Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review

Sergey Levine

Presented by: Achint Kumar

Duke University

June 27, 2023

Desiderata

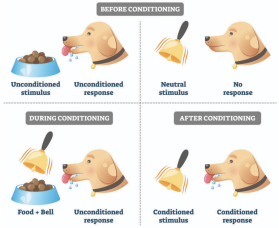
Desiderata

1. Introduction
   - Classical Conditioning
   - Operant Conditioning
   - Reinforcement Learning
2. Maximum Entropy Reinforcement Learning
   - Motivation
   - Probabilistic Inference
   - Variational Inference
3. Some Generalized Algorithms
   - Soft Q-Learning
   - Entropy Regularized Policy Gradient
   - Soft actor-critic Algorithm

Introduction

Classical Conditioning (Pavlov)

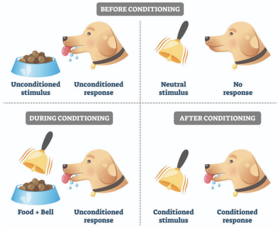
- Reward associated with stimuli (or state), $r(s_t)$
- Motivates TD learning



Introduction

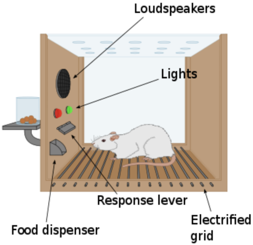
Classical Conditioning (Pavlov)

- Reward associated with stimuli (or state), $r(s_t)$
- Motivates TD learning



Operant Conditioning (Thorndike, Skinner)

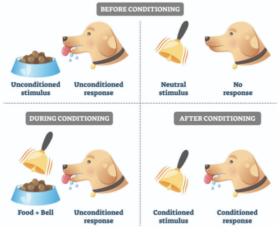
- Reward associated with actions, $r(a_t)$
- Motivates Policy Gradient for multi-arm bandits



Introduction

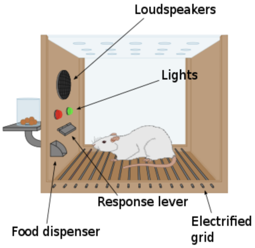
Classical Conditioning (Pavlov)

- Reward associated with stimuli (or state), $r(s_t)$
- Motivates TD learning



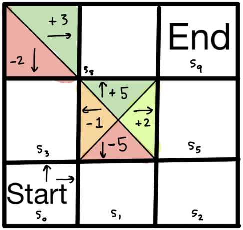
Operant Conditioning (Thorndike, Skinner)

- Reward associated with actions, $r(a_t)$
- Motivates Policy Gradient for multi-arm bandits



Reinforcement Learning

- Reward associated with both stimuli and actions, $r(s_t, a_t)$
- Motivates Q-learning, actor-critic learning



Reward Prediction Error Hypothesis

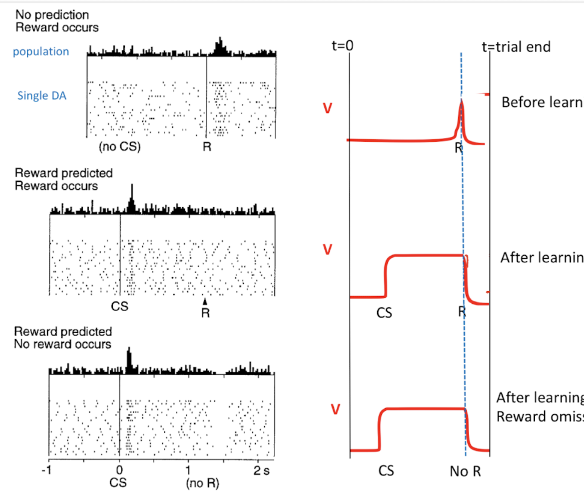
- Dopamine neurons in VTA were recorded in classical conditioning experiment (Schultz, et.al. 1997)
- Define *value function*, $V(s_t)$ which measures predicted reward
- Dopamine response can be modelled as,

$$\delta(t) = r(s_t) + \frac{dV}{dt} = r(s_t) + V(s_{t+1}) - V(s_t)$$

$\delta(t)$ is RPE

- Value function can be learnt by Temporal Difference(TD) learning algorithm. Update rule:

