Reinforcement learning and control as probabilistic inference: tutorial and Review

Paper by Sergey Levine, 2018

October 21, 2018

UBC MLRG 2018

Goals

"In this article, we will discuss how a generalization of the reinforcement learning ... is equivalent to exact probabilistic inference in the case of deterministic dynamics, and variational inference in the case of stochastic dynamics."

Previous work

- Kalman duality, max entropy RL, KL divergence control, stochastic optimal control
- Big idea: learning via probabalistic graphical models (PGM)

tl;dr

- Bayesian version of Q-learning, policy gradient, etc is like changing max's to soft-max's.
- The devil is in the details. Different implementations can give you different variances.
- You get to control $p(a|s)$ but you do NOT get to control $p(s_{t+1}|s_t, a_t)$.

Outline

- Background and notation
- Graphical models
  - Policy search as probabilistic inference
  - Deterministic vs stochastic case
- Variational inference and stochastic dynamics
  - Maximum entropy reinforcement learning with fixed dynamics
  - Structured variational inference
- Approximate inference with function approximation
  - Maximum entropy policy gradients
  - Maximum entropy actor-critic algorithms
  - Soft Q-learning

Background and notation

Notation

- $s \in \mathcal{S}$ states, $a \in \mathcal{A}$ actions
- environment has Markov property

$$p(s_{t+1}|s_1, a_1, ..., s_t, a_t) = p(s_{t+1}|s_t, a_t)$$

  - Deterministic: s_{t+1} fixed given s_t, a_t.
  - Stochastic: use transition probability.

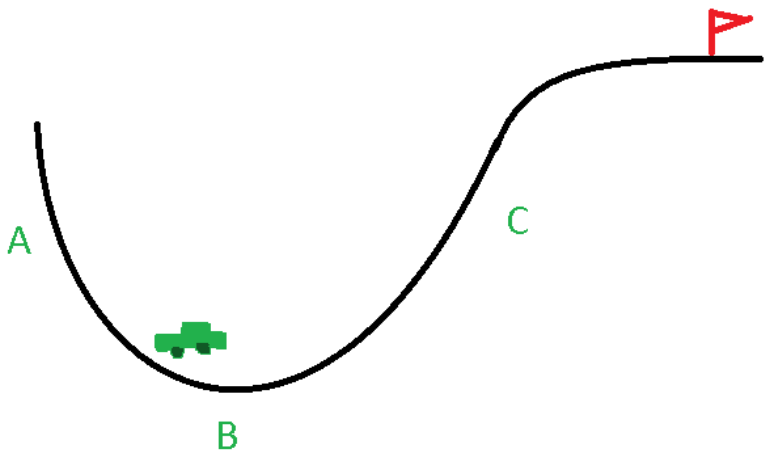
- Trajectory $\tau = \{s_t, a_t : t = 1, ..., T\}$
- Reward function: $r(s_t, a_t)$
- (Usual) Q function and value function

$$Q^\tau(s_t, a_t) = \sum_{t=1}^{T} \mathbb{E}_\tau[r(s_t, a_t)], \qquad V^\tau(s_t) = \max_{a_t \in \mathcal{A}} Q^\tau(s_t, a_t)$$

  (no discounts)

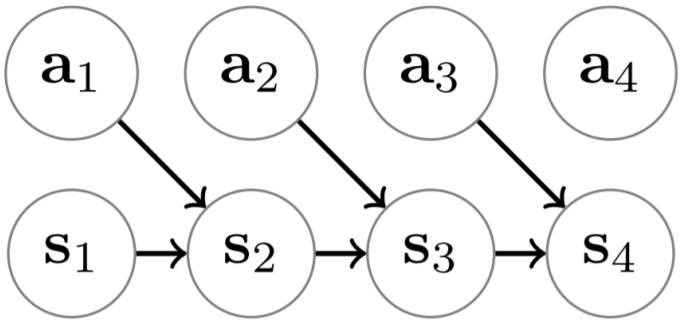
- Goal: Find the optimal policy $p(a_t|s_t, ??)$

Example: mountain car



- States: position, velocity
- Actions: Move forward with force f, backward with force $-f$
- Reward: If car reaches flag, $r(s_t, a_t) = 1$ and terminate. Else, $r(s_t, a_t) = 0$.

System model



(a) graphical model with states and actions

- Focus on discrete time
- Note no $p(a_t|s_t)$ (assume random / uniform).

