10-708 PGM (Spring 2019): Homework 4

Andrew ID:	[your Andrew ID]
Name:	[your first and last name]
Collaborators:	[Andrew IDs of all collaborators, if any]

1 Reinforcement Learning (Lisa) [30 pts]

All questions and material for this section are contained in the Colab notebook:

`https://colab.research.google.com/drive/1VkoRfg_thJuRyWlZXFNvPcRrviRlxO9I`

You do not need to submit your Colab notebook, as there are no implementation questions.

Important Note: As mentioned in Section 2.1 of the Colab notebook, we do not assume the action prior $p(a_t \mid s_t)$ to be uniform, and we define $\beta_t(s_t, a_t)$ differently, so derivations may be slightly different than in the original tutorial [1].

1.1 Exercise 1.5.1: Q-Learning (2 pts)

Train Q-Learning, then evaluate the learned greedy policy $\pi_{\text{greedy}}(a \mid s) := \arg\max_a Q(s, a)$ on the environment.

(1 pt) Describe the policy π_{greedy} in words – how does it behave?

Solution

(1 pt) How does π_{greedy} compare to the uniform policy in Section 1.4.1 in terms of average total reward?

Solution

1.2 Exercise 2.3.1: Non-Uniform Action Priors (4 pts)

(4 pts) Let $r(s_t, a_t)$ and $p(a_t \mid s_t)$ be any given reward function and action prior, respectively. Show that there exists some reward function $r_1(s_t, a_t)$ such that the posterior distribution $p(\tau \mid o_{1:T})$ is equal for the following combinations of reward function and action prior:

1. $r(s_t, a_t)$ and $p(a_t \mid s_t)$.
2. $r_1(s_t, a_t)$ and a uniform action prior.

Write down the expression for $r_1(s_t, a_t)$ in terms of $r(s_t, a_t)$ and $p(a_t \mid s_t)$.

Solution

1.3 Exercise 2.4.1: Derivation of $\beta_t(s_t)$ Update (2 pts)

(2 pts) Show that the backward messages satisfy the following update equation for $\beta_t(s_t)$:

$$\beta_t(s_t) = \sum_{a_t \in \mathcal{A}} \beta_t(s_t, a_t) \tag{1}$$

Solution

1.4 Exercise 2.4.2: Derivation of $\beta_t(s_t, a_t)$ Update (6 pts)

(6 pts) Show that the backward messages satisfy the following update equation for $\beta_t(s_t, a_t)$:

$$\beta_t(s_t, a_t) = \sum_{s_{t+1} \in \mathcal{S}} \beta_{t+1}(s_{t+1}) \mathcal{T}(s_{t+1} \mid s_t, a_t) p(O_t, a_t \mid s_t) \tag{2}$$

Solution

1.5 Exercise 2.4.3: Derivation of the Optimal Policy (5 pts)

(5 pts) Show that the optimal policy $p(a_t \mid s_t, O_{t:T})$ satisfies

$$p(a_t \mid s_t, O_{t:T}) \propto \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)} \tag{3}$$

Solution

1.6 Exercise 2.6.1: Uniform Action Prior (1 pts)

Run message-passing algorithm using a **uniform** action prior $p(a_t \mid s_t) = \frac{1}{|\mathcal{A}|}$.

(1 pts) How does the learned policy compare to the Q-Learning policy π_{greedy} from Exercise 1.5.1 in terms of behavior and average total reward?

Solution

1.7 Exercise 2.6.2: Soft Action Prior (2 pts)

Run message-passing algorithm using a "soft" action prior $\pi(a \mid s; \phi)$ for $\phi = 0.5$.

(1 pts) How does the learned policy compare to the one from Exercise 2.6.1 (using uniform action prior) in terms of behavior and average total reward?

Solution

(1 pts) (True or False) If $\phi > \frac{1}{|\mathcal{A}|}$, then using the action prior $\pi(a \mid s; \phi)$ instead of a uniform action prior is equivalent to changing the reward function $r(s_t, a_t)$ such that the agent receives relatively greater reward for taking the action $a_t = \rightarrow$ in any state, and less reward otherwise.

Solution

1.8 Exercise 2.6.3: Hard Action Prior (1 pts)

Run message-passing algorithm using a "hard" action prior $\pi(a \mid s; \phi)$ for $\phi = 1.0$.

(1 pts) How does the learned policy compare to the Q-Learning policy π_{greedy} from Exercise 1.5.1 in terms of behavior and average total reward?