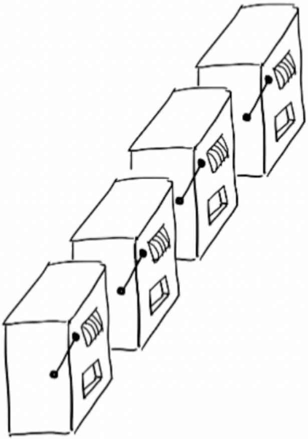
$$V(s_t) \leftarrow V(s_t) + \alpha\delta(t)$$



Mult-arm bandit problem

- Each bandit(slot machine) has a reward probability distribution. Find a *policy* $\pi(a)$ that maximizes total reward:

$$\max_{\pi} \sum_{t=0}^{T} \mathbb{E}_{\pi}\left[r(a_t)\right]$$



Mult-arm bandit (slot machines)

Policy Gradient algorithm

- Parameterize policy with θ as $\pi_\theta(a_t)$. For bandit problem it could be softmax function,

$$\pi_\theta(a_t) = \frac{e^{\theta_{a_t}}}{\sum_b e^{\theta_b}}$$

- Total average return is,

$$J(\theta) = \sum_{t=0}^{T} \mathbb{E}_\pi \left[r(a_t) \right]$$

- Perform gradient ascent on θ,

$$\begin{aligned} \theta &\leftarrow \theta + \alpha \nabla J(\theta) \\ &= \theta + \alpha \sum_{t=1}^{T} \sum_{a_t} \left[r(a_t) \nabla \pi_\theta(a_t) \right] \\ &= \theta + \alpha \sum_{t=1}^{T} \sum_{a_t} \left[(r(a_t) - b_t) \nabla \pi_\theta(a_t) \right], \text{ including baseline} \end{aligned}$$

Reinforcement Learning

Value function	$V(s) \to Q(s,a), A(s,a)$
Reward function	$r(a), r(s) \to r(a,s)$
Policy function	$\pi(a) \to \pi(a\|s)$

Advantage function is defined as,

$$A(s,a) = Q(s,a) - V(s)$$

The elements are closely related to reward $r(s,a)$

Classical Conditioning to Reinforcement Learning

We saw for classical conditioning,

$$V(s_t) \leftarrow V(s_t) + \alpha[r(s_t) + V(s_{t+1}) - V(s_t)]$$

For reinforcement learning replace $V(s_t) \rightarrow Q(s_t, a_t)$.
Algorithm:

1. Initialize Q(s,a) randomly. Q(FINAL,.)=0
2. Use $\epsilon-$greedy to determine policy $\pi(a|s)$
3. Go from state-action s_t, a_t to s_{t+1} using policy, $\pi(a|s_t)$.
4. Update action-value function using *on-policy* learning (SARSA algorithm),

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

   where a_{t+1} derived from policy $\pi(a|s_{t+1})$.
   Alternatively, update action-value function using *off-policy* learning (Q-learning algorithm)

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

5. Repeat 1-4 till s_{t+1} is final state.
6. Repeat 5 until Q function stabilizes.

Classical Conditioning to Reinforcement Learning

We saw for classical conditioning,

$$V(s_t) \leftarrow V(s_t) + \alpha[r(s_t) + V(s_{t+1}) - V(s_t)]$$

For reinforcement learning replace $V(s_t) \rightarrow Q(s_t, a_t)$.
Algorithm:

1. Initialize Q(s,a) randomly. Q(FINAL,.)=0
2. Use $\epsilon-$greedy to determine policy $\pi(a|s)$
3. Go from state-action s_t, a_t to s_{t+1} using policy, $\pi(a|s_t)$.
4. Update action-value function using *on-policy* learning (SARSA algorithm),

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

   where a_{t+1} derived from policy $\pi(a|s_{t+1})$.
   Alternatively, update action-value function using *off-policy* learning (Q-learning algorithm)