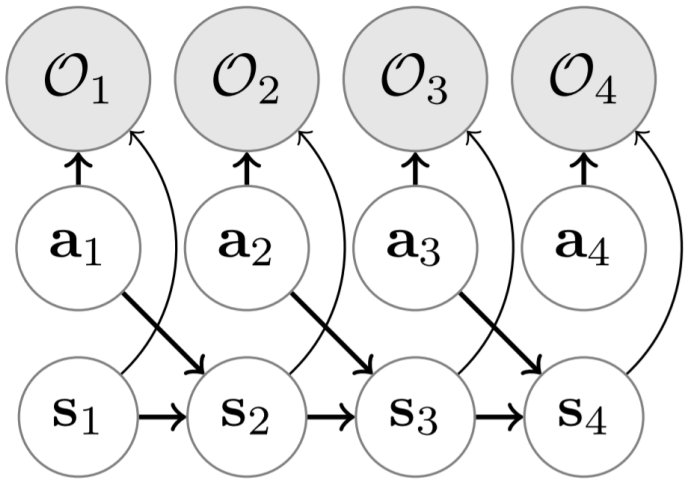
Policy search as probabilistic inference

Policy

- Optimality: $O_t = \{\text{event that at time } t, \text{ policy was optimal}\}$
- Model

$$p(O_t = 1 | s_t, a_T) = \exp(r(s_t, a_t))$$

- Optimal policy $\pi(a_t|s_t) = p(a_t|s_t, O_{t:T} = 1)$



(b) graphical model with optimality variables

Inference

Goal: learn this trajectory via inference (e.g. sun-product inference alg., HMMs, Viterbi, forward-backward algos)

- Forward message: Play a game, calculate

$$p(O_T|s_T, a_T) = \exp(r(s_T, a_T))$$

- Backward messages: from $t = T, ..., 1$

$$\underbrace{p(O_{t:T}|s_t)}_{\beta_t(s_t)} = \int_{\mathcal{A}} \underbrace{p(O_{t:T}|s_t, a_T)}_{\beta_t(s_t,a_t)} \underbrace{p(a_t|s_t)}_{1/|\mathcal{A}|} da_t$$

$$\underbrace{p(O_{t:T}|s_t, a_t)}_{\beta_t(s_t,a_t)} = \int_{\mathcal{S}} \underbrace{p(O_{t+1:T}|s_{t+1})}_{\beta_{t+1}(s_{t+1})} \underbrace{p(s_{t+1}|s_t, a_t)}_{\text{sample}} \underbrace{p(O_t|s_t, a_t)}_{\exp(r(s_t,a_t))} ds_{t+1}$$

Inference

Compute policy (no θ)

$$
\begin{aligned}
\pi(a_t|s_t) \quad &:= \quad p(a_t|s_t, O_{t:T}) = \frac{p(s_t, a_t|O_{t:T})}{p(s_t|O_{t:T})} \\
&\overset{\text{Bayes}}{=} \quad \frac{p(O_{t:T}|s_t, a_t)\overbrace{p(a_t|s_t)}^{1/|\mathcal{A}|}p(s_t)}{p(O_{t:T}|s_t)p(s_t)} \\
&\propto \quad \frac{p(O_{t:T}|s_t, a_t)}{p(O_{t:T}|s_t)} = \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)}
\end{aligned}
$$

Generalized Q-learning

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

Then

$$\begin{aligned} V(s_t) &= \log \int_{\mathcal{A}} \exp(Q(s_t, a_t)) da_t \quad \text{(softmax)} \\ &\approx \max_{a_t} Q(s_t, a_t) \end{aligned}$$

Generalized Q-learning (Deterministic)

$$Q(s_t, a_t) := \log(\beta_t(s_t, a_t)), \qquad V(s_t) := \log(\beta_t(s_t)) \approx \max_{a_t} Q(s_t, a_t)$$

- Recall

$$\begin{aligned} \beta_t(s_t, a_t) &= \int_{\mathcal{S}} \beta_{t+1}(s_{t+1}) p(s_{t+1}|s_t, a_t) \exp(r(s_t, a_t)) ds_{t+1} \\ &= \mathbb{E}_{s_{t+1}|s_t, a_t}[\beta_t(s_t) \exp(r(s_t, a_t))] \end{aligned}$$

