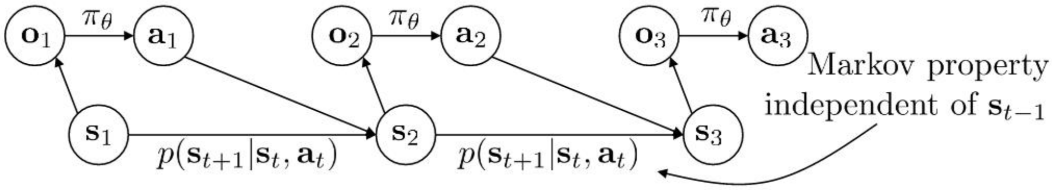
Lecture 1　Supervised Learning of Behaviors



Notations: s_t-state, o_t -observation, a_t -action, $\pi_\theta(a_t|o_t)$, $\pi_\theta(a_t|s_t)$ -policy(maybe parameters of NN)

Dataset Aggregation(DAgger):

训练参数生成的结果与数据集结果不一致. 如何使$p_{data}(o_t)=p_{\pi_\theta}(o_t)$. 参数训练并不容易, 于是采用的思想是改变$p_{data}(o_t)$. 这里$p(o_t)$表示经过时间$t$到达状态的概率分布.

DAgger Algorithm

1. Train $\pi_\theta(a_t|o_t)$ from human data $\mathcal{D}=\{o_0,\ a_0...o_N,\ a_N\}$
2. Run sequentially with $\pi_\theta(a_t|o_t)$ to get generated dataset $\mathcal{D}_\pi=\{o_1',\ o_2'...o_M'\}$
3. Label $\mathcal{D}_\pi$ with actions $\{a_t'\}$ by human
4. Aggregate: $\mathcal{D}\leftarrow\mathcal{D}\cup\mathcal{D}_\pi$, then repeat from step 1.

DAgger 算法存在的问题:

- 通常而言 Markov 性是不现实的, 因此假设$\pi_\theta(a_t|o_1,\ o_2...o_t)$是更合理的. 这里可以采用 RNN 结构(通常是 LSTM cell), 并且每个 cell 的π_θ参数可部分共享
- Multimodal behavior. 在一些情境中解决方法不止一种(e.g. 左转或右转), 简单有限情形可以用 softmax. 但复杂情况下单一的 Gaussian distribution 无法表述(e.g. 左右皆可就会出现峰值在直行的错误方案). 可以用 mixture of Gaussian 或 latent variable model 解决

Reward 与 cost function 的选择:
考虑走钢丝模型(即走错一步后不能回到正确轨道). 假设

$$r(s,a)=\log p(a=\pi^*(s)|s) \tag{1.1}$$

$$c(s,a)\begin{cases}0, & \text{if } a=\pi^*(s)\\ 1, & \text{otherwise}\end{cases} \tag{1.2}$$

若考虑优化结果为$\pi_\theta(a \neq \pi^*(s)|s) \leqslant \epsilon$ for all $s \in \mathcal{D}_{train}$. 则简单模仿学习(naive behavioral cloning)的代价

$$\begin{aligned} \mathbb{E}\left[\sum_t c(s_t, a_t)\right] &\leqslant \epsilon T + (1-\epsilon)\left(\epsilon(T-1) + (1-\epsilon)(\ldots)\right) \\ &\sim O(\epsilon T^2) \end{aligned} \tag{1.3}$$

这一结果随T快速增加，不合适.

- 改进：将 for all $s \in \mathcal{D}_{train}$改为 for $s \sim p_{train}(s)$，在 DAgger 算法下，$p_{train}(s) \to p_\theta(s)$，则

  $E\left[\sum_t c(s_t, a_t)\right] \leqslant \epsilon T$.

