Lecture 17: Maximum Entropy RL

Logistics:

- Project Proposals due yesterday. Mengdi and I will review them and email feedback. Come to office hours if you'd like to discuss further.
- HW6 due yesterday.
- HW7 just released, due next Mon. Will be implementing the stuff that we see today.

1 Stochastic policies

So far in this course, we've primarily been looking at *deterministic* policies, policies that output a single action for a given state. These policies are sometimes denoted $\pi(s) \to a$. For example, when doing Q-learning, select the single action that maximizes the Q function. Indeed, one can show that every MDP has a deterministic policy as a solution.[1] Intuitively, the space of all deterministic policies is smaller than the set of stochastic policies.

However, optimizing over deterministic policies has a few limitations. The biggest one is about exploration – if we want to learn by trial and error, collecting *diverse* data is important. So, when using a method like DDPG, we have to add some noise to the actions taken in the environment. This means that the actual policy used for data collection is different from the one that we're optimizing. Mathematically, this is fine, because DDPG is an off-policy algorithm. But aesthetically, it would be nice if we could optimize over the same policy that we're using for data collection. Using stochastic policies also has the potential to make optimization easier (intuition: In the TD target, because we're sampling actions now, we are averaging the Q function w.r.t. the action dimension) and avoid local optima.

Broadly, our aim will be

- If there are many ways to solve the task, find all of them!
- If there are many paths to a goal, try all possible paths, but more frequently use short paths.

2 A First Attempt (Williams and Peng, 1991)

"The use of entropy maximization is designed to help keep the search alive by preventing convergence to a single choice of output, especially when several choices all lead to roughly the same reinforcement value." – Williams and Peng (1991)

To start, we need a measure of stochasticity. We'll use perhaps the most natural measure: entropy. In particular, we'll look at the entropy of the distribution over actions, given the current state:

$$\mathcal{H}_\pi[a \mid s] \triangleq \mathbb{E}_{\pi(a|s)}[-\log \pi(a \mid s)]. \tag{1}$$

Note the negative sign, which means that maximizing entropy corresponds to trying to sample actions with low likelihood.

One straightforward way of learning policies that maximize entropy is to add the above entropy maximization term into the actor loss of an actor-critic method. The quote above

[1] When there are multiple reward maximizing policies, it's possible that there are also stochastic policies that are optimal.

is from one early paper that does this. We will use a weight of α on the entropy term

$$\max_{\pi} \mathbb{E}_{\pi}[Q(s,a) - \alpha \log \pi] \qquad \text{(actor loss)}$$

$$\min_{Q} \mathbb{E}_{p(s,a,s')} \left[\left(Q(s,a) - (r(s,a) + \gamma Q(s', a')) \right)^2 \right]. \qquad \text{(critic loss)}$$

Intuitively, this idea make sense – we're optimizing the policy to take actions that have high entropy *now* and yield high returns *in the future*. But, we can do better. In particular, if we want to maximize entropy across an entire trajectory, then t may be worthwhile to take low-entropy actions now so that we can take higher entropy actions in the future. For example, if the agent is walking along the top of a cliff, initially taking a few (deterministic) actions away from the cliff can enable it to act more randomly in the future. This idea of taking actions now that put the agent in a good position for obtaining good outcomes (either high rewards or the ability to act randomly) looks exactly the standard RL problem.

3 Maximum Entropy RL (Ziebart, 2010)

This is exactly what maximum entropy (MaxEnt) reinforcement learning does. It treats the actor entropy as part of the RL problem itself, maximizing the discounted cumulative sum of both reward and entropy. Intuitively, we can think about this as just doing RL on the augmented reward function $r^{\pi}(s,a) \triangleq r(s,a) - \alpha \log \pi(a \mid s)$. However, as we'll see below, there are a few complications with doing this directly, so we'll have to make a few minor tweaks to the actor critic algorithm. However, it should be said that these tweaks are small. As you'll see in the homework, it's possible to take an off-the-shelf actor-critic method or Q-learning and convert it into a MaxEnt RL method with just a few lines of code!

The Objective. Before going further, let's formally define the MaxEnt RL objective:

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)) \right] \tag{2}$$

$$= \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}_{\pi}[a_t \mid s_t]) \right]. \tag{3}$$

As before, we've included a weight of α on the entropy term. When $\alpha = 0$, we recover the standard RL problem.