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

5. Repeat 1-4 till s_{t+1} is final state.
6. Repeat 5 until Q function stabilizes.

Classical Conditioning to Reinforcement Learning

We saw for classical conditioning,

$$V(s_t) \leftarrow V(s_t) + \alpha[r(s_t) + V(s_{t+1}) - V(s_t)]$$

For reinforcement learning replace $V(s_t) \rightarrow Q(s_t, a_t)$.
Algorithm:

1. Initialize Q(s,a) randomly. Q(FINAL,.)=0
2. Use $\epsilon-$greedy to determine policy $\pi(a|s)$
3. Go from state-action s_t, a_t to s_{t+1} using policy, $\pi(a|s_t)$.
4. Update action-value function using *on-policy* learning (SARSA algorithm),

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

   where a_{t+1} derived from policy $\pi(a|s_{t+1})$.
   Alternatively, update action-value function using *off-policy* learning (Q-learning algorithm)

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

5. Repeat 1-4 till s_{t+1} is final state.
6. Repeat 5 until Q function stabilizes.

Q Learning

Q function update rule is given by,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

we will see generalization of this rule (soft Q-learning) later.

Operant Conditioning to Reinforcement Learning

Earlier we saw, policy gradient algorithm.

1. Parameterize policy, $\pi_\theta(a|s)$ (more general than before)
2. Optimize average expected reward, $J(\theta)$ by,

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

In actor-critic learning, we parameterize both policy and value(or Q or A) function. It combines policy gradient with TD learning.

1. Parameterize policy, $\pi_\theta(a|s)$ with θ and value, $V_w(s)$ with w.
2. In state s, take action a and observe s' and $r(s,a)$. Update θ and w by,

$$w \leftarrow w + \alpha_w \delta \nabla V_w(s)$$
$$\theta \leftarrow \theta + \alpha_\theta \delta \nabla \log \pi_\theta(a|s)$$

where $\delta = r(s,a) + V(s') - V(s)$

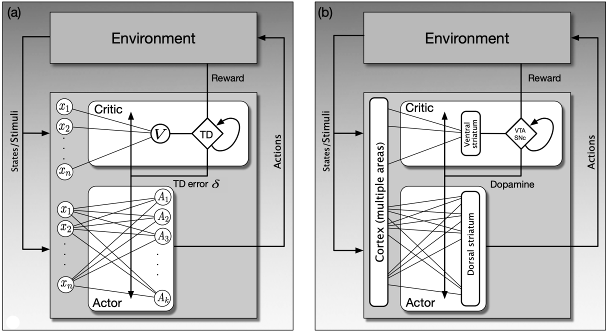
Actor-Critic Model

- Actor: Dorsal Striatum
- Critic: Ventral Striatum. Sends TD error to actor,

$$\text{Good action}, \delta > 0$$

$$\text{Bad action}, \delta < 0$$

- TD Error: VTA



Desiderata

Introduction

1. Regular formulation:

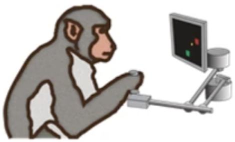
$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} r_t\right]$$

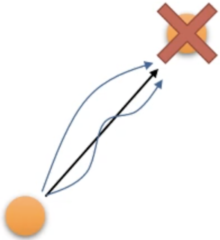
2. Maximum entropy formulation

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} r_t + \beta\mathcal{H}(\pi(a_t|s_t))\right]$$

Motivation-1

- Stochastic behaviour is more robust in constantly changing environments
- Ability to model suboptimal behaviour is useful for inverse RL (determining reward function from behaviour)



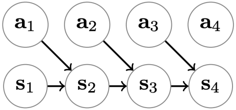


Motivation-2

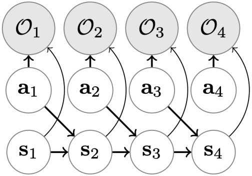
- We assume that there are observable binary *optimality* variables $\mathcal{O}_t$ where, $\mathcal{O}_t = 1$ denotes time step t is *optimal* and $\mathcal{O}_t = 0$ denotes that it is not optimal. We define,

$$p(\mathcal{O}_t = 1 | s_t, a_t) = \exp(r(s_t, a_t))$$

Note, all rewards must be negative for normalizability. There is no loss of generality.