- 若$p_{train}(s) \neq p_\theta(s)$，有

$$p_\theta(s_t) = (1-\epsilon)^t p_{train}(s_t) + (1-(1-\epsilon)^t) p_{mistake}(s_t) \tag{1.4}$$

其中前一部分是每一步正确的概率，后一部分是出现任何一次错误的概率. $p_{mistake}(s_t)$是一个无法估计量.
由式(1.4)可以得到

$$\begin{aligned} |p_\theta(s_t) - p_{train}(s_t)| &= (1-(1-\epsilon)^t)\,|p_{mistake}(s_t) - p_{train}(s_t)| \\ &\leqslant 2(1-(1-\epsilon)^t) \\ &\leqslant 2\epsilon t \end{aligned} \tag{1.5}$$

于是

$$\begin{aligned} \sum_t \mathbb{E}_{p_\theta(s_t)}[c_t] &= \sum_t \sum_{s_t} p_\theta(s_t) c_t(s_t) \\ &\leqslant \sum_t \sum_{s_t} p_{train}(s_t) c_t(s_t) + |p_\theta(s_t) - p_{train}(s_t)| c_{\max} \\ &\leqslant \sum_t \epsilon + 2\epsilon t \leqslant \epsilon T + 2\epsilon T^2 \sim O(\epsilon T^2) \end{aligned} \tag{1.6}$$

Lecture 2　Reinforcement Learning

目标：在给定 trajectory distribution

$$p_\theta(\tau) \equiv p_\theta(s_1, a_1 \ldots s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2.1}$$

的条件下优化

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{p_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right] \tag{2.2}$$

Q-function & value function:

Q-function 表示在s_t时采取a_t的总回报

$$Q^{\pi}(s_t,a_t)=\sum_{t'=t}^{T}\mathbb{E}_{\pi_\theta}[r(s_{t'},a_{t'})|s_t,a_t] \tag{2.3}$$

Value function 表示在s_t下预期总回报

$$\begin{aligned} V^{\pi}(s_t) &= \sum_{t'=t}^{T}\mathbb{E}_{\pi_\theta}[r(s_{t'},a_{t'})|s_t] \\ &= \mathbb{E}_{a_t\sim\pi(a_t|s_t)}[Q^{\pi}(s_t,a_t)] \end{aligned} \tag{2.4}$$

几种 RL 算法:

- Policy gradients *(lecture 3)*
  1. Run the policy and generate samples.
  2. Estimate returns $R_\tau=\sum_t r(s_t,a_t)$
  3. Improve policy $\theta\leftarrow\theta+\alpha\nabla_\theta\mathbb{E}\left[\sum_t r(s_t,a_t)\right]$
  4. Back to step 1 and repeat.
- Actor-critic (model-based) *(lecture 4)*
  1. Run the policy and generate samples.
  2. Fit a model ($V(s)$ or $Q(s,a)$)
  3. Improve policy $\theta\leftarrow\theta+\alpha\nabla_\theta\mathbb{E}\left[\sum_t r(s_t,a_t)\right]$
  4. Back to step 1 and repeat.

Lecture 3　Policy Gradients

先考虑 policy 有限的情况:

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}}\,\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t r(s_t,a_t)\right] \\ &\equiv \underset{\theta}{\operatorname{argmax}}\,\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\tau)\right]\equiv\underset{\theta}{\operatorname{argmax}}\,J(\theta) \end{aligned}.$$

于是

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\
&= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]
\end{aligned} \tag{3.1}
$$

根据(2.1)可以得到

$$
\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) \tag{3.2}
$$

$$
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \tag{3.3}
$$

- 与 maximum likelihood: $\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right)$相比，ML 的$\pi_\theta$是 ground truth，而 policy gradients 的是前一次优化得到的.

存在的问题: (3.3)式给出的优化方式 variance 很大.
优化思路：减去一个 baseline.

$$
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p_\theta(\tau) [r(\tau) - b] \tag{3.4}
$$

依据：类似式(3.1)，$\mathbb{E}[\nabla_\theta \log p_\theta(\tau) \cdot b] = 0$. 因此这一操作是 unbiased. 进一步方差

$$
\begin{aligned}
\mathrm{Var} &= \mathbb{E}_{\tau \sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)[r(\tau) - b])^2] - \mathbb{E}^2_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)[r(\tau) - b]] \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)[r(\tau) - b])^2] - \mathbb{E}^2_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]
\end{aligned} \tag{3.5}
$$

若希望$\frac{d\mathrm{Var}}{db} = 0$，可以得到$b = \frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]}$，其中$g(\tau) = \nabla_\theta \log p_\theta(\tau)$.

