Carnegie Mellon
School of Computer Science

Deep Reinforcement Learning and Control

Maximum Entropy Reinforcement Learning

CMU 10-403

Katerina Fragkiadaki

RL objective

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_t R(s_t, a_t) \right]$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[\sum_t R(s_t, a_t) \right]$$

MaxEntRL objective

Promoting stochastic policies

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=1}^{T} \underbrace{R(s_t, a_t)}_{\text{reward}} + \alpha \underbrace{\mathrm{H}(\pi(\cdot \mid s_t))}_{\text{entropy}} \right]$$

Why?

- Better exploration
- Learning alternative ways of accomplishing the task
- Better generalization, e.g., in the presence of obstacles a stochastic policy may still succeed.

Principle of Maximum Entropy

Policies that generate similar rewards, should be equally probable. We do not want to commit.

Why?

- Better exploration
- Learning alternative ways of accomplishing the task
- Better generalization, e.g., in the presence of obstacles a stochastic policy may still succeed.

Reinforcement Learning with Deep Energy-Based Policies,Haarnoja et al.

We have seen this before.

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t;\theta')$
        Receive reward r_t and new state s_{t+1}
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    for $i \in \{t-1, \ldots, t_{start}\}$ **do**
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')\big(R - V(s_i;\theta'_v) + \beta \nabla_{\theta'} H(\pi(s_t;\theta'))\big)$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

"We also found that adding the entropy of the policy π to the objective function improved exploration by discouraging premature convergence to suboptimal deterministic policies. This technique was originally proposed by (Williams & Peng, 1991)"

Mnih et al., Asynchronous Methods for Deep Reinforcement Learning

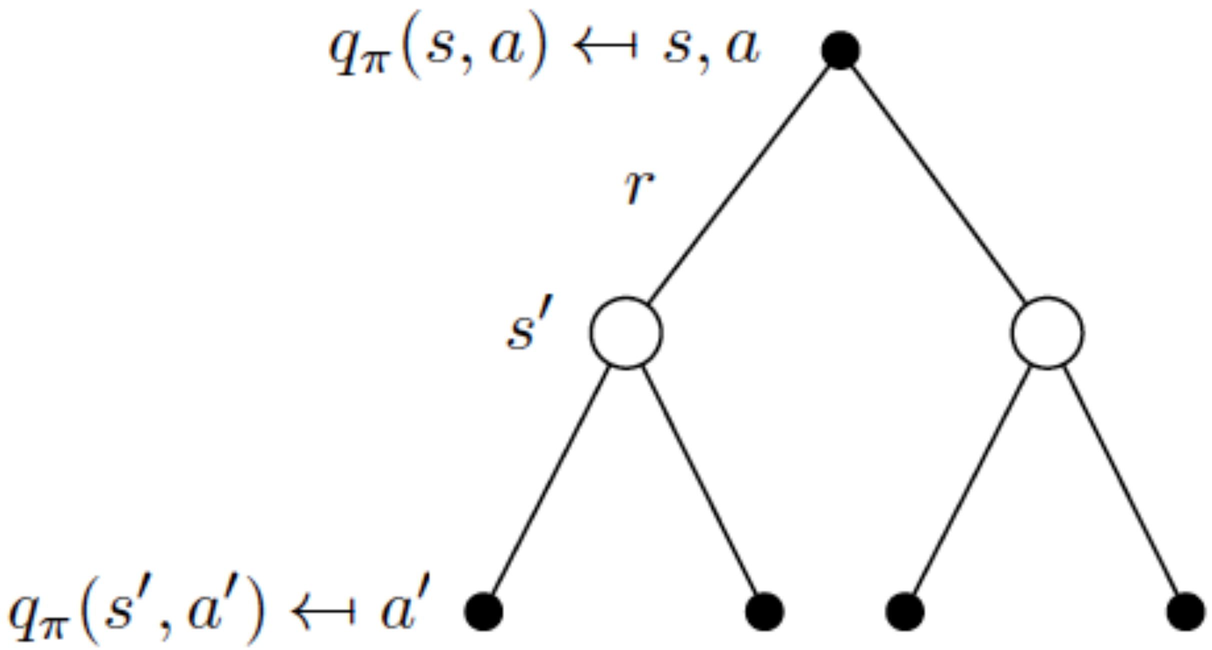
MaxEntRL objective

Promoting stochastic policies

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=1}^{T} \underbrace{R(s_t, a_t)}_{\text{reward}} + \alpha \underbrace{\mathrm{H}(\pi(\cdot \mid s_t))}_{\text{entropy}} \right]$$

How can we maximize such an objective?