One way to think about this objective is with a regularization lens. In the same way that we might add a small weight decay term to a supervised learning problem to make the weights closer to 0, here we're adding a small entropy term to make the policy probabilities closer to uniform. This intuition suggests that we might expect MaxEnt policies to overfit less to the training data and generalize better to new tasks; later we'll see that this intuition is correct.

Critic Objective. Like in standard actor critic methods, we will learn a Q-function. The difference now is that we'll estimate the expected discounted sum of rewards *plus entropy*. We'll estimate this using TD learning. We'll start by figuring out what the right Bellman equation is.

> **The wrong solution.** Looking at the MaxEnt RL objective (Eq. 2), one might be tempted to write down the following sort of Bellman question:
>
> $$\mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)) \right] \qquad \text{(not good)}$$
>
> $$= r(s_0, a_0) - \alpha \log \pi(a_0 \mid s_0) + \gamma \mathbb{E}_{\pi} \left[\sum_{t=1}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)) \right].$$

We could turn this into an update equation for the Q-function:

$$Q^\pi(s,a) \leftarrow r(s,a) - \log \pi(a \mid s) + \gamma \mathbb{E}_\pi[Q^\pi(s,a)], \tag{4}$$

alongside a corresponding actor update:

$$\max_\pi \mathbb{E}_\pi[Q(s,a)]. \tag{5}$$

However, this algorithm would have a hard time converging for a subtle reason: the policy that optimizes the MaxEnt RL objective is always stochastic, yet the policy that optimizes Eq. 5 is always deterministic. To fix this, we're going to do a subtle trick and perform a slight change of variables.

We're going to define the Q function at time t to only start including the entropy terms starting at time $t+1$:

$$Q(s_0,a_0) \triangleq \mathbb{E}_\pi \left[r(s_0,a_0) + \sum_{t=1}^{\infty} (r(s_t,a_t) - \alpha \log \pi(a_t \mid s_t)) \right]. \tag{6}$$

With this definition, we can write the overall MaxEnt RL objective (Eq. 2) as expected Q-values at the initial state *plus that missing entropy term from the first state*:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t,a_t) - \alpha \log \pi(a_t \mid s_t)) \right] = \mathbb{E}_{\substack{p(s_0) \\ \pi(a_0|s_0)}} [Q(s_0,a_0) - \alpha \log \pi(a_0 \mid s_0)] \tag{7}$$

This will be prove useful for obtaining an actor update that yields stochastic policies. To get a glimpse of this, note that the RHS of the equation above includes a $\log \pi(a_0 \mid s_0)$ term, so optimizing the RHS w.r.t. $\pi(a_0 \mid s_0)$ will result in a stochastic policy.

The MaxEnt RL Q-function (Eq. 6) satisfies the following identity:

$$Q(s_t,a_t) = r(s_t,a_t) + \gamma \mathbb{E}_{\pi(a_{t+1}|s_{t+1})}[Q(s_{t+1},a_{t+1}) - \alpha \log \pi(a_{t+1} \mid s_{t+1})]. \tag{8}$$

We can turn this into the critic loss for MaxEnt RL:

$$\min_Q \mathbb{E}_{p(s,a,s')}[(Q(s,a) - y_t)^2] \qquad \text{where} \quad y_t = r(s,a) + \gamma \left(Q(s',a') - \alpha \log \pi(a' \mid s') \right). \tag{9}$$

Actor Loss The actor loss will be to maximize the discounted sum of rewards and entropy, with the sum starting at the current time step for both. Because the Q function only starts including the entropy starting at the next state, we'll have to include an additional entropy term in the actor loss (which will turn out to be a good thing):

$$\max_\pi \mathbb{E}_{p(s_0)\pi(a_0|s_0)} \left[\sum_{t=0}^{\infty} r(s_t,a_t) - \alpha \log \pi(a_t \mid s_t) \right] = \mathbb{E}_{p(s_0)\pi(a_0|s_0)} \left[Q(s_0,a_0) - \alpha \log \pi(a_0 \mid s_0) \right]. \tag{10}$$

Q: Is the amount of exploration the same in all states? A: No, depends on Q. Note that this is the case even though α is the same across all states.

4 Soft Actor Critic (Haarnoja et al., 2018)

Perhaps the first practical actor-critic method for MaxEnt RL was soft actor critic (SAC).[2] The algorithmic template is similar to DDPG. The two main components are an actor network and a critic network. The actor periodically collects data that is stored in a replay buffer, and the actor and the critic are trained on data sampled from that buffer (using the actor and critic losses noted above). A target critic network is used to compute the TD target.

