variable policy to approximate the optimal actions, augment the graphical model with the policy's action distribution, and condition on a new set of optimality variables $\mathcal{P}_{0:T}$. (c) By marginalizing out the actions $\mathbf{a}_t$, we are left with a new model that is structurally identical to the one in (a), where $\mathbf{h}_t$ has taken the role of the original actions.

Pseudo Code

Algorithm 1 Latent Space Policy Learning

Input: True environment $p(\mathbf{s}_{t+1}|, \mathbf{s}_t, \mathbf{h}_t^{(0)})$, where $\mathbf{h}_t^{(0)}$ corresponds to the physical actions $\mathbf{a}_t$.

Input: Reward specifications $\{\mathcal{R}_0, \mathcal{R}_1, ..., \mathcal{R}_{K-1}\}$.

for $i = 0$ to $K - 1$ **do**

Initialize the weights of layer f_i.

Learn the weights of f_i so that $\mathbf{h}_t^{(i)} = f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t)$, where $\mathbf{h}_t^{(i+1)} \sim p(\mathbf{h}_t^{(i+1)})$, optimizes $\mathcal{R}_i$ on $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{h}_t^{(i)})$.

Embed the new layer f_i into the environment: $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{h}_t^{(i+1)}) \leftarrow p(\mathbf{s}_{t+1}|\mathbf{s}_t, f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t))$.

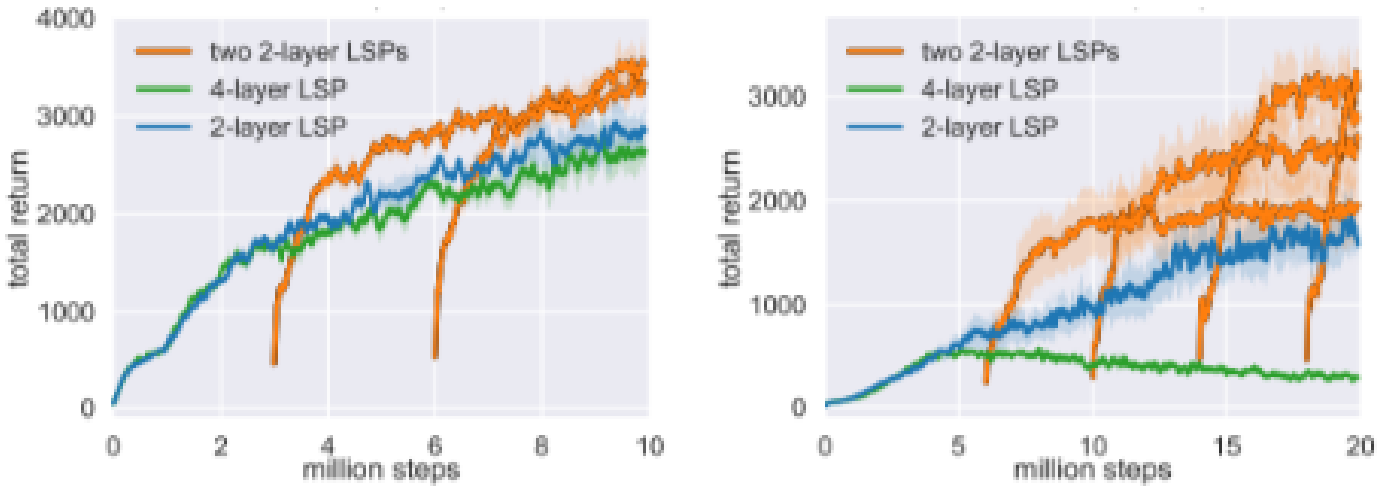
end for

Output: A hierarchical policy $f = f_0 \circ f_1 \circ ... \circ f_{K-1}$.

Some Comments

- The algorithm use deterministic sub-policy at f_k to make marginalization tractable, since f_k is a NN, it might still be expressive enough.
- Since f_k is deterministic, it won't introduce extra stochasticity when embedding f_k to create a new environment.
- The algorithm doesn't cripple lower level policy. At each level k, the policy is still trained to maximize the reward function R_k.

Partial Results



(a) Ant (rllab) (b) Humanoid (rllab)

Figure 4. (a, b) We trained two-level policies for the most challenging benchmark tasks, Ant and Humanoid, by first training a single level policy (blue) and then freezing it and adding a second policy level. We repeated this procedure by starting training of the top-level policy at multiple points in time (orange). We also trained a single policy with four invertible layers end-to-end (green) for comparison. In each case, stagewise training of two levels yields the best performance.

Final Remark

The control as inference theoretic framework is useful to think about new algorithm or justified (some of) existing RL algorithms.

Questions?

Thanks!