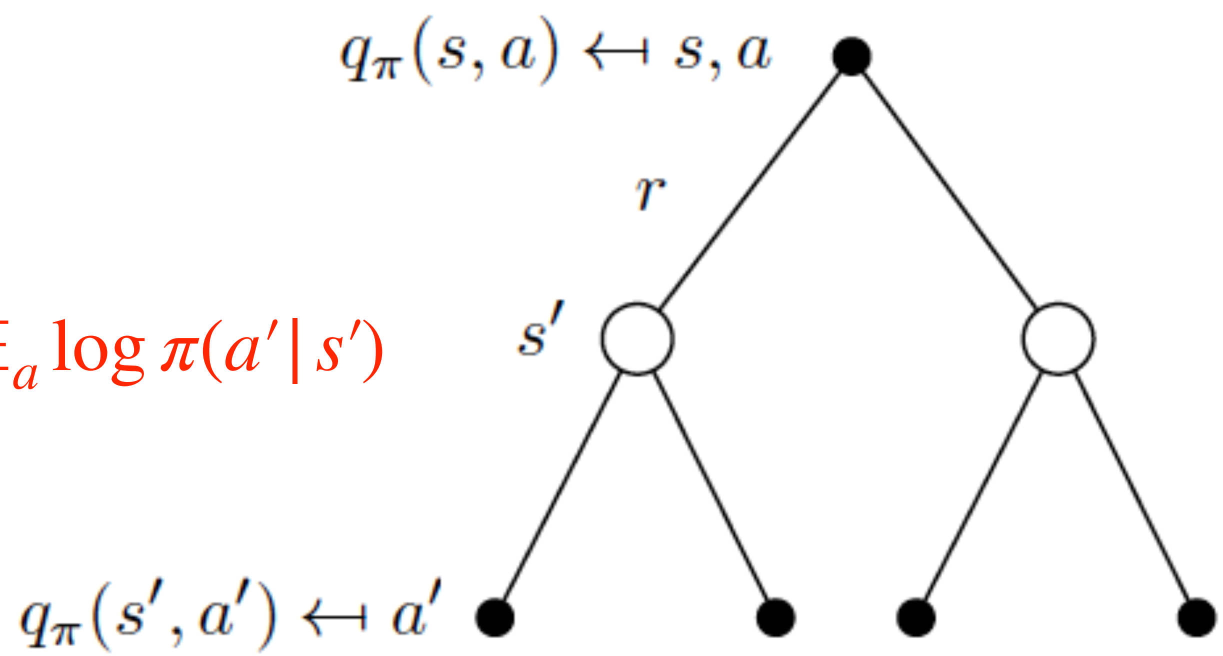
Recall:Back-up Diagrams



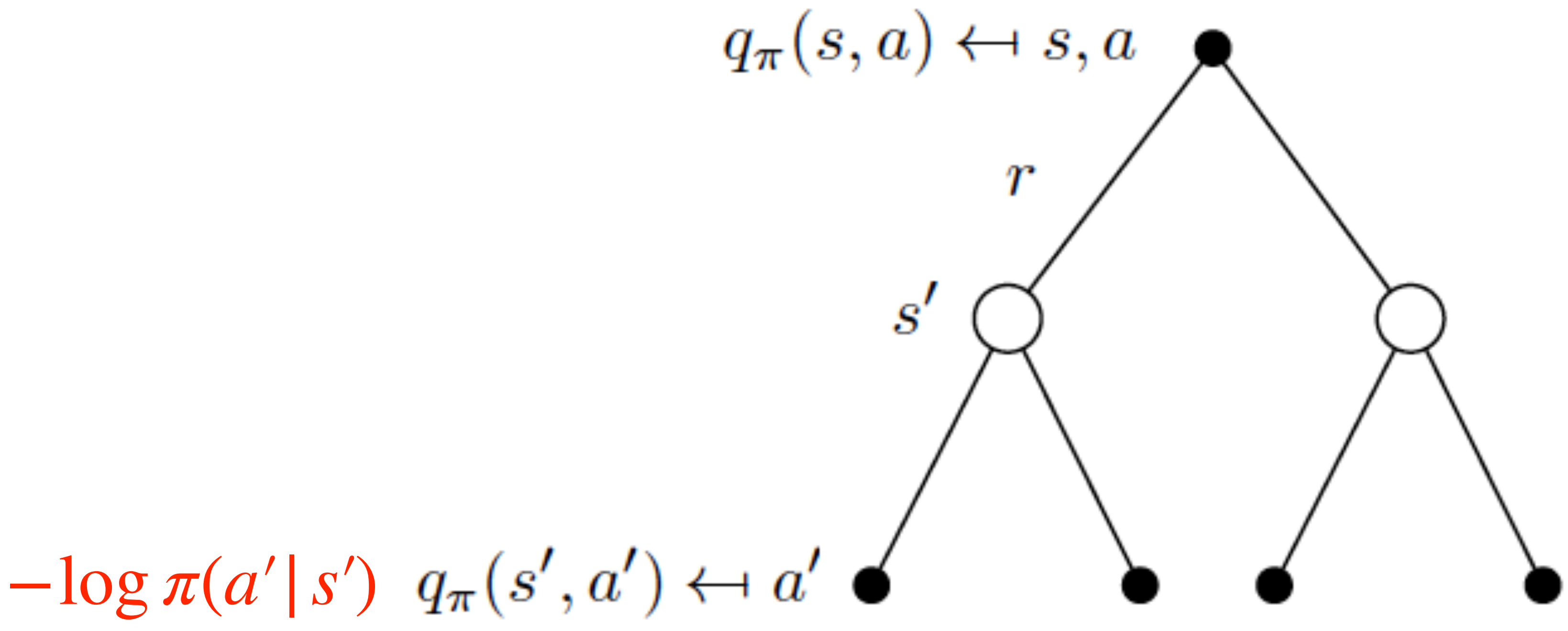
$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \,|\, s,a) \sum_{a' \in \mathcal{A}} \pi(a' \,|\, s') q_\pi(s',a')$$

Back-up Diagrams for MaxEnt Objective

$$H(\pi(\,\cdot\,|\,s')) = -\,\mathbb{E}_a \log \pi(a'|\,s')$$



Back-up Diagrams for MaxEnt Objective



$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') \Big(q_\pi(s',a') - \log(\pi(a' \mid s')) \Big)$$

(Soft) policy evaluation

Bellman backup equation:

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') q_\pi(s',a')$$

Soft Bellman backup equation:

$$q_\pi(s,a) = r(s,a) + \sum_{a',s'} T(s' \mid s,a')\Big(q_\pi(s',a') - \log(\pi(a' \mid s'))\Big)$$

Bellman backup update operator:

$$Q(s_t,a_t) \leftarrow r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1},a_{t+1}}[Q(s_{t+1},a_{t+1} \mid s_{t+1})]$$

Soft Bellman backup update operator:

$$Q(s_t,a_t) \leftarrow r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1},a_{t+1}}\left[Q(s_{t+1},a_{t+1}) - \log(\pi(a_{t+1} \mid s_{t+1}))\right]$$

Soft Bellman backup update operator is a contraction

$$Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} \left[Q(s_{t+1}, a_{t+1}) - \log(\pi(a_{t+1} \mid s_{t+1})) \right]$$

$$\begin{aligned} Q(s_t, a_t) &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho}[\mathbb{E}_{a_{t+1} \sim \pi}[Q(s_{t+1}, a_{t+1}) - \log(\pi(a_{t+1} \mid s_{t+1}))]] \\ &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho, a_{t+1} \sim \pi} Q(s_{t+1}, a_{t+1}) + \gamma \mathbb{E}_{s_{t+1} \sim \rho} \mathbb{E}_{a_{t+1} \sim \pi}[-\log \pi(a_{t+1} \mid s_{t+1})] \\ &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho, a_{t+1} \sim \pi} Q(s_{t+1}, a_{t+1}) + \gamma \mathbb{E}_{s_{t+1} \sim \rho} \mathrm{H}(\pi(\cdot \mid s_{t+1})) \end{aligned}$$