Q: Given a DDPG implementation, what specific lines of code would you have to change to implement SAC? A: make actor stochastic, add $\log \pi(a \mid s)$ to actor loss, add $\log \pi(a \mid s)$ to reward when pulling from the replay buffer.

[2]Note that algorithm names are not capitalized, even though the acronym is.

Q: What if you added $\log \pi(a \mid s)$ to reward when inserting into the replay buffer? A: For the critic loss, this would be biased because it would correspond to the wrong policy. For the actor, this would completely break things, because the $\log \pi(a \mid s)$ term in the actor loss now wouldn't depend on the policy that we're actually optimizing.

There tends to be a bit of confusion around what we actually call SAC because the paper itself was updated a few times in the years after it was published. For example, the original implementation called for value functions and included experiments with a Gaussian mixture model for the policy; experiments later determined that these components weren't necessary, so most implementations today do not include these components. On the other hand, after the TD3 paper (which was published afterwards) found that learning two Q functions and taking the minimum in the TD target improved performance, the SAC paper was updated to also include this idea. This explains why both papers have plots showing that their method is better than the other.

One important consideration when applying SAC is how to choose the temperature term α. I'd generally advise trying out a few different values and simply seeing what works best. In the literature (e.g., Haarnoja et al. (2018)), you'll sometimes see discussions of dynamically tuning α so that the entropy term reaches a desired target value. The main benefit of doing this is that users sometimes have a better intuition for what the target entropy should be, rather than what the value of α should be. One consideration when choosing α is that it depends on the reward scale. If you double all your rewards, then you'll also have to double α if you want to learn the same policy. One corollary of this is that you could keep α fixed at 1 and just change your reward scale.

5 Discrete Actions

The SAC method above in theory works with both discrete and continuous actions. However, in the same way that Q-learning allows us to avoid the need for a explicit actor when solving discrete action problems, it's possible to derive a MaxEnt RL method for discrete actions that doesn't require an explicit actor. The key idea is that, when working with discrete actions, the actor objective in Eq. 10 as a closed form solution.

5.1 Finding the Optimal Actor

To find this optimal actor, we have to maximize Eq. 10 while maintaining the constraint that $\pi(a \mid s)$ represent a valid probability distribution. This entails ensuring that the probabilities sum to one and that they are non-negative. As the log function isn't defined for negative probabilities, it will be sufficient to enforce the constraint that the probabilities sum to one. Let's fix a state s, so we get the following constrained optimization problem:

$$\max_{\pi} \sum_{a} \pi(a \mid s)(Q(s,a) - \alpha \log \pi(a \mid s)) \qquad \text{s.t. } \sum_{a} \pi(a \mid s) = 1. \tag{11}$$

You should think about $\pi(\cdot \mid s)$ as a big vector, where each coordinate contains the probability for a different action. We can solve this constrained optimization method using the method of Lagrange multiplies (which goes by many names, including the Euler Lagrange method; it is a special case of using the KKT conditions). We start by writing down the *relaxed* optimization problem, moving the hard constraint into the objective with a certain penalty term:

$$\mathcal{L}(\pi) = \sum_{a} \pi(a \mid s)(Q(s,a) - \alpha \log \pi(a \mid s)) + \lambda \left(\sum_{a} \pi(a \mid s) - 1 \right). \tag{12}$$

We now take the derivative of this term w.r.t. $\pi(a \mid s)$ (i.e., the a^{th} coordinate of that big long vector):

$$\frac{d}{d\pi(a \mid s)} = \pi(a \mid s)(\frac{-\alpha}{\pi(a \mid s)} + Q(s,a) - \alpha \log \pi(a \mid s) + \lambda \tag{13}$$

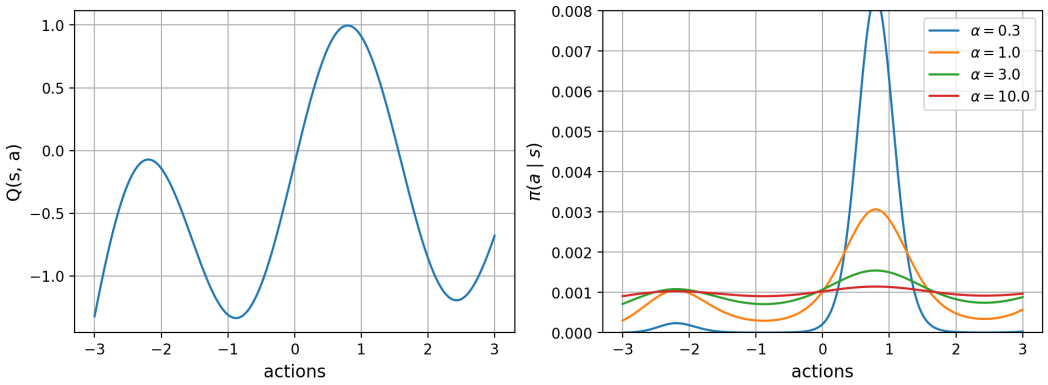

Figure 1: Visualizing the optimal policy for a particular Q function, and the effect of the entropy coefficient.