- If dynamics are deterministic, expectation goes away

$$\beta_t(s_t, a_t) = \beta_{t+1}(s_{t+1}) \exp(r(s_t, a_t))$$

$$Q(s_t, a_t) = V(s_{t+1}) + r(s_t, a_t)$$

which is how Q-learning works

Generalized Q-learning (Deterministic)

$$Q(s_t, a_t) := \log(\beta_t(s_t, a_t)), \qquad V(s_t) := \log(\beta_t(s_t)) \approx \max_{a_t} Q(s_t, a_t)$$

$$\beta_t(s_t, a_t) = \mathbb{E}_{s_{t+1}|s_t,a_t}[\beta_{t+1}(s_{t+1}) \exp(r(s_t, a_t))]$$

- If dynamics are stochastic

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \log\left(\mathbb{E}_{s_{t+1}|s_t,a_t}[\exp(V(s_{t+1}))]\right) \\ &\approx r(s_t, a_t) + \max_{s_{t+1}|s_t,a_t} V(s_{t+1}) \end{aligned}$$

- "This creates risk seeking behavior: if an agent behaves according to this Q-function, it might take actions that have extremely high risk, so long as they have some non-zero probability of a high reward. Clearly, this behavior is not desirable in many cases, and the standard PGM described in this section is often not well suited to stochastic dynamics."

Inference $\rightarrow$ optimization

The best trajectory ever

$$
\begin{aligned}
p(\tau|O_{1:T}=1) &\propto p(\tau, O_{1:T}) \\
&= p(s_1)\prod_{t=1}^{T} p(O_t=1|s_t,a_t)p(s_{t+1}|s_t,a_t) \\
&= p(s_1)\prod_{t=1}^{T} \exp(r(s_t,a_t))p(s_{t+1}|s_t,a_t) \\
&= \underbrace{\left(p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t,a_t)\right)}_{\text{trajectory}} \underbrace{\left(\sum_{t=1}^{T} \exp(r(s_t,a_t))\right)}_{\text{reward for that trajectory}}
\end{aligned}
$$

Exact inference

The best trajectory ever (notation $p(\tau) = p(\tau|O_{1:T} = 1)$)

$$p(\tau) \quad \propto \quad \left(p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t) \right) \left(\sum_{t=1}^{T} \exp(r(s_t, a_t)) \right)$$

What we actually learned

$$\hat{p}(\tau) \quad \propto \quad \left(p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t, O_{1:T}) \right) \left(\prod_{t=1}^{T} \pi(a_t|s_t) \right)$$

where $\pi(a_t|s_t) = p(a_t|s_t, O_{1:T} = 1) \neq p(a_t|s_t)$ Exact inference

$$p(\tau) = \hat{p}(\tau) \iff D_{KL}(\hat{p}(\tau)||p(\tau)) = 0$$

Deterministic inference

$$
\begin{aligned}
D_{KL}(\hat{p}(\tau)||p(\tau)) &= -\mathbb{E}_{\tau\sim\hat{p}}[\log(p(\tau)) - \log(\hat{p}(\tau))] \\
&= -\mathbb{E}_{\tau\sim\hat{p}}[(\log(p(s_1) + \sum_{t=1}^{T}(\log(p(s_{t+1}|s_t,a_t) + r(s_t,a_t)) \\
&\quad -(\log(p(s_1) + \sum_{t=1}^{T}(\log(\overbrace{p(s_{t+1}|s_t,a_t,O_{1:T})}^{=p(s_{t+1}|s_t,a_t)}) + \log(\pi(a_t|s_t)))] \\
&= -\mathbb{E}_{\tau\sim\hat{p}}\left[\sum_{t=1}^{T} r(s_t,a_t) - \log(\pi(a_t|s_t))\right] \\
&= \sum_{t=1}^{T}(\mathbb{E}_{s_t,a_t\sim\hat{p}}[\log(\pi(a_t|s_t)) - r(s_t,a_t)]) \\
&= \sum_{t=1}^{T}\Bigg(-\underbrace{\mathbb{E}_{s_t\sim\hat{p}}[\mathcal{H}(\pi(a_t|s_t))]}_{\text{Expected conditional entropy}} - \underbrace{\mathbb{E}_{s_t,a_t\sim\hat{p}}[r(s_t,a_t)]}_{\text{expected reward}}\Bigg)
\end{aligned}
$$

Deterministic inference

$$-D_{KL}(\hat{p}(\tau)||p(\tau)) = \sum_{t=1}^{T} \underbrace{\mathbb{E}_{s_t \sim \hat{p}}[\mathcal{H}(\pi(a_t|s_t))]}_{\text{Expected conditional entropy}} + \underbrace{\mathbb{E}_{s_t, a_t \sim \hat{p}}[r(s_t, a_t)]}_{\text{expected reward}}$$