Rewrite the reward as:

$$r_{soft}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho} \mathrm{H}(\pi(\cdot \mid s_{t+1}))$$

Then we get the old Bellman operator, which we know is a contraction

Soft Bellman backup update operator

$$Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} \left[Q(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1} \mid s_{t+1}) \right]$$

$$\begin{aligned} Q(s_t, a_t) &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho}[\mathbb{E}_{a_{t+1} \sim \pi}[Q(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1} \mid s_{t+1})]] \\ &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho, a_{t+1} \sim \pi} Q(s_{t+1}, a_{t+1}) + \gamma \alpha \mathbb{E}_{s_{t+1} \sim \rho} \mathbb{E}_{a_{t+1} \sim \pi}[-\log \pi(a_{t+1} \mid s_{t+1})] \\ &\leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho, a_{t+1} \sim \pi} Q(s_{t+1}, a_{t+1}) + \gamma \alpha \mathbb{E}_{s_{t+1} \sim \rho} H(\pi(\cdot \mid s_{t+1})) \end{aligned}$$

We know that:

$$Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho}[V(s_{t+1})]$$

Which means that:

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)]$$

Review: Policy Iteration (unknown dynamics)

Policy iteration iterates between two steps:

1. Policy evaluation: Fix policy, apply Bellman backup operator till convergence

$$q_\pi(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s',a'} q_\pi(s',a')$$

4. Policy improvement: Update the policy

$$\pi'(s) \quad \doteq \quad \arg\max_a q_\pi(s,a)$$

Soft Policy Iteration

Soft policy iteration iterates between two steps:

1. Soft policy evaluation: Fix policy, apply Bellman backup operator till convergence

$$q_\pi(s,a) = r(s,a) + \mathbb{E}_{s',a'}\left(q_\pi(s',a') - \alpha\log(\pi(a' \mid s'))\right)$$

This converges to q_π

2. Soft policy improvement: Update the policy:

$$\pi' = \arg\min_{\pi_k \in \Pi} D_{KL}\left(\pi_k(\cdot \mid s_t) \,||\, \frac{\exp(\mathrm{Q}^\pi(\mathrm{s_t}, \cdot))}{Z^\pi(s_t)}\right)$$

Leads to a sequence of policies with monotonically increasing soft q values
This so far concerns tabular methods. Next we will use function approximations for policy and action values

Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Review: Policy Improvement theorem for deterministic policies

Let π, π' be any pair of determinstic policies such that, for all $s \in \mathcal{S}$:

$q_\pi(s, \pi'(s)) \geq v_\pi(s)$.

Then π' must be as good as or better than π, that is:

$v_{\pi'(s) \geq v_\pi(s)}$

Review: Policy Improvement theorem for deterministic policies

Let π, π' be any pair of determinstic policies such that, for all $s \in \mathcal{S}$:

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) .$$

Then π' must be as good as or better than π, that is:

$$v_{\pi'(s) \geq v_\pi(s)}$$

$$\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] && \text{(by (4.6))} \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] && \text{(by (4.7))} \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s\right] \\
&\ \ \vdots \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s\right] \\
&= v_{\pi'}(s).
\end{aligned}$$

Review: Policy Improvement theorem for deterministic policies

Let π, π' be any pair of determinstic policies such that, for all $s \in \mathcal{S}$:

$q_\pi(s, \pi'(s)) \geq v_\pi(s)$.