Setting the derivative equal to zero we get:

$$-\alpha + Q(s,a) - \alpha \log \pi(a \mid s) + \lambda = 0 \tag{14}$$

$$\alpha \log \pi(a \mid s) = Q(s,a) + \lambda - \alpha \tag{15}$$

$$\pi(a \mid s) = e^{\frac{1}{\alpha}Q(s,a)} e^{\frac{\lambda}{\alpha}-1}. \tag{16}$$

Finally, we solve for the value of λ that makes $\sum_a \pi(a \mid s) = 1$, we get

$$\pi(a \mid s) = \frac{e^{\frac{1}{\alpha}Q(s,a)}}{\sum_{a'} e^{\frac{1}{\alpha}Q(s,a')}} = \textsc{SoftMax}(\tfrac{1}{\alpha}Q(s,\cdot)). \tag{17}$$

The softmax is why this is called "soft" actor critic.

5.2 Intuition for the Entropy Coefficient α

Intuitively, this makes a lot of sense: the policy simply samples actions in proportion to their Q value. The entropy coefficient α determines how "peaky" this distribution is. When α is very small, Q/α is very large, so the softmax is approximately the same as choosing the argmax action. When α is very large, $Q/\alpha \approx 0$, so the softmax is approximately the same as choosing a uniform distribution over actions. You might recall from the bandits lecture at the beginning of the class that it seemed strange to always choose the best action if the second best action was nearly as good. MaxEnt RL provides a rigorous justification for using the SoftMax strategy, which avoids this dilemma. See Fig. 1 for a visualization

5.3 Critic Loss with this Optimal Actor

In the setting with discrete actions, we can replace the $\log \pi$ entropy term in the TD target with the ground truth expression for π^*; this is analogous to how Q-learning replaces the expectation over π with the argmax over actions.

$$y_t = r(s,a) + \mathbb{E}_{\pi(a'|s')}[Q(s',a') - \alpha \log \pi(a' \mid s')] \tag{18}$$

$$= r(s,a) + \alpha \mathbb{E}_{\pi(a'|s')}[\tfrac{1}{\alpha}Q(s',a') - \log e^{\frac{1}{\alpha}Q(s',a')} \sum_{a''} e^{\frac{1}{\alpha}Q(s',a'')}] \tag{19}$$

$$= r(s,a) + \alpha \left(\cancel{\mathbb{E}_{\pi(a'|s')}[\tfrac{1}{\alpha}Q(s',a') - \tfrac{1}{\alpha}Q(s',a')]} + \log \sum_{a''} e^{\frac{1}{\alpha}Q(s',a'')} \right) \tag{20}$$

$$= r(s,a) + \alpha \log \sum_{a''} e^{\frac{1}{\alpha}Q(s',a'')}. \tag{21}$$

Thus, in the tabular setting, we can write the updates for MaxEnt RL as:

$$Q(s,a) \leftarrow r(s,a) + \alpha \log \sum_{a''} e^{\frac{1}{\alpha} Q(s',a'')}. \tag{22}$$

Note how this resembles Q-learning, in that there is no explicit policy. To get intuition for the LogSumExp expression, we start by noting that the SoftMax policy converges to the argmax policy when $\alpha \to 0$. In this same limit, the LSE converges to the maximum over the Q-values:

$$\lim_{\alpha \to 0^+} \alpha \log \sum_{a''} e^{\frac{1}{\alpha} Q(s',a'')} = \max_{a''} Q(s',a''). \tag{23}$$

Intuitively this makes sense, as it would mean that we're simply doing the standard Q-learning (i.e., $\max_a$) updates. Thus, we see that we can recover Q-learning as a special case of MaxEnt RL.

6 Theory

6.1 Benefits of MaxEnt Policies

Exploration and Optimization. Smooths out the loss landscape, helps discover multiple solutions to the problem. Argument before about mitigating the mismatch between the exploration policy and the training policy.