(a) graphical model with states and actions



(b) graphical model with optimality variables

Applying Bayes Rule

Let $\tau = \{s_{1:T}, a_{1:T}\}$. By Bayes rule,

$$p(\tau|\mathcal{O}_{1:T} = 1) = \frac{p(\tau)p(\mathcal{O}_{1:T} = 1|\tau)}{p(\mathcal{O}_{1:T} = 1)}$$
$$\propto p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\exp(r(s_t, a_t))$$
$$= \left[p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\right]\exp(\sum_{t=1}^{T} r(s_t, a_t))$$

- Most probable trajectory is one with highest reward. But suboptimal trajectories are also possible with exponentially decreasing probability.
- Explains stochastic monkey behaviour.

Applying Bayes Rule

Let $\tau = \{s_{1:T}, a_{1:T}\}$. By Bayes rule,

$$
\begin{aligned}
p(\tau|\mathcal{O}_{1:T} = 1) &= \frac{p(\tau)p(\mathcal{O}_{1:T} = 1|\tau)}{p(\mathcal{O}_{1:T} = 1)} \\
&\propto p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t) \exp(r(s_t, a_t)) \\
&= \left[p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t) \right] \exp(\sum_{t=1}^{T} r(s_t, a_t))
\end{aligned}
$$

- Most probable trajectory is one with highest reward. But suboptimal trajectories are also possible with exponentially decreasing probability.
- Explains stochastic monkey behaviour.

Policy Search as Probabilistic Inference

Goal is to find optimal policy $\pi(a_t|s_t, \mathcal{O}_{t:T})$. This will be done by computing backward messages. We will need,

- State-action backward message: $\beta_t(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t)$. It is probability of optimality from time t to T given that it begins at (s_t, a_t).
- State backward message: $\beta_t(s_t) = p(\mathcal{O}_{t:T}|s_t)$. It is probability of optimality from time t to T given that it begins at s_t.

$$\begin{aligned}\beta_t(s_t) = p(\mathcal{O}_{t:T}|s_t) &= \int p(\mathcal{O}_{t:T}|s_t, a_t)p(a_t|s_t)\, da_t \\ &= \mathbb{E}_{a_t \sim p(a_t|s_t)}[\beta_t(s_t, a_t)]\end{aligned}$$

Action prior, $p(a_t|s_t)$ is assumed to be uniform without loss of generality.

Message Passing Algorithm for backward message-1

The recursive message passing algorithm for computing $\beta_t(s_t, a_t)$ proceeds from the last time step $t = T$ backward through time to $t = 1$. Base case, at $t = T$,

$$\beta_t(s_T, a_T) = p(\mathcal{O}_T | s_T, a_T) = \exp(r(s_T, a_T))$$

Recursive case is given as following,

$$\begin{aligned}
\beta_t(s_t, a_t) = p(\mathcal{O}_{1:t} | s_t, a_t) &= \int p(\mathcal{O}_{t:T}, s_{t+1} | s_t, a_t) ds_{t+1} \\
&= p(\mathcal{O}_t | s_t, a_t) \int p(\mathcal{O}_{t+1:T} | s_{t+1}) p(s_{t+1} | s_t, a_t) ds_{t+1} \\
&= p(\mathcal{O}_t | s_t, a_t) [\mathbb{E}_{s_{t+1} \sim p(s_{t+1} | s_t, a_t)} [\beta_{t+1}(s_{t+1})]
\end{aligned}$$

Message Passing Algorithm for backward message-2

1. Base case:

$$\beta_T(s_T, a_T) = p(\mathcal{O}_T|s_T, a_T) = \exp(r(s_T, a_T))$$
$$\beta_T(s_t T = \mathbb{E}_{a_T \sim p(a_T|s_T)}[\beta_T(s_T, a_T)]$$

2. Run loop from $t = T - 1$ to 1

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

Connecting to standard RL

Run loop from $t = T - 1$ to 1

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

Take logs of both equation. Define,

$$V(s_t) = \log \beta_t(s_t)$$
$$Q(s_t, a_t) = \log \beta_t(s_t, a_t)$$

First equation gives,

$$Q(s_t, a_t) = \log[p(\mathcal{O}_t|s_t, a_t)] + \log \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}[\exp[V(s_{t+1})]]$$
$$= r(s_t, a_t) + \max_{s_{t+1}} V(s_{t+1}) \text{ BAD!}$$

Second equation gives,

$$V(s_t) = \log \int \exp(Q(s_t, a_t))da_t \approx \max_{a_t} Q(s_t, a_t)$$

It is like value iteration algorithm for deterministic dynamics. Problem with stochastic dynamics.