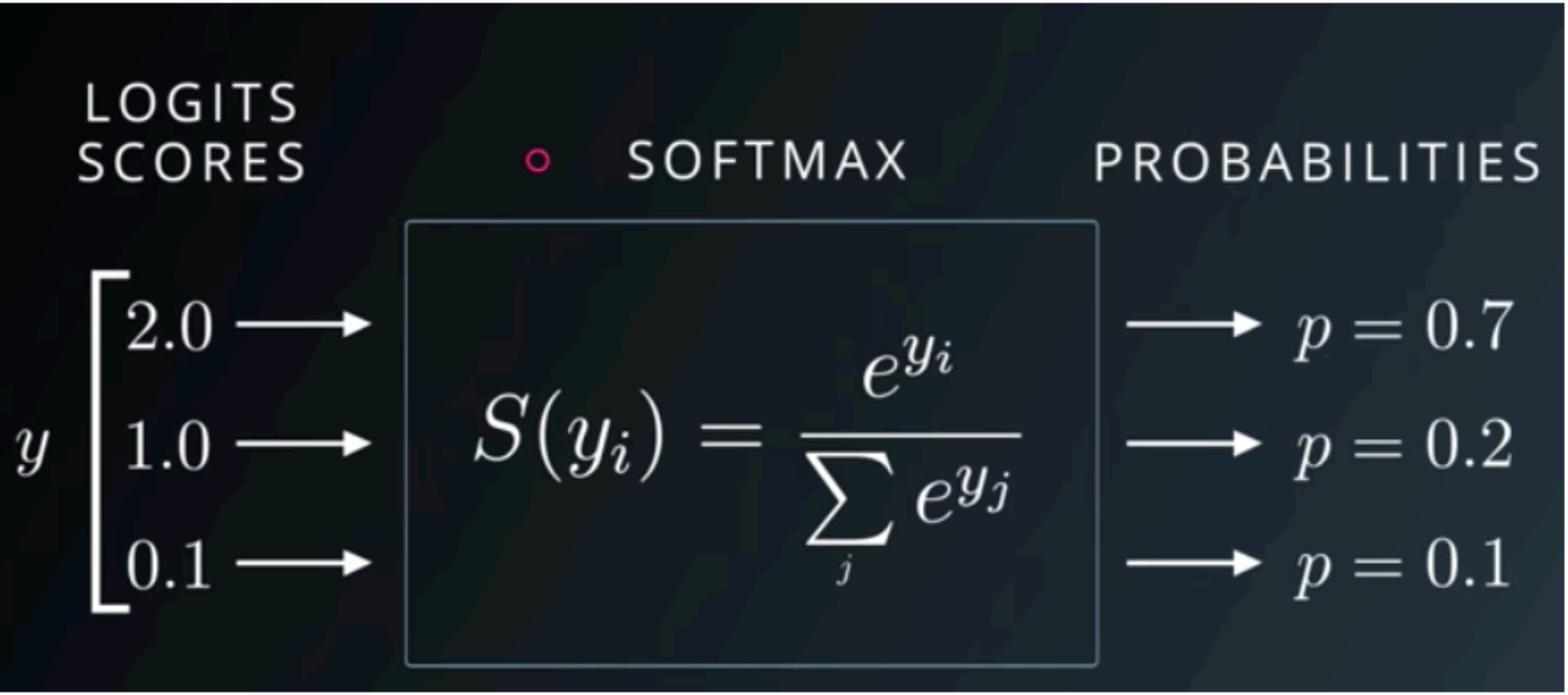
Then π' must be as good as or better than π, that is:

$v_{\pi'(s) \geq v_\pi(s)}$

$$\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s', r|s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}$$

$$\pi' = \arg\min_{\pi_k \in \Pi} D_{KL}\left(\pi_k(\cdot \mid s_t) \,||\, \frac{\exp(\mathrm{Q}^\pi(\mathrm{s_t}, \cdot))}{Z^\pi(s_t)}\right)$$

SoftMax


LOGITS
SCORES
SOFTMAX
PROBABILITIES
y
$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$
$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$
$p = 0.7$
$p = 0.2$
$p = 0.1$

Soft Policy Iteration - Approximation

Use function approximations for policy and action value functions:

$\pi_\phi(a_t \mid s_t) \qquad Q_\theta(s_t)$

Soft Policy Iteration - Approximation

Use function approximations for policy and action value functions:

$\pi_\phi(a_t \mid s_t) \qquad Q_\theta(s_t)$

1. Learning the state-action value function:

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \hat{Q}(\mathbf{s}_t,\mathbf{a}_t)\right)^2\right]$$

Semi-gradient method:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t,\mathbf{s}_t)\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \left(r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\left(Q_{\bar{\theta}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \alpha\log\left(\pi_\phi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right)\right)\right)\right)$$

Soft Policy Iteration - Approximation

Use function approximations for policy and action value functions:

$\pi_\phi(a_t | s_t) \qquad Q_\theta(s_t)$

3. Learning the policy:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D_{KL}} \left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q_\theta(\mathbf{s}_t, \cdot)\right)}{Z_\theta(\mathbf{s}_t)} \right) \right]$$

$$Z_\theta(s_t) = \int_{\mathcal{A}} \exp(Q_\theta(s_t, a_t))da_t$$

$$\nabla_\phi J_\pi(\phi) = \nabla_\phi \mathbb{E}_{s_t \in D} \mathbb{E}_{a_t \sim \pi_\phi(a|s_t)} \log \frac{\pi_\phi(a_t | s_t)}{\frac{\exp(Q_\theta(s_t, a_t))}{Z_\theta(s_t)}}$$

independent of \phi

$$\nabla_\phi J_\pi(\phi) = \nabla_\phi \mathbb{E}_{s_t \in D, \epsilon \sim \mathcal{N}(0,I)} \log \frac{\pi_\phi(a_t | s_t)}{\exp(Q_\theta(s_t, a_t))}$$

The variable w.r.t. which we take gradient parametrizes the distribution inside the expectation.

Soft Policy Iteration - Approximation

Use function approximations for policy and action value functions:

$$\pi_\phi(a_t | s_t) \qquad Q_\theta(s_t)$$

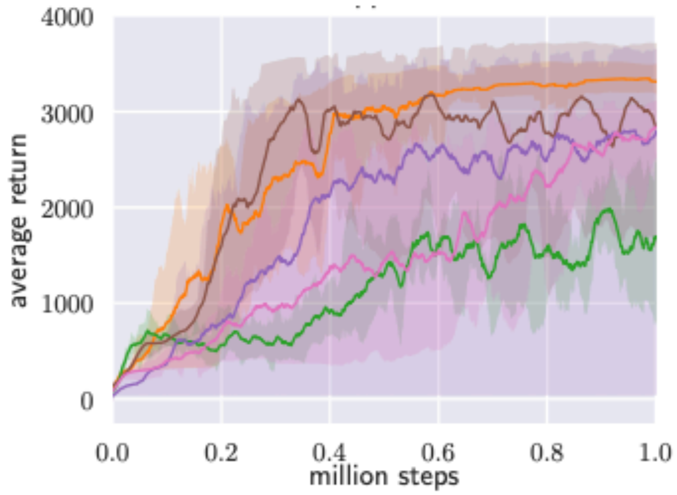
3. Learning the policy:

$$\nabla_\phi J_\pi(\phi) = \nabla_{\phi} \mathbb{E}_{s_t \in D} \mathbb{E}_{a_t \sim \pi_{\phi}(a|s_t)} \log \frac{\pi_\phi(a_t | s_t)}{\exp(Q_\theta(s_t, a_t))}$$

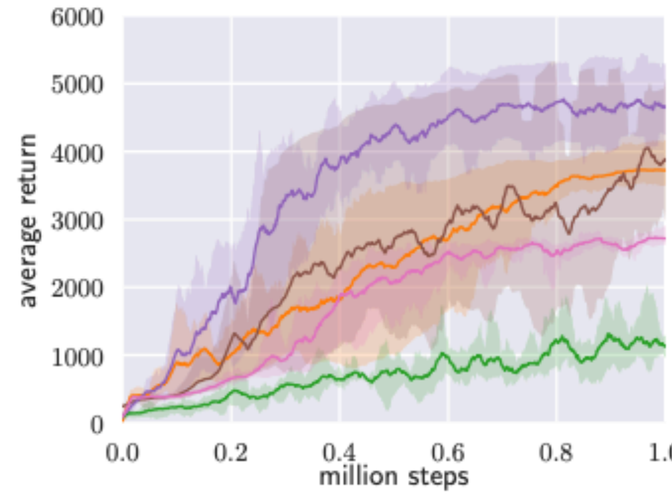
Reparametrization trick. The policy becomes a deterministic function of Gaussian random variables (fixed Gaussian distribution):

$$a_t = f_\phi(s_t, \epsilon) = \mu_\phi(s_t) + \epsilon \Sigma_\phi(s_t), \quad \epsilon \sim \mathcal{N}(0, I)$$