On-policy & off-policy

按照(3.3)的优化算法每次都需要重新采样(step 1.)，需要依赖$\pi_\theta(a_t|s_t)$. 如果采样成本高或很难采样，就不适合.

改进：用 off-policy. 假设真实θ分布的样本只有一部分，则问题变为

$$
\begin{aligned}
\theta^* &= \arg\max_{\theta'} J(\theta') \\
J'(\theta') &= \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau) \right]
\end{aligned} \tag{3.6}
$$

于是

$$
\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \tag{3.7}
$$

由(2.1)得到

$$\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} = \frac{\prod \pi_{\theta'}(a_t|s_t)}{\prod \pi_\theta(a_t|s_t)} \tag{3.8}$$

代入(3.7)可得

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\prod_{t=1}^{T}\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\right)\left(\sum_{t=1}^{T}r(s_t|a_t)\right)\right] \tag{3.9}$$

考虑到 future actions 不影响当前权重，已有 reward 不影响后续决策．可以简化为

$$\begin{aligned}\nabla_{\theta'} J(\theta') &= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\left(\prod_{t'=1}^{t}\frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})}\right)\left(\sum_{t'=t}^{T}r(s_{t'}|a_{t'})\right)\right] \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{\pi_{\theta'}(s_{i,t},a_{i,t})}{\pi_\theta(s_{i,t},a_{i,t})}\nabla_{\theta'}\log\pi_{\theta'}(a_{i,t}|s_{i,t})\cdot\hat{Q}_{i,t} \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{\pi_{\theta'}(s_{i,t})}{\pi_\theta(s_{i,t})}\frac{\pi_{\theta'}(a_{i,t}|s_{i,t})}{\pi_\theta(a_{i,t}|s_{i,t})}\nabla_{\theta'}\log\pi_{\theta'}(a_{i,t}|s_{i,t})\cdot\hat{Q}_{i,t}\end{aligned} \tag{3.10}$$

(这里 3.10 式存疑?为什么可以化为求和形式)

忽略 $\frac{\pi_{\theta'}(s_{i,t})}{\pi_\theta(s_{i,t})}$ 项，这样$\left(\prod_{t'=1}^{t}\frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})}\right)$项原本的指数增长化为线性增长.

Lecture 4　Actor-Critic Algorithms

- 一些概念区分:

  $Q^\pi(s_t,a_t)$(2.3): total reward from taking a_t in s_t.

  $V^\pi(s_t)$(2.4): total reward from s_t.

  $A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t)$: how much better a_t is.

从(3.4)出发，这里选择$b = A^\pi(s_t,a_t)$以降低 variance.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(a_{i,t}|s_{i,t})A^\pi(s_{i,t},a_{i,t}) \tag{4.1}$$

解释：选择 Advance function 会倾向选择更好的 action，因此是 biased．但显著降低了 variance.
考虑到

$$Q^\pi(s_t,a_t) \approx r(s_t,a_t) + V^\pi(s_{t+1}) \tag{4.2}$$

这里$\approx$是由于$t+1$时的策略不再是π，但很接近．于是

$$A^\pi(s_t,a_t) = r(s_t,a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \tag{4.3}$$

只需拟合$V^\pi(s)$.

- 这里不适合用 Monte Carlo 方法，因为这样 simulator 需要 reset. 因此直接使用单次结果 $V^{\pi}(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$构建训练集$\{(s_{i,t}, V^{\pi}(s_{i,t}))\}$做监督学习.

- 利用 bootstrapped 思想进一步改进: $\left\{\left(s_{i,t}, r(s_{i,t}, a_{i,t}) + \hat{V}_{\phi}^{\pi}(s_{i,t+1})\right)\right\} \leftarrow \{(s_{i,t}, V^{\pi}(s_{i,t}))\}$. 可以降低 variance 但会增加 bias.