- " Therefore, minimizing the KL-divergence corresponds to maximizing the expected reward and the expected conditional entropy, in contrast to the standard control objective in Equation (1), which only maximizes reward. Hence, this type of control objective is sometimes referred to as *maximum entropy reinforcement learning* or *maximum entropy control*."
- " However, that in the case of stochastic dynamics, the solution is not quite so simple. ...[T]his objective is difficult to optimize in a model-free setting. ...[I]t also results in an optimistic policy that assumes a degree of control over the dynamics that is unrealistic in most control problems. "

Variational inference and stochastic dynamics

Maximum entropy reinforcement learning with fixed dynamics

- Previously, in stochastic settings, we implicitly optimized $p(s_{t+1}|s_t, a_t, O_{1:T})$, which is unrealistic
- That is, our policy always assumed state transitions would be optimal, so "risky is ok".
- tl;dr: Let's replace what we previously did with

$$p(s_{t+1}|s_t, a_t, O_{1:T}) \rightarrow p(s_{t+1}|s_t, a_t)$$

and just optimize.

Stochastic case: min. KL div. without optimism

Then we recover a familiar equation

$$-D_{KL}(\hat{p}(\tau)||p(\tau)) \quad = \quad \sum_{t=1}^{T} \mathbb{E}_{s_t,a_t\sim\hat{p}}[r(s_t,a_t) + \mathcal{H}(\pi(a_t|s_t))]$$

and

$$\mathbb{E}_{a_t|s_t\sim\hat{p}}[r(s_t,a_t) + \mathcal{H}(\pi(a_t|s_t))] + \mathbb{E}_{a_t|s_t\sim\hat{p}}\mathbb{E}_{s_{t+1}\sim\hat{p}}[V(s_{t+1})]$$

$$= \quad -D_{KL}\left(\pi(a_t|s_t)||\frac{\exp(Q(s_t,a_t))}{\exp(V(s_t))}\right) + V(s_t)$$

with

$$Q(s_t,a_t) := r(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim p}[V(s_{t+1})] \leftarrow \text{ not risky!}$$

(use whiteboard to fill in details)

- At iteration t, want to minimize

$$\begin{aligned} & \mathbb{E}_{a_t|s_t\sim\hat{p}}[r(s_t,a_t)+\mathcal{H}(\pi(a_t|s_t))]+\mathbb{E}_{a_t|s_t\sim\hat{p}}\mathbb{E}_{s_{t+1}\sim\hat{p}}[V(s_{t+1})] \\ = \quad & -D_{KL}\left(\pi(a_t|s_t)||\frac{\exp(Q(s_t,a_t))}{\exp(V(s_t))}\right)+V(s_t) \end{aligned}$$

  Therefore

$$\pi(a_t|s_t)=\begin{cases}\exp(r(s_T,a_T)-V(s_T)) & \text{if } t=T \\ \exp(Q(s_t,a_t)-V(s_t)) & \text{else.}\end{cases}$$

- Also, if we choose

$$V(s_t)=\log\int_{\mathcal{A}}\exp(Q(s_t,a_t))da_t,$$

  then we recover the standard Q-learning with max $\rightarrow$ softmax.

Evidence based lower bound (ELBO)

$$\log(p(O_{1:T})) \quad \geq \quad \mathbb{E}_{s_{1:T}, a_{1:T} \sim \hat{p}} \left[\sum_{t=1}^{T} r(s_t, a_t) - \log(\hat{p}(a_t | s_t)) \right]$$

Proof: whiteboard

" Intuitively, this means that this objective attempts to find the closest match to the maximum entropy trajectory distribution, subject to the constraint that the agent is only allowed to modify the policy, and not the dynamics"

Approximate inference with function approximation

Maximum entropy policy gradients

- Up until now, the "optimal" policy $\pi(a_t|s_t) = p(a_t|s_t, O_{1:T})$ is an inferred probability distribution
- Now we want to incorporate policy gradient methods, where $\pi(a_t|s_t) = \pi_\theta(a_t|s_t)$ and θ is a vector of parameters
- Define q distribution such that $q(s) = p(s)$, $q_\theta(a|s) = \pi_\theta(a|s)$.