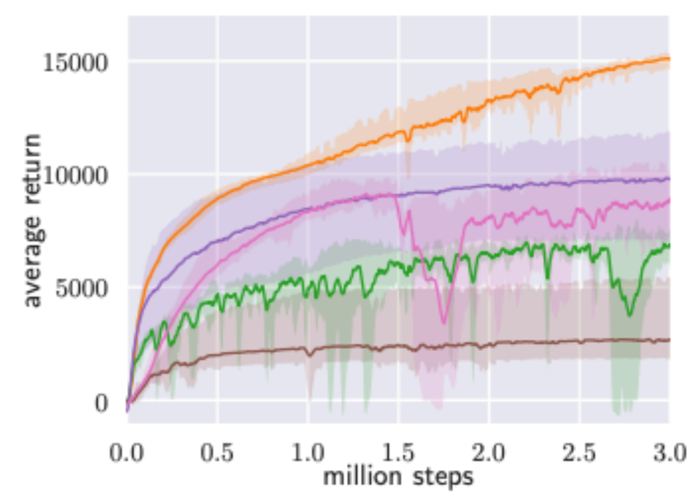
$$\nabla_\phi J_\pi(\phi) = \nabla_{\phi} \mathbb{E}_{s_t \in D, \epsilon \sim \mathcal{N}(0,I)} \log \frac{\pi_\phi(a_t | s_t)}{\exp(Q_\theta(s_t, a_t))}$$