Solution

1.9 Exercise 3.1.1: Unknown Transition Dynamics (2 pts)

Suppose we don't know the transition dynamics $\mathcal{T}(s_{t+1} \mid s_t, a_t)$.

(1 pt) (True or False) Can you learn the optimal policy via Q-learning?

Solution

(1 pt) (True or False) Can you learn the optimal policy via Message-Passing?

Solution

1.10 Exercise 3.1.2: Equivalence (5 pts)

(5 pts) Is it the case that the optimal message-passing policy can be equivalent to the one discovered by Q-learning? If yes, under which conditions? If no, why not?

Solution

2 Bayesian Nonparameterics (Maruan) [30 pts]

2.1 Some properties of the Dirichlet distribution (10 pts)

Let $(\pi_1, \pi_2, \ldots, \pi_n) \sim \mathrm{Dir}(\alpha_1, \ldots, \alpha_n)$.

(5 pts) Show that $(\pi_1 + \pi_2, \pi_3 \ldots, \pi_n) \sim \mathrm{Dir}(\alpha_1 + \alpha_2, \alpha_3, \ldots, \alpha_n)$. (*Hint: Prove by induction.*)

Solution

(5 pts) Show that

$$\frac{(\pi_2, \ldots, \pi_n)}{\pi_2 + \cdots + \pi_n} \sim \mathrm{Dir}(\alpha_2, \ldots, \alpha_n).$$

Solution

2.2 Posterior of the Dirichlet Process (10 pts)

Let H be a distribution over Θ and let α be a positive scalar. For any finite, measurable partition $A_1, \ldots, A_r$ of Θ, G is defined to be a Dirichlet process with the base distribution H and concentration parameter α, denoted by $G \sim \mathrm{DP}(\alpha, H)$, if

$$G(A_1), \ldots, G(A_r) \sim \mathrm{Dir}(\alpha H(A_1), \ldots, \alpha H(A_r)).$$

Suppose we have observations $X_1, \ldots, X_n$, which we assume are drawn from G. Assuming we have the prior $G \sim \mathrm{DP}(\alpha, H)$, derive the posterior distribution for $G \mid X_1, \ldots, Xn$.

(*Hint: Thinking about conjugacy between the Dirichlet and Multinomial distributions would be helpful.*)

Solution

2.3 Gaussian Processes (10 pts)

(5 pts) The main trick behind deep kernel learning [2] is to define a kernel function on the input space through another kernel on the latent space. If we have inputs $x \in \mathbb{R}^p$ that are transformed by a deep neural net into some new representation $h(x)$, we compute the kernel, $\kappa(x, x')$, in the following manner:

$$\kappa(x, x') = k(h(x), h(x')), \tag{1}$$

where is $k(\cdot, \cdot)$ is called the base kernel. Assuming that the hidden space is bounded and $k(\cdot, \cdot)$ is a valid kernel function (see definition below) defined on the hidden space, prove that $\kappa(x, x')$, as defined in (1), is similarly a valid kernel function on the input space.

Definition (Valid kernel function). We call a two-argument function, $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ a valid kernel function if it is:

1. symmetric, *i.e.*, $\kappa(x, x') = \kappa(x', x)$ for all $(x, x') \in \mathcal{X}^2$, and

2. positive-semidefinite, *i.e.*, $\sum_{i=1}^{n}\sum_{j=1}^{n}\kappa(x_i, x_j)c_i c_j \geq 0$ for any finite sequence $x_1, \ldots, x_n \in \mathcal{X}$ and any $c_1, \ldots, c_n \in \mathbb{R}$.

Solution

(5 pts) A simple first-order autoregressive process, AR(1), is defined as follows:

$$y_t = a + by_{t-1} + \varepsilon_t,\ t = 1, 2, \ldots,\ y_0 = a, \tag{2}$$

where a and b are some constants and $\varepsilon_t \sim \mathcal{N}(0, 1)$ is Gaussian noise. AR(1) defines a distribution over sequence of discrete values, $\{y_0, y_1, \ldots\}$ (to sample from this distribution, you can simply run the forward autoregressive recursion).

Derive a mean, $\mu(t)$, and a kernel, $k(t, t')$, functions for a Gaussian process that defines a distribution over functions, $y(t)$, that coincides with AR(1) for all $t = 1, 2, \ldots$.

(*Hint: Unroll the recursion to compute the mean and covariance functions.*)

Solution

References

[1] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[2] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.