0.1 Derivation

我们的算法是分别对上下层策略进行更新，接下来我们证明，单独更新上层策略和单独更新下层策略都可以使得整体策略变优。

在接下来的推导过程中，我们用上标或下标 h 和 l 来表示强化学习中的一系列参数，是针对上层策略（high）还是下层策略（low）。

首先我们定义对整体策略好坏的度量方法。注意到，对于上层策略来说，底层策略相当于环境动力学（dynamics）。当底层策略发生了变化时，可以看做是环境动力学发生了变化。我们用 $\mathcal{E}(\pi_l)$ 表示环境，并且环境与 π_l 有关。定义 π_h 带来的折扣奖励期望

$$\begin{aligned} \eta(\pi_h) = V_h(s_0^h) = \mathbb{E}_{s_0^h, a_0^h, \ldots}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} r_h(s_t^h, a_t^h)\right], \text{where} \\ s_0^h \sim \rho_0^h(s_0^h), a_t^h \sim \pi_h(s_t^h), s_{t+1} \sim P(s_{t+1}^h | s_t^h, a_t^h, \mathcal{E}(\pi_l)) \end{aligned} \tag{1}$$

这个期望值同时和 π_h 和 π_l 有关，也就是我们对整体策略的度量。

首先，我们说明，在下层策略不变的情况下，利用 TRPO 更新上层策略，可以保证整体策略的单调递增，即 $\eta(\pi_h)$ 单调递增。这一步很简单，既然底层策略不变，那么可以认为底层策略是包含在环境中的，因此 TRPO 去优化上层策略，就相当于 TRPO 直接作用于一个单层的强化学习结构，对应的优化目标将是近似单调递增的。注意到对于上层策略来说，它得到的 R 就是上下层策略作为一个整体得到的真实 R。

接下来，我们推导，采用 $\frac{\gamma_h V_h(s_{t+k}^h) - V_h(s_t^h)}{k}$ 作为上层给下层的输入，从而优化 π_l，也可以使得整体策略变优。

我们优化策略的算法采用的是 TRPO[1]。在 TRPO 当中，我们最终希望去优化的东西是 $V_\pi(s_0)$。TRPO 算法等价于在基于策略梯度定理给出的策略梯度优化中，增加了对步长的限制，从而确保策略更新的单调性。在策略梯度方法的论文[2] 中，Sutton 指出，对一个策略优化的目标函数 J 既可以采用 $V_\pi(s_0)$ 的定义，也可以采用 R 的平均值的定义，它们可以分别推导得出策略梯度定理，而这个定理中的策略梯度，与 TRPO 试图优化的替代函数的梯度是相等的。因此，TRPO 也可以理解成是在最大化 $\mathbb{E}_{s,a\sim\pi}[R(s,a)]$。

如果把 R 的均值作为优化目标，我们有

$$\nabla J(\theta) = \nabla \mathbb{E}_{s\sim\pi}[\sum_a \pi(s,a) R(s,a)] = \nabla \mathbb{E}_{s,a\sim\pi}[R(s,a)]. \tag{2}$$

我们定义了

$$R_l(s^l_{t+i}, a^l_{t+i})|_{i=0,1,\ldots,k-1} = \frac{V_h(s^h_{t+k}) - V_h(s^h_t)}{k}. \tag{3}$$

前面我们提到，TRPO 完成的任务是最大化 $\mathbb{E}_{s,a\sim\pi}[R(s,a)]$，因此我们在利用 TRPO 针对底层策略进行优化的时候，相当于最大化 $\mathbb{E}_{s,a\sim\pi_l,\pi_h}[R_l(s_l,a_l)]$。注意这里，我们把 π_h 看做是一个固定的概率分布函数，而不对它进行优化。π_h 可以看做是环境的一部分，而它输出的隐式编码（latent code）则是我们观察（observation）的一部分。

根据 (3) 中的结果，我们两边取期望，可以得到

$$\mathbb{E}_{s^l,a^l\sim\pi_l,\pi_h}[R_l(s^l,a^l)] = \frac{1}{k}\mathbb{E}_{s^h,a^h\sim\pi_l,\pi_h}[\gamma_h V_h(s^h_{t+k}) - V_h(s^h_t)]. \tag{4}$$

回忆在强化学习中，我们关于优势函数 $A(s,a)$ 的定义，为

$$A(S_t,A_t) = Q(S_t,A_t) - V(S_t) = R(S_t,A_t) + \gamma V(S_{t+1}) - V(s). \tag{5}$$

由于在稀疏强化学习问题中有稀疏奖励条件：

$$R(S_t,A_t) = 0, \forall t \neq t_{end}. \tag{6}$$

因此，(5) 变为

$$A(S_t,A_t) = \gamma V(S_{t+1}) - V(s). \tag{7}$$

由 (4) 和 (7) 可知，TRPO 优化下层策略的结果，等效于优化了上层的优势函数，也就是

$$\mathbb{E}_{s^l,a^l\sim\pi_l,\pi_h}[R_l(s^l,a^l)] = \frac{1}{k}\mathbb{E}_{s^h,a^h\sim\pi_l,\pi_h}[A^h(s^h_t,a^h_t)]. \tag{8}$$