Discounted factor:

为了避免可能的$V^{\pi} \to \infty$，这里选择加入 discount. 即

$$V_{\phi}^{\pi}(s_{i,t}) \approx r(s_{i,t}, a_{i,t}) + \gamma \hat{V}_{\phi}^{\pi}(s_{i,t+1}),\ (\gamma < 1) \tag{4.4}$$

$$\hat{A}^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t) \tag{4.5}$$

Online Actor-Critic Algorithm

1. Take actions $a \sim \pi_{\theta}(a|s)$, obtain (s, a, s', r).
2. Update $\hat{V}_{\phi}^{\pi}$ with the target $r + \gamma \hat{V}_{\phi}^{\pi}(s')$.
3. Evaluate $\hat{A}^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t)$.
4. $\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta}(a|s) \hat{A}^{\pi}(s, a)$.
5. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$.

- (4.5)式给出的$\hat{A}^{\pi}$具有 lower variance，但 value function 经常是 biased. 而其等效表达式

$$\hat{A}^{\pi}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_{\phi}^{\pi}(s_t) \tag{4.6}$$

是 single sample estimate，因此具有 low bias high variance.

- 改进思路: n-step estimation. 调整λ就可以权衡 bias /variance.

$$\begin{aligned} \hat{A}^{\pi}(s_t, a_t) &= \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_{\phi}^{\pi}(s_t) + \gamma^{n} \hat{V}_{\phi}^{\pi}(s_{t+n}) \\ \hat{A}_{GAE}^{\pi}(s_t, a_t) &= \sum_{n=1}^{\infty} w_n \hat{A}_n^{\pi}(s_t, a_t),\ w_n \propto \lambda^{n-1} \end{aligned} \tag{4.7}$$

Lecture 5　Value Function Methods

在 Actor-critic 算法中，仍然保留了 policy gradient 步骤．考虑每一步直接选择$\hat{A}^{\pi}$最大的 action. 即

$$\pi'(a_t|s_t)=\begin{cases}1, \text{ if } a_t=\operatorname{argmax}_{a_t} A^{\pi}(s_t,a_t) \\ 0, \text{ else}\end{cases}. \tag{5.1}$$

或写作$\pi'(s)=a$.

Policy Iteration with Dynamic Programming

1. $V^{\pi}(s)\leftarrow r(s,\pi(s))+\gamma\mathbb{E}_{s'\sim p(s'|s,\pi(s))}[V^{\pi}(s')]$.
2. Set $\pi\leftarrow\pi'$.
3. Return to step 1.

可以用 tensor $\mathcal{T}$ (shape = # state × # state × # actions)记录 DP 状态.

或者也可以不用 policy，直接用 value iteration. (这里用Q^{π}替代A^{π}以简化)

Value Iteration with Dynamic Programming

1. $Q(s,a)\leftarrow r(s,a)+\gamma\mathbb{E}[V(s')]$.
2. Set $V(s)\leftarrow\max_a Q(s,a)$.
3. Return to step 1.

- Fitted Q-iteration: 在不需要学习 policy 的条件下优化(fitted value iteration)

Fitted Q-iteration

1. Collect dataset $\{(s_i,a_i,s_i',r_i)\}$ using specific policy.
2. Set $y_i\leftarrow r(s_i,a_i)+\gamma\cdot\max_{a_i'}Q_{\phi}(s_i',a_i')$.
3. Set $\phi\leftarrow\operatorname{argmin}_{\phi}\left(\frac{1}{2}\sum_i\|Q_{\phi}(s_i,a_i)-y_i\|^2\right)$.
4. Repeat from step 2 for K times.

Q-learning 与 Q-iteration 的核心是接近的，只需将上面 step 3 修改为

$$\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi(s_i,a_i)}{d\phi}\left[Q_\phi(s_i,a_i) - y_i\right] \tag{5.2}$$

将贪心算法(5.1)得到的 policy 和 Q 记作π^*, Q^*. 考虑到可能陷入 local maximum, 将其修改为ϵ-greedy search 算法

$$\pi(a_t|s_t) = \begin{cases} 1-\epsilon, & a_t = \mathrm{argmax}_{a_t} Q_\phi(s_t,a_t) \\ \dfrac{\epsilon}{|\mathcal{A}|-1}, & \text{otherwise} \end{cases}. \tag{5.3}$$

- ✧ 需要注意的是, Q-learning 并非一个 gradient descent 过程(因为并不是对 target function 的梯度下降). 因此 Q-learning, fitted Q-iteration 都不保证一定收敛.
- ✧ Value iteration 是一个收敛过程, 而 fitted value iteration 不一定收敛.(证明略)