Connecting to standard RL

Run loop from $t = T - 1$ to 1

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

Take logs of both equation. Define,

$$V(s_t) = \log \beta_t(s_t)$$
$$Q(s_t, a_t) = \log \beta_t(s_t, a_t)$$

First equation gives,

$$Q(s_t, a_t) = \log[p(\mathcal{O}_t|s_t, a_t)] + \log \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}[\exp[V(s_{t+1})]]$$
$$= r(s_t, a_t) + \max_{s_{t+1}} V(s_{t+1}) \text{ BAD!}$$

Second equation gives,

$$V(s_t) = \log \int \exp(Q(s_t, a_t))da_t \approx \max_{a_t} Q(s_t, a_t)$$

It is like value iteration algorithm for deterministic dynamics. Problem with stochastic dynamics.

Connecting to standard RL

Run loop from $t = T - 1$ to 1

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

Take logs of both equation. Define,

$$V(s_t) = \log \beta_t(s_t)$$
$$Q(s_t, a_t) = \log \beta_t(s_t, a_t)$$

First equation gives,

$$Q(s_t, a_t) = \log[p(\mathcal{O}_t | s_t, a_t)] + \log \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\exp[V(s_{t+1})]]$$
$$= r(s_t, a_t) + \max_{s_{t+1}} V(s_{t+1}) \text{ BAD!}$$

Second equation gives,

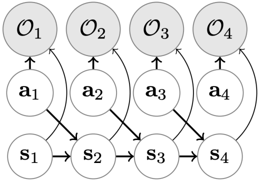
$$V(s_t) = \log \int \exp(Q(s_t, a_t)) da_t \approx \max_{a_t} Q(s_t, a_t)$$

It is like value iteration algorithm for deterministic dynamics. Problem with stochastic dynamics.

Computing Optimal Policy

$$
\begin{aligned}
p(a_t|s_t, \mathcal{O}_{1:T}) = \pi(a_t|s_t) &= p(a_t|s_t, \mathcal{O}_{t:T}) \\
&= \frac{p(a_t, s_t|\mathcal{O}_{t:T})}{p(s_t|\mathcal{O}_{t:T})} \\
&= \frac{p(\mathcal{O}_{t:T}|a_t, s_t)p(a_t, s_t)/p(\mathcal{O}_{t:T})}{p(\mathcal{O}_{t:T}|s_t)p(s_t)/p(\mathcal{O}_{t:T})} \\
&= \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)} = \exp(Q - V) = \exp(A(s_t, a_t))
\end{aligned}
$$

Actions with more advantage are exponentially more likely.



Problem with soft value iteration

Recall we had,

$$Q(s_t, a_t) \approx r(s_t, a_t) + \max_{s_{t+1}} V(s_{t+1})$$

$$V(s_t) \approx \max_{a_t} Q(s_t, a_t)$$

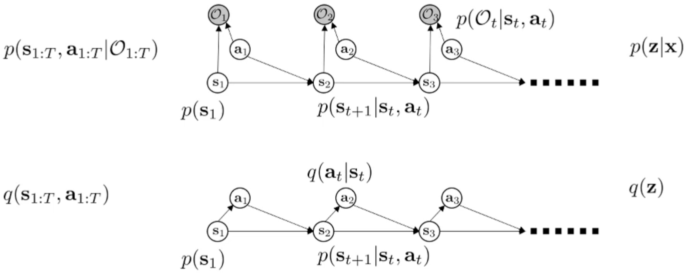
The problem stems from the fact that,

$$p(s_{t+1}|s_t, a_t, \mathcal{O}_{1:T}) \neq p(s_{t+1}|s_t, a_t)$$

We would like to find another distribution $q(s_{1:T}, a_{1:T})$ that is close $p(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T})$ but has the dynamics $p(s_{t+1}|s_t, a_t)$.