Maximum entropy objective function

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{s_t, a_t \sim q}[r(s_t, a_t) - \mathcal{H}(q_\theta(a_t|s_t))]$$

Computing the policy gradient

$$\mathbb{E}_{s_t,a_t}[f_\theta(s_t,a_t)] = \int_{s_t,a_t} f_\theta(s_t,a_t) \underbrace{\prod_{t'=1}^{t} q_\theta(a_{t'}|s_{t'})p(s_{t'}|s_{t'-1},a_{t'-1})}_{p_\theta(s_t,a_t)} d(s_t,a_t)$$

REINFORCE trick

$$\nabla_\theta q_\theta(a_{t'}|s_{t'}) = q_\theta(a_{t'}|s_{t'})\nabla_\theta \log(q_\theta(a_{t'}|s_{t'}))$$

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{s_t,a_t\sim q}[r(s_t,a_t) - \mathcal{H}(q_\theta(a_t|s_t))]$$

$$\nabla_\theta J \overset{\text{magic}}{=} \sum_{t=1}^{T} \mathbb{E}\left[\nabla_\theta \log q_\theta(a_t|s_t)\left(\sum_{t'=t}^{T} r(s_{t'},a_{t'}) - \log q_\theta(a_{t'}|s_{t'}) - 1\right)\right]$$

$$\overset{\text{bias}}{=} \sum_{t=1}^{T} \mathbb{E}\left[\nabla_\theta \log q_\theta(a_t|s_t)\left(\sum_{t'=t}^{T} r(s_{t'},a_{t'}) - \log q_\theta(a_{t'}|s_{t'}) - b(s_{t'})\right)\right]$$

Policy gradient

$$\nabla_\theta J \quad = \quad \sum_{t=1}^{T} \mathbb{E}\left[\nabla_\theta \log q_\theta(a_t|s_t)\hat{A}(s_t, a_t)\right]$$

where

$$\hat{A}(s_t, a_t) = \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - \log q_\theta(a_{t'}|s_{t'}) - b(s_{t'})\right)$$

is the *advantage* at time t.

- Any standard advantage estimator, such as the GAE estimator (Schulman et al., 2016), can be used in place of the standard baselined Monte Carlo return above. Again, the only necessary modification is to add $logq_\theta(a_t|s_t)$ to the reward at each time step t'."
- "As with standard policy gradients, a practical implementation of this method estimates the expectation by sampling trajectories from the current policy, and may be improved by following the natural gradient direction."

Maximum entropy actor critic

- Actor = policy, critic = Value function
- Message passing

$$\begin{aligned}
&\mathbb{E}_{s_{t+1:T}, a_{t+1:T}|s_t, a_t}[\log p(O_{t:T}|s_{t:T}, a_{t:T})] \\
&= \log p(O_t|s_t, a_t) + \\
&\mathbb{E}_{s_{t+1}|s_t, a_t}\left[\underbrace{\mathbb{E}_{a_t|s_t}\left[\sum_{t'=t+1}^{T} \log p(O_{t'}|s_{t'}, a_{t'}) - \log q(a_{t'}|s_{t'})\right]}_{=:V(s_t)}\right]
\end{aligned}$$

then

$$V(s_t) = \mathbb{E}_{a_t|s_t}\left[\underbrace{\mathbb{E}_{s_{t+1}|s_t, a_t}[V(s_{t+1})] + \log p(O_t|s_t, a_t)}_{:=Q(s_t, a_t)} - \log q(a_t|s_t)\right]$$

- Optimal policy

$$q^*(a_t|s_t) = \frac{\exp(Q(s_t, a_t))}{\log \int_{\mathcal{A}} \exp Q(s_t, a_t) da_t}$$

Maximum entropy actor critic

- In general, $V(s_t)$ and $Q(s_t, a_t)$ are not optimal, but correspond to current policy.
- However, at convergence, when $q(a_t|s_t) = q^*(a_t|s_t)$, then

$$
\begin{aligned}
V(s_t) &= \mathbb{E}_{a_t|s_t}[Q(s_t, a_t) - \log q(a_t|s_t)] \\
&\overset{\text{greedy policy def}}{=} \mathbb{E}_{a_t|s_t}\left[Q(s_t, a_t) - Q(s_t, a_t) + \log \int_{\mathcal{A}} \exp Q(a_t, s_t) da_t\right] \\
&= \log \int_{\mathcal{A}} \exp Q(a_t, s_t) da_t \\
&\approx \max_{a_t \in \mathcal{A}} Q(a_t, s_t)
\end{aligned}
$$