Deep Q-learning Network(DQN) with replay buffer

DQN with Buffer & Target Network

1. Save target network parameters: $\phi' \leftarrow \phi$

Repeat N **times: {**

2. Collect dataset $\{(s_i, a_i, s_i', r_i)\}$ using specific policy, adding to buffer $\mathcal{B}$

Repeat K **times: {**

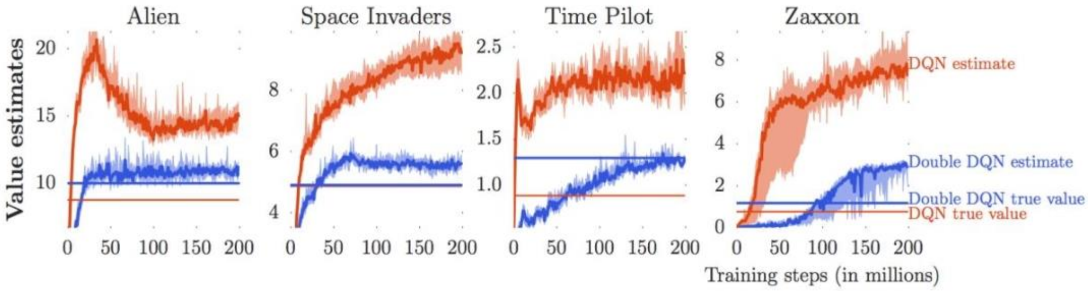
3. Sample a batch (s_j, a_j, s_j', r_j) from $\mathcal{B}$.

4. Set $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi(s_j,a_j)}{d\phi}\left[Q_\phi(s_j,a_j) - \left[r(s_j,a_j) + \gamma \max_{a_j'} Q_{\phi'}(s_j', a_j')\right]\right]$.

}}

5. Repeat from step 1.

- Q-learning 中存在的 over-estimation 问题:



有高估 Q function 的倾向. 解释: $\mathbb{E}[\max(X_1,\ldots,X_n)] \geq \max(\mathbb{E}[X_1],\ldots,\mathbb{E}[X_n])$.

因此 step 4.中$\max_{a'} Q_{\phi'}(s', a')$会高估 next value.

- 解决思路：解耦. 注意到

$$\max_{a'} Q_{\phi'}(s', a') = Q_{\phi'}(s',\ \mathrm{argmax}_{a'} Q_{\phi'}(s', a')) \tag{5.4}$$

所以采用 double Q-learning 可以解决.

$$\begin{aligned} Q_{\phi_A}(s,a) &\leftarrow r + \gamma Q_{\phi_B}\left(s',\ \mathrm{argmax}_{a'} Q_{\phi_A}(s', a')\right) \\ Q_{\phi_B}(s,a) &\leftarrow r + \gamma Q_{\phi_A}\left(s',\ \mathrm{argmax}_{a'} Q_{\phi_B}(s', a')\right) \end{aligned} \tag{5.5}$$

不需要额外设置两套 network，直接利用ϕ，ϕ'.

$$y = r(s,a) + \gamma Q_{\phi'}(s', \mathrm{argmax}_{a'} Q_{\phi}(s', a')) \tag{5.6}$$

这里虽然ϕ，ϕ'并不是完全 de-correlated，但已经可以缓解.

连续空间中的 Q-learning 应用:

$\max_{a'} Q_{\phi'}(s', a')$或$\mathrm{argmax}_{a'} Q_{\phi'}(s', a')$的计算会很困难.

解决思路:

1. 添加一个 inner loop，用随机梯度下降 SGD 进行
2. 简单随机采样，化连续为离散$\max_a Q(s,a) \approx \max\{Q(s,a_1), ..., Q(s,a_N)\}$
3. 学习一个 approximate maximizer:

$$\mu_\theta(s) \approx \mathrm{argmax}_a Q(s,a) \tag{5.7}$$

DDPG Algorithm

1. Collect dataset $\{(s_i, a_i, s_i{}', r_i)\}$ using specific policy, adding to buffer $\mathcal{B}$
2. Sample a batch $(s_j, a_j, s_j{}', r_j)$ from $\mathcal{B}$.
3. Compute $y_j = r(s_j, a_j) + \gamma Q_{\phi'}(s_j{}', \mu_{\theta'}(s_j{}'))$ with target net $Q_{\phi'}$ and $\mu_{\theta'}$.
4. Set $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi(s_j, a_j)}{d\phi} [Q_\phi(s_j, a_j) - y_j]$.
5. Set $\theta \leftarrow \theta + \beta \sum_j \frac{d\mu(s_j)}{d\theta} \frac{dQ_\phi(s_j, \mu(s_j))}{da}$
6. Update ϕ', θ' (e.g., Polyak averaging) and repeat from step 1.

Lecture 6　Advanced Policy Gradient

首先证明一个结论:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right]. \tag{6.1}$$

证明:

$$\begin{aligned}
J(\theta') - J(\theta) &= J(\theta') - \mathbb{E}_{s_0 \sim p(s_0)}[V^{\pi_\theta}(s_0)] \\
&= J(\theta') - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}[V^{\pi_\theta}(s_0)] \\
&= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(s_t)\right] \\
&= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t [r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)]\right] \\
&= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]
\end{aligned}$$

其中第二行是由于无论对θ或θ'，在s_0处的边际分布相同.

利用 importance sampling 思想，于是

$$\begin{aligned}
J(\theta') - J(\theta) &= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t \sim \pi_{\theta'}(a_t|s_t)}[\gamma^t A^{\pi_\theta}(s_t, a_t)]\right] \\
&= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]\right]
\end{aligned} \tag{6.2}$$

类似于式(1.5)给出的证明，当$\pi_{\theta'}$与π_θ接近时，$|p_{\theta'}(s_t) - p_\theta(s_t)| \leqslant 2\epsilon t$. 这时

$$\begin{aligned}
J(\theta') - J(\theta) &\approx \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}\left[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]\right] \\
&= \bar{A}(\theta')
\end{aligned} \tag{6.3}$$

可以利用$\theta' \leftarrow \mathrm{argmax}_{\theta'} \bar{A}(\theta)$优化$\pi'$.

构造优化函数

$$\mathcal{L}(\theta', \lambda) = \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}\left[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]\right] - \lambda[D_{KL}(\pi_{\theta'}(a_t|s_t) || \pi_\theta(a_t|s_t)) - \epsilon] \tag{6.4}$$

Dual Gradient Descent
1. Maximize $\mathcal{L}$ with respect to θ'.
2. $\lambda \leftarrow \lambda + \alpha[D_{KL}(\pi_{\theta'}(a_t|s_t) || \pi_\theta(a_t|s_t)) - \epsilon]$.

Lecture 7　Optimal control & planning

Closed-loop & open loop: 闭环指的是每一步都观察s_t，然后做出反应a_t. 而开环指的是一开始就根据s_1做出一系列反应.

下面考虑开环的问题. 可以概括为

$$\boldsymbol{A} = \mathrm{argmax}_{\boldsymbol{A}} J(\boldsymbol{A}) \tag{7.1}$$

$$\boldsymbol{A} = a_1, \ldots a_T \tag{7.2}$$

采用随机优化(stochastic optimization)方法:

- 连续分布情况下采样时考虑 cross-entropy method(CEM)

Cross-entropy Method
1. Sample $\boldsymbol{A}_1, \ldots \boldsymbol{A}_N$ from $p(\boldsymbol{A})$.
2. Evaluate $J(\boldsymbol{A}_1), \ldots J(\boldsymbol{A}_N)$.
3. Pick the elites $\boldsymbol{A}_{i_1}, \ldots, \boldsymbol{A}_{i_M}$ and refit $p(\boldsymbol{A})$.

- 离散情况下采用 Monte Carlo tree search(MCTS).
  具体的树算法比较多. 核心是选择最大 reward 的树节点，同时探索未曾访问的结点.