Structured Variational Inference-1

- Find another distribution $q(s_{1:T}, a_{1:T})$ that is close to $p(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T})$ but has the dynamics $p(s_{t+1}|s_t, a_t)$.
- Let $x = \mathcal{O}_{1:T}$ and $z = (s_{1:T}, a_{1:T})$. Find q(z) to approximate $p(z|x)$. This can be solved by Variational Inference.
- Let $q(s_{1:T}, a_{1:T}) = p(s_1) \prod_t p(s_{t+1}|s_t, a_t) q(a_t|s_t)$



Structured Variational Inference-2

Let $x = \mathcal{O}_{1:T}$ and $z = (s_{1:T}, a_{1:T})$. Variational lower bound is given by,

$$\log p(x) \geq \mathbb{E}_{z \sim q(z)}[\log p(x, z) - \log q(z)]$$

Substituting variables we get,

$$\begin{aligned}
\log p(\mathcal{O}_{1:T}) \geq & \mathbb{E}_{(s_{1:T}, a_{1:T}) \sim q}[\log p(s_1) + \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) + \sum_{t=1}^{T} \log p(\mathcal{O}_{1:T}|s_t, a_t)] \\
& - \log p(s_t) - \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) - \sum_{t=1}^{T} \log q(a_t|s_t)] \\
& = \mathbb{E}_{(s_{1:T}, a_{1:T}) \sim q}[\sum_{t=1}^{T} r(s_t, a_t) - \log q(a_t|s_t)] \\
& = \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim q}[r(s_t, a_t) + \mathcal{H}q(a_t|s_t)]
\end{aligned}$$

Structured Variational Inference-3

Optimizing Variational lower bounds leads to soft value iteration algorithm,

- for t=T-1 to 1:

$$
\begin{aligned}
Q(s, a) &\leftarrow r(s, a) + \mathbb{E}[V(s')] \\
V(s) &\leftarrow \text{softmax}_a(Q(s, a))
\end{aligned}
$$

Traditional value iteration has the form,

- for t=T-1 to 1:

$$
\begin{aligned}
Q(s, a) &\leftarrow r(s, a) + \mathbb{E}[V(s')] \\
V(s) &\leftarrow \max_a(Q(s, a))
\end{aligned}
$$

Desiderata

Soft Q-Learning

For standard Q-learning,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$
$$\pi(a_t|s_t) \leftarrow \epsilon\text{-greedy}[\text{argmax}_a Q(a, s_t)]$$

For soft Q-learning,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \text{softmax}_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$
$$\pi(a_t|s_t) \leftarrow \exp(A(s_t, a_t))$$

Policy Gradient

For standard Policy Gradient,

- Total average return is,

$$J(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\pi} \left[r(s_t, a_t) \right]$$

- Perform gradient ascent on θ,

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) = \theta + \alpha \sum_{t=1}^{T} \mathbb{E}_{a_t \sim \pi(a_t|s_t)} \left[(r(s_t, a_t) - b_t) \nabla \log \pi_\theta(a_t|s_t) \right]$$

For Entropy Regularized Policy Gradient,

- Total average return is,

$$J(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\pi} \left[r(s_t, a_t) + \mathcal{H}(q(a_t|s_t)) \right]$$

- Perform gradient ascent on θ,

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) = \theta + \alpha \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t) \sim q(s_t,a_t)} [\nabla_\theta \log q_\theta(a_t|s_t) A(s_t, a_t)]$$

Soft actor-critic Algorithm

- Critic: Update Q-function to evaluate current policy:

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{s' \sim p_s, a' \sim \pi}[Q(s',a') - \log \pi(a'|s')]$$

  This converges to Q^π.

- Actor: Update the policy with gradient of information projection:

$$\pi_{new} = \arg\min_{\pi'} D_{KL}\left(\pi'(.|s) || \frac{1}{\mathcal{Z}} \exp Q^{\pi_{old}}(s,.)\right)$$

  In practice, only take one gradient step on this objective