Maximum entropy actor critic

- Greedy policy

$$\begin{aligned} q^*(a_t|s_t) &= \underset{q(a_t|s_t)}{\operatorname{argmax}} \mathbb{E}_{s_t}[V(s_t)] \\ &= \underset{q(a_t|s_t)}{\operatorname{argmax}} \underbrace{\mathbb{E}_{s_t}[\mathbb{E}_{a_t|s_t}[Q(s_t,a_t) - \log q(a_t|s_t)]]}_{=:J(\theta)} \end{aligned}$$

- Gradient of objective, via REINFORCE + bias

$$\nabla_\theta J(\theta) := \mathbb{E}_{s_t}\left[\mathbb{E}_{a_t|s_t}\left[\nabla \log q(a_t|s_t)(Q(s_t,a_t) - \log q(a_t|s_t) - b(s_t))\right]\right]$$

- Compare with gradient for policy-only update

$$\nabla_\theta J(\theta) := \mathbb{E}_{s_t}\left[\mathbb{E}_{a_t|s_t}\left[\nabla \log q(a_t|s_t)(r(s_t,a_t) - \log q(a_t|s_t) - b(s_t))\right]\right]$$

- " The modification lies in the use of the backward message $Q(s_t, a_t)$ in place of the Monte Carlo advantage estimate. The algorithm therefore corresponds to an actor-critic algorithm, which generally provides lower variance gradient estimates."

Function fitting action-critic

- $V(s_t) \rightarrow V_\psi(s_t)$, $Q(s_t, a_t) \rightarrow Q_\phi(s_t, a_t)$ and ψ, ϕ are parameters, e.g. neural network
- We try to minimize two objectives

$$\mathcal{E}(\phi) = \mathbb{E}_{s_t, a_t}\left[(r(s_t, a_t) + \mathbb{E}_{s_{t+1}|s_t, a_t}[V_\psi(s_{t+1})] - Q_\phi(s_t, a_t))^2\right]$$

$$\mathcal{E}(\psi) = \mathbb{E}_{s_t}\left[(\mathbb{E}_{a_t|s_t}[Q_\phi(s_t, a_t) - \log q(a_t|s_t)] - V_\psi(s_t, a_t))^2\right]$$

- " It may be beneficial to keep track of both $V(s_t)$ and $Q(s_t, a_t)$ networks. This is perfectly reasonable in a message passing framework, and in practice might have many of the same benefits as the use of a target network, where the updates to Q and V can be staggered or damped for stability."
- "Policy iteration or actor-critic methods might be preferred (over, for example, direct Q-learning), since they explicitly handle both approximate messages and approximate factors in the structured variational approximation. "

Soft Q-learning

- We can also just drop the neural networks for value $V(s_t)$ and policy $q(a_t|s_t)$

$$V(s_t) = \log \int_{\mathcal{A}} \exp(Q(s_t, a_t))da_t, \qquad q(a_t|s_t) = \exp(Q(s_t, a_t) - V(s_t))$$

- Now only one neural network: $Q_\phi(s_t, a_t)$
- Only one objective function to minimize

$$\mathcal{E}(\phi) = \frac{1}{2} \underbrace{\mathbb{E}_{s_t, a_t \sim q}}_{\text{not function of } \phi} \left[(r(s_t, a_t) + \mathbb{E}_{s_{t+1}|s_t, a_t}[\underbrace{V(s_{t+1})}_{\text{not function of } \phi}] - Q_\phi(s_t, a_t))^2 \right]$$

and writing $Q^t := Q_\phi(s_t, a_t)$,

$$\nabla_\phi \mathcal{E} = \mathbb{E}\left[\nabla_\phi Q^t \overbrace{\left(Q^t - \left(r(s_t, a_t) + \underbrace{\mathbb{E}\left[\log \int_{\mathcal{A}} \exp Q^{t+1} da_{t+1} \right]}_{\approx \max_{a \in \mathcal{A}} Q^{t+1}} \right) \right)}^{\approx \text{ standard Q learning update}} \right]$$

Soft Q-learning: integrals

- "In the case of discrete actions, this update is straightforward to implement, since the integral is replaced with a summation, and the policy can be extracted simply by normalizing the Q-function. In the case of continuous actions, a further level of approximation is needed to evaluate the integral using samples. Sampling from the implicit policy is also non-trivial, and requires an approximate inference procedure, as discussed by Haarnoja et al. (Haarnoja et al., 2017)."
- Actually, in general, evaluating $\mathbb{E}_{s_t}$, $\mathbb{E}_{s_{t+1}|s_t,a_t}$, $\mathbb{E}_{a_t|s_t}$ is nontrivial.