Linear-quadratic regulator:
在优化

$$\min_{u_1,\ldots,u_T} \left[c(x_1,u_1) + c(f(x_1,u_1),u_2) + \ldots + c(\ldots,u_T) \right] \tag{7.3}$$

时考虑线性情况，即

$$f(x_t,u_t) = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_t \tag{7.4}$$

$$c(x_t,u_t) = \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T c_t \tag{7.5}$$

具体推导过程略. 可以利用回溯方法得到各步最优的u_t.(详见课件)

- 对非线性情况，可以利用 iterative LQR 或 DDP 方法.(略)

隐空间模型(latent space model)
在 fully observed 空间中，需要优化的是

$$\max_\phi \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\phi(s_{t+1,i} | s_{t,i}, a_{t,i}) \tag{7.6}$$

而在 latent space model 中则修改为

$$\max_{\phi,\psi}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}\left[\log p_{\phi}(s_{t+1,i}|s_{t,i},a_{t,i})+\log p_{\phi}(o_{t,i}|s_{t,i})+\log p_{\phi}(r_{t,i}|s_{t,i})\right] \tag{7.7}$$

这里期望是由于状态s原则上不是完全确定的. 因此可以写为

$$s_t \sim q_{\psi}(s_t|o_t),\ s_{t+1}\sim q_{\psi}(s_{t+1}|o_{t+1}) \tag{7.8}$$

即$s_t=g_{\psi}(o_t)$.

Model-based RL with Latent State

1. Run base policy $\pi_0(a_t|o_t)$ to collect $\mathcal{D}=\{(o,a,o')_i\}$

Repeat N **times{**

2. Learn $p_{\phi}(s_{t+1}|s_t,a_t)$, $p_{\phi}(r_t|s_t)$, $p(o_t|s_t)$, $g_{\psi}(o_t)$.

Repeat M **times{**

3. Plan through the model to choose actions.
4. Execute the first planned action, observe o'.
5. Append (o,a,o') to $\mathcal{D}$.

}}

再考虑闭环的情况.

$$\pi=\mathrm{argmax}_{\pi}\,\mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{t}r(s_t,a_t)\right] \tag{7.9}$$

Model-based RL(closed loop)

1. Run base policy $\pi_0(a_t|o_t)$ to collect $\mathcal{D}=\{(o,a,o')_i\}$
2. Learn dynamics model $f(s,a)$ to minimize $\sum_{i}\|f(s_i,a_i)-s_i{}'\|^2$
3. Backpropagate through $f(s,a)$ into policy to optimize $\pi_{\theta}(a_t|o_t)$.
4. Run $\pi_{\theta}(a_t|o_t)$ and append to $\mathcal{D}$.
5. Repeat from step 2.

- 存在的问题: 过长的传导链容易导致 gradient vanishing/exploding.

- 解决方法: (1) 使用 model-free RL 方法. 如 policy gradient. (2) 使用简单 policy 替代 NN. 如 LQR.

Lecture 8　Exploration

如何判断可以得到一个最优的 exploration strategy:

- 1-step/stateless 问题可以归为 partial observable MDP 问题.
- Small, finite MDP 问题可以用 Bayesian model 处理
- Large/infinite MDP 问题没有最优方法.

第一类: 以老虎机(bandit)为例

首先定义衡量 exploration strategy 的指标 regret:

$$\mathrm{Reg}(T) = T\mathbb{E}[r(a^*)] - \sum_{t=1}^{T} r(a_t) \tag{8.1}$$

1. optimistic exploration 方法:

$$a = \mathrm{argmax}\, \hat{\mu}_a + \sqrt{\frac{2\ln T}{N(a)}} \tag{8.2}$$

   其中$\hat{\mu}_a$是 average reward，$N(a)$是采取a的次数. 后面这一项的目的是确保有 variance.

2. 若$r(a_i)$~$p_{\theta_i}(r_i)$，可以用 Thompson sampling.

$$\begin{gathered} \theta_1 \ldots \theta_n \sim \hat{p}(\theta_1 \ldots \theta_n) \\ a = \mathrm{argmax}_a \mathbb{E}_{\theta_a}[r(a)] \end{gathered} \tag{8.3}$$

3. Information gain(见下/讲义)

第二类: 有状态(MDP)，以吃豆人(Pacman)为例

1. 可以将 optimistic exploration 修改为 count-based. 添加 exploration bonus.

$$r^+(s,a) = r(s,a) + \mathcal{B}(N(s)) \tag{8.4}$$

   其中$\mathcal{B}$(bonus)是递减函数.