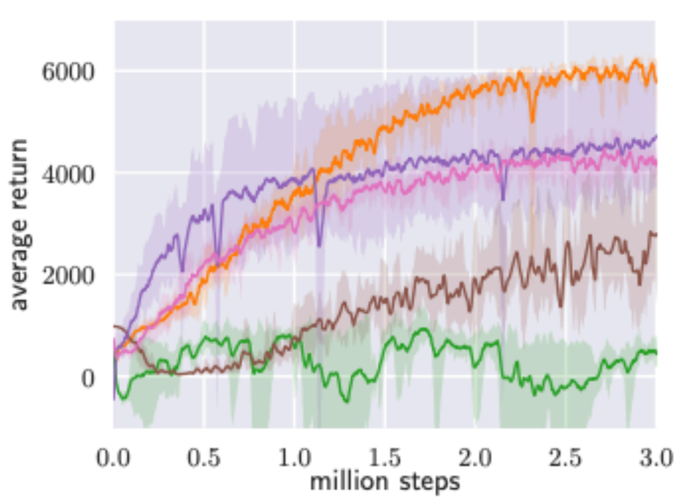
(a) Hopper-v1



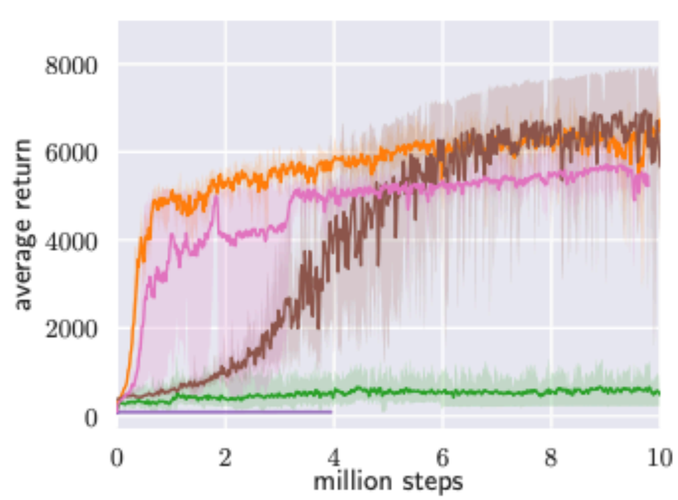
(b) Walker2d-v1



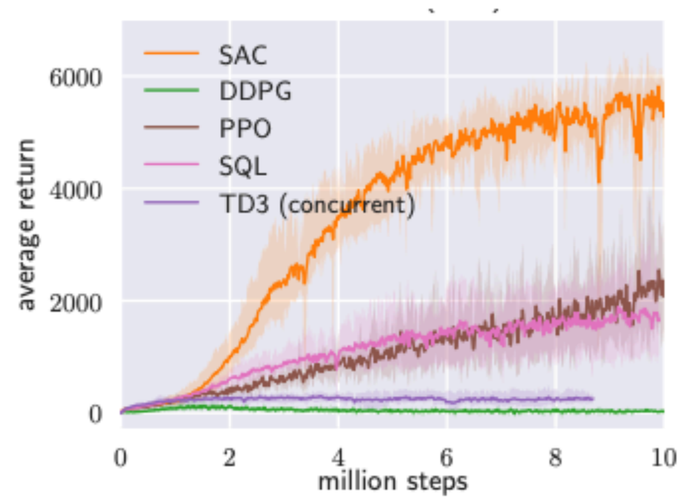
(c) HalfCheetah-v1



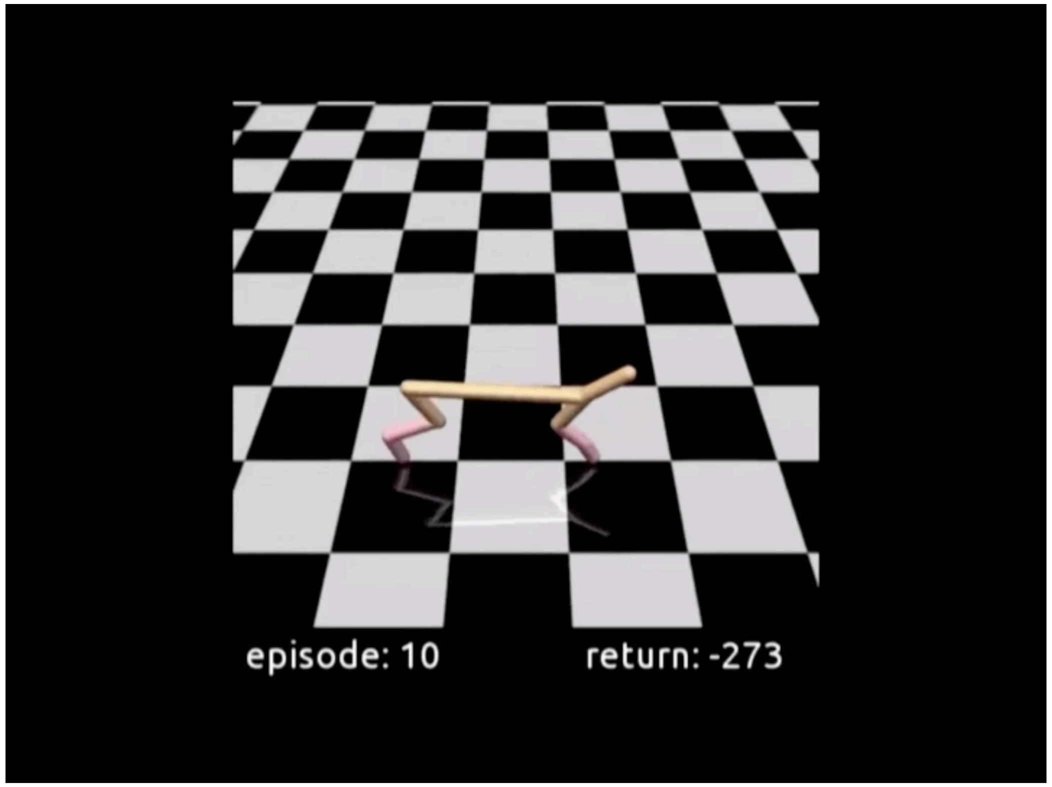
(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)



Composability of Maximum Entropy Policies

Imagine we want to satisfy two objectives at the same time, e.g., pick an object up while avoiding an obstacle. We would learn a policy to maximize the addition of the the corresponding reward functions:

$$r^C(s,a) = \frac{1}{C}\sum_{i=1}^{C} r_i(s,a)$$

MaxEnt policies permit to obtain the resulting policy's optimal Q by simply adding the constituent Qs:

$$Q_C^*(s,a) \approx \frac{1}{C}\sum_{i=1}^{C} Q_i^*(s,a)$$

We can theoretically bound the suboptimality of the resulting policy w.r.t. the policy trained under the addition of rewards. We cannot do this for deterministic policies.

Composable Deep Reinforcement Learning for Robotic Manipulation, Haarnoja et al.

https://www.youtube.com/watch?time_continue=82&v=FmMPHL3TcrE