Robustness.

- This was observed empirically in a bunch of prior work [show videos]
- Intuition: Because we're learning many ways of solving the task, we should still be able to solve the task if one of those strategies is disabled. If an adversary perturbs the environment (reward function or dynamics), still expect to succeed some fraction of the time.
- Formal result (Eysenbach and Levine, 2021):

$$f(J_{\text{MaxEnt}}(\pi; p, \tilde{r})) \leq \min_{\tilde{p},\tilde{r}} \mathbb{E}_{\tilde{p},\pi}[\sum_t r(s_t, a_t)]. \tag{24}$$

  The function $f(\cdot)$ is some monotone increasing function. On the RHS, the adversary gets to select whichever reward function $\tilde{r}(s,a)$ and dynamics $\tilde{p}(s' \mid s,a)$ makes our policy perform worst. There are some constraints on the allowable options – they have to remain close to the original dynamics and reward function. The main takeaway is that optimizing the MaxEnt RL objective (LHS) also makes your policy do better on adversarial perturbations (RHS).

6.2 Bounded Rationality

We often want to model decision making under resource constraints. For example, getting the best action/policy might take too much data or too much compute, and getting close to the optimal performance is good enough. Interestingly, we know that human and other animals do act stochastically (Gul et al., 2014). One argument for why this is a good idea is that this reflects limits in their computational abilities (other arguments include handling partial observability). MaxEnt RL provides a nice formalism for thinking about this (Jeon et al., 2020).

Instead of thinking about humans as trying to find the single best policy, we could think about them as trying to find a policy that is "good enough"; that is, whose expected discounted return is within ϵ of the best policy. This gives us a set of possible policies; we might say that humans simply choose randomly within this set. Formally, this corresponds to the notion of *satisficing* (Simon, 1956):

$$\max_{\pi} \mathbb{E}_{\pi}[\sum_t \mathcal{H}_{\pi}[a_t \mid s_t]] \qquad \text{s.t.} \quad \mathbb{E}_{\pi}\left[\sum_t r(s_t, a_t)\right] \geq (1-\epsilon) R_{\max}. \tag{25}$$

This optimization problem is equivalent to the standard MaxEnt RL problem, for a particular choice of α (which depends on ϵ).

6.3 Distribution Matching

In MaxEnt RL, we're learning not one strategy but an entire distribution over strategies/paths. We can reinterpret MaxEnt RL as performing a sort of distribution matching. The policy induces a certain distribution over trajectories:

$$\pi(\tau) = p_0(s_0) \prod_t p(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t). \tag{26}$$

We now introduce a desired distribution over trajectories, with a higher weight associated with trajectories with higher returns:

$$p^*(\tau) = \frac{1}{Z} p_0(s_0) \prod_t p(s_{t+1} \mid s_t, a_t) e^{r(s_t, a_t)}. \tag{27}$$

Note that the exponential can be rewritten as $e^{\sum_t r(s_t, a_t)}$. Now, we could think about trying to align these two distributions, trying to learn the policy s.t. these two distributions are as close as possible. Since we want to *sample* from the policy, we'll use the reverse KL divergence:

$$\min_{\pi(a|s)} D_{\mathrm{KL}}(\pi(\tau) \| p^*(\tau)) = \mathbb{E}_{\pi(\tau)} \left[\log \tfrac{\pi(\tau)}{p^*(\tau)} \right] \tag{28}$$

$$= \mathbb{E}_{\pi(\tau)} \big[\log p_0(s_0) + \sum_t (\log p(s_{t+1} \mid s_t, a_t) + \log \pi(a_t \mid s_t)) \tag{29}$$

$$- \log p_0(s_0) - \sum_t (\log p(s_{t+1} \mid s_t, a_t) + r(s_t, a_t)) \big] \tag{30}$$

$$= -\mathbb{E}_{\pi(\tau)} \left[\sum_t (r(s_t, a_t) - \log \pi(a_t \mid s_t)) \right]. \tag{31}$$

This is exactly the MaxEnt RL objective! Thus, we see that doing MaxEnt RL can be seen as trying to align two distributions.

References

Eysenbach, B. and Levine, S. (2021). Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*.

Gul, F., Natenzon, P., and Pesendorfer, W. (2014). Random choice as behavioral optimization. *Econometrica*, 82(5):1873–1912.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Jeon, H. J., Milli, S., and Dragan, A. (2020). Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33:4415–4426.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological review*, 63(2):129.

Williams, R. J. and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.