接下来我们证明，通过最大化这个期望值，就近似等效于优化了整体策略的表现。

我们用 $\tilde{\eta}(\pi_h)$ 来表示 π_l 发生变化（变为 $\tilde{\pi}_l$）而 π_h 没有发生变化以后的折扣奖励值期望。将 TRPO 根据[3] 得到的引理 1（**Lemma 1**）稍加变形，我们可以得到类似的结果：（具体推导参见附录中证明 1）

$$\tilde{\eta}(\pi_h) = \eta(\pi_h) + \mathbb{E}_{s^h_0,a^h_0,\ldots\sim\pi_h,\mathcal{E}(\tilde{\pi}_l)}\left[\sum_{t=0,k,2k,\ldots}\gamma_h^{t/k} A_{\pi_h}(s^h_t,a^h_t)\right]. \tag{9}$$

等式 (9) 右边第二项的含义就是，新的整体策略相对于旧的整体策略的优势。我们想要最大化的就是这个优势。

接下来，我们定义折扣访问频率 ρ 为

$$\rho_{\pi_h}(s^h) = \sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} P(s_t^h = s|\mathcal{E}(\pi_l)) \tag{10}$$

当底层策略发生改变而上层策略不变的时候，ρ_{π_h} 相应地变为 $\tilde{\rho}_{\pi_h}$。

表达式 (9) 中，等式右侧第二项可以变形为（参见 TRPO 论文中的等式（2）推导，写在附录中证明 2）

$$\sum_{s^h} \tilde{\rho}_{\pi_h}(s^h) \sum_{a^h} \pi_h(a^h|s^h) A_{\pi_h}(s^h, a^h). \tag{11}$$

对于一个起点随机产生的，非周期、不可约的稳态马尔可夫问题，我们有

$$P(s_0^h = s^h) = P(s_1^h = s^h) = \ldots = P(s_i^h = s^h) \tag{12}$$

注意到这里 $P(s_i^h = s^h)$ 也就是采样中状态的分布。

将 (12) 带入 (11)，可以得到

$$\tilde{\rho}_{\pi_h}(s^h) = \frac{1}{1-\gamma_h} P(s_i^h = s^h) \tag{13}$$

从而，我们可以把 (9) 写成期望的形式：

$$\tilde{\eta}(\pi_h) = \eta(\pi_h) + \frac{1}{1-\gamma_h} \mathbb{E}_{s^h, a^h \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \Big[A_{\pi_h}(s^h, a^h) \Big]. \tag{14}$$

我们发现最大化 (8) 恰好也就最大化了 (14)。由此我们证明了，只要利用 TRPO 对下层策略进行了更新，$\eta(\pi_h)$，也就是整体策略的表现度量，也将是近似单调递增的（因为 TRPO 是近似单调递增的）。

综上所述，如果我们单次更新，分别固定 π_h 更新 π_l 或者是固定 π_l 更新 π_h，都将使得整体策略的价值 $\eta(\pi_h)$ 单调递增。

在实际操作中，为了提高采样效率，我们每次采样之后，都既优化 π_h 又优化 π_l。实验结果表面，同时优化上下层策略虽然在理论上无法保证整体策略 $\eta(\pi_h)$ 的单调增特性，却能够将采样效率提高为交替优化单层策略的两倍。也就是说，在每次更新 π_l 和 π_h 变化都较小的情况下，同时优化两者并不会导致整体策略效果变差。

参考文献

[1] Schulman J, Levine S, Moritz P, et al. Trust region policy optimization[C]//ICML. [S.l.: s.n.], 2015.

[2] Sutton R S, McAllester D A, Singh S P, et al. Policy gradient methods for reinforcement learning with function approximation[M]//Solla S A, Leen T K, Müller K. Advances in Neural Information Processing Systems 12. [S.l.]: MIT Press, 2000: 1057-1063

[3] Kakade S, Langford J. Approximately optimal approximate reinforcement learning[C]//ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning. [S.l.: s.n.], 2002: 267-274.

证明 1 源自 TRPO 论文的引理 1，在表达式上有区别，但是思路相同。

证明 1：

$$\tilde{\eta}(\pi_h) = \eta(\pi_h) + \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_{\pi_h}(s_t^h, a_t^h) \right]. \tag{15}$$

证：

$$\mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_{\pi_h}(s_t^h, a_t^h) \right] \tag{16}$$

$$= \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} (r_h(s_t^h) + \gamma_h V_{\pi_h}(s_{t+k}^h) - V_{\pi_h}(s_t^h)) \right] \tag{17}$$

$$= \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[-V_{\pi_h}(s_0^h) + \sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} r_h(s_t^h) \right] \tag{18}$$

$$= -\eta(\pi_h) + \tilde{\eta}(\pi_h) \tag{19}$$

证明 2：

$$\mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_{\pi_h}(s_t^h, a_t^h) \right] = \sum_{s^h} \tilde{\rho}_{\pi_h}(s^h) \sum_{a^h} \pi_h(a^h | s^h) A_{\pi_h}(s^h, a^h). \tag{20}$$

证：

$$\mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi_h, \mathcal{E}(\tilde{\pi}_l)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_{\pi_h}(s_t^h, a_t^h) \right] \tag{21}$$

$$= \sum_{t=0,k,2k,\ldots} \sum_{s^h} P(s_t^h = s^h | \pi_h, \mathcal{E}(\pi_l)) \sum_{a^h} \pi_h(a^h|s^h) \gamma_h^{t/k} A_{\pi_h}(s^h, a^h) \tag{22}$$

$$= \sum_{s^h} \sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} P(s_t^h = s^h | \pi_h, \mathcal{E}(\pi_l)) \sum_{a^h} \pi_h(a^h|s^h) A_{\pi_h}(s^h, a^h) \tag{23}$$

$$= \sum_{s^h} \tilde{\rho}_{\pi_h}(s^h) \sum_{a^h} \pi_h(a^h|s^h) A_{\pi_h}(s^h, a^h). \tag{24}$$