   但事实上 state 数目巨大，可以用$p_\theta(s)$代替$N(s)$.

Exploring with pseudo-counts

1. Fit model $p_\theta(s)$ to all states $\mathcal{D}$ seen so far.
2. Take new step and observe s_i.
3. Fit new model $p_{\theta'}(s)$ to $\mathcal{D} \cup s_i$.

4. Use $p_\theta(s)$ and $p_{\theta'}(s)$ to estimate $\hat{N}(s)$.

5. Set $r_i^+ = r_i + \mathcal{B}\big(\hat{N}(s)\big)$ and repeat from step 1.

其中 step 4.用到了$p_\theta(s_i) = \frac{\hat{N}(s_i)}{\hat{n}}$, $p_{\theta'}(s_i) = \frac{\hat{N}(s_i)+1}{\hat{n}+1}$. 进而可解出$\hat{N}(s_i)$. 式中$\hat{n}$是预计访问过的状态.

2. 对 Thompson sampling 的改进：考虑状态后，通过抽样不同的 Q-function 代替之前的仅考虑 reward 分布. (这里利用了 Q-learning 是 off-policy 的特点，不同的 Q-function 对抽样无影响) (详见 bootstrapped DQN paper, Osband)

- 以信息论的观点来看 exploration:

  目标即使得$p(G|S)$尽可能确定化，其中G是目标，S是 final state.

  借助熵的概念，即

$$\max[\mathcal{H}(p(G)) - \mathcal{H}(p(G|S))]. \tag{8.5}$$

  其中$\mathcal{H}$为熵.

Lecture 9　Inverse Reinforcement Learning

目标：通过观察推测 reward function.

即给定$s \in \mathcal{S}$, $a \in \mathcal{A}$，以及可能给到的$p(s'|s,a)$. 依据$\pi^*(\tau)$抽样得到$\{\tau_i\}$，进而学习$r_\psi(s,a)$. 也需要进一步学习$\pi^*(a|s)$.

- 如何学习 reward function?

1. 传统的线性方法可归结为

$$\max_{\psi,m} m \text{ s.t. } \psi^T \mathbb{E}_{\pi^*}[f(s,a)] \geqslant \max_{\pi \in \Pi} \psi^T \mathbb{E}_\pi[f(s,a)] + m \tag{9.1}$$

其中$r_\psi(s,a) = \psi^T f(s,a)$.

利用支持向量机(SVM)的思想以及考虑到需要让π, π^*尽量接近，上式可修改为

$$\min_\psi \frac{1}{2}\|\psi\|^2 \text{ s.t. } \psi^T \mathbb{E}_{\pi^*}[f(s,a)] \geqslant \max_{\pi \in \Pi} \psi^T \mathbb{E}_\pi[f(s,a)] + D(\pi,\pi^*). \tag{9.2}$$

其中D是差异函数.(数学过程见讲义)

2. 深度学习方法

假设$p(\mathcal{O}_t|s_t,a_t,\psi)=\exp(r_\psi(s_t,a_t))$. 根据 Bayes 公式可得

$$p(\tau|\mathcal{O}_{1:T},\psi)\propto\exp\left(\sum_t r_\psi(s_t,a_t)\right). \tag{9.3}$$

于是采用最大似然法(ML)，目标函数为

$$\max_\psi\frac{1}{N}\sum_{i=1}^{N}\log p(\tau|\mathcal{O}_{1:T},\psi)=\max_\psi\frac{1}{N}\sum_{i=1}^{N}r_\psi(\tau_i)-\log Z \tag{9.4}$$

这里尤其要注意$\log Z$项，本质上是 log normalizer 项，以避免模型给没有出现过的(expert policy 不采用的)路径赋予高 reward. 这里Z称为 partition function，形式为

$$Z=\int p(\tau)\exp(r_\psi(\tau))d\tau. \tag{9.5}$$

省略中间的数学推导(见讲义)，得到了 *MaxEnt* IRL 算法(见 paper).

- 换个角度看，上述过程目标是训练出 reward function，使 expert policy 与 trained(fake) policy 的 reward 区别尽可能大. 这一思想符合 GAN 的算法.(见 paper. Finn, Christiano)