本文主要基于各方法相关论文，第二部分出现的一些公式参考了 berkeley cs 294 课程，可参见 http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html 。

inverse reinforcement learning，又称为 Imitation learning 或者 learning form demonstration。常规的强化学习是从环境中得到 reward，利用其找到使累积 reward 的期望最大的策略。而很多情况下，我们没有即时回报或者回报特别稀疏，而此时如果有人的一些行为示例，我们可以反过来从正确的示例中学习回报函数，从而指导策略的学习。那么我们如何定义这个回报函数呢，最主要的思想就是**我们认为产生示例的专家策略是最优的，所以要学习这样一个回报函数，它使得示例在这个回报函数下是最优的，也就是利用该回报学习到的最优策略是可以匹配示例的轨迹的。在传统方法中，我们将问题变为，找到回报函数使得示例的累计回报的期望比其他任何策略都大。**

一．传统方法

1. 学徒学习

在常规的强化学习中，reward 是一个来自环境的值，我们可以根据经验学习 reward 在 action 和 state 条件下的概率模型 p(r|s,a)。而现在，我们将 reward 建模为一个在 state 空间或者 state-action 对上的函数，表示进入的状态\采取的 action 是不是好的。学徒学习定义了一个线性的 reward 函数：

$$R^*(s) = w^* \cdot \phi(s)$$

其中 Φ(s)是状态的特征，也可以是一组基函数。如果求出系数 w，那我们就得到了 reward 函数。这里限制 reward 是一个小于 1 的值，因此使 w 的 1 范数小于 1。

在这个 reward 下用初始状态的 value 值衡量一个 policy 的好坏：

$$\begin{aligned} E_{s_0\sim D}[V^\pi(s_0)] &= E[\sum_{t=0}^{\infty}\gamma^t R(s_t)|\pi] \\ &= E[\sum_{t=0}^{\infty}\gamma^t w\cdot\phi(s_t)|\pi] \\ &= w\cdot E[\sum_{t=0}^{\infty}\gamma^t\phi(s_t)|\pi] \end{aligned}$$

其中 E 表示在 policy π 和环境的状态转移概率下的期望。定义累积的状态特征的期望值为 μ(π)：

$$\mu(\pi) = E[\sum_{t=0}^{\infty}\gamma^t\phi(s_t)|\pi] \in \mathbb{R}^k$$

则 policy 的 value 函数可以写成以下两部分，则衡量一个 policy 的好坏，只取决于参数 w：

$$E_{s_0\sim D}[V^\pi(s_0)] = w\cdot\mu(\pi)$$

如果我们有一些示例 $\{s_0^{(i)}, s_1^{(i)}, \ldots\}_{i=1}^m$， 利用这 m 个示例中的轨迹，可以得到 π_E 的经验特征期望：

$$\hat{\mu}_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi(s_t^{(i)})$$

假设示例来自于最优的 policy π_E，则要找到合适的参数 w，使得在此 reward 下的最优策略是 π_E。为了达到这个目的，可以寻找使 π_E 和其他 policy 的表现差距最大的 w，找到这个 w 后，我们可以反过来学习新的 policy。最终的目的是要找到和 π_E 特征期望几乎相同的 policy：

$$\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \epsilon$$

从而这两个策略的表现也几乎相同：

$$
\begin{aligned}
&|E[\textstyle\sum_{t=0}^{\infty}\gamma^t R(s_t)|\pi_E] - E[\sum_{t=0}^{\infty}\gamma^t R(s_t)|\tilde{\pi}]| \\
&\quad = \quad |w^T\mu(\tilde{\pi}) - w^T\mu_E| \\
&\quad \leq \quad \|w\|_2\|\mu(\tilde{\pi}) - \mu_E\|_2 \\
&\quad \leq \quad 1\cdot\epsilon = \epsilon
\end{aligned}
$$

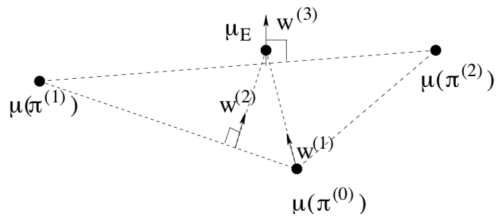
不等式的意思是两个 μ 差距在 w 上的投影，一定小于这两个 μ 的差距。
整个算法的步骤如下：

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = \max_{w:\|w\|_2\leq 1} \min_{j\in\{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.
3. If $t^{(i)} \leq \epsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T\phi$.
5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and go back to step 2.

其中第二步的优化目标可以类比 SVM 最大化最小间隔，不同的是 SVM 是对已有的正负样本点分割，而这里是有一个固定的正样本。不断产生新的负样本。这个目标可以写成如下标准的优化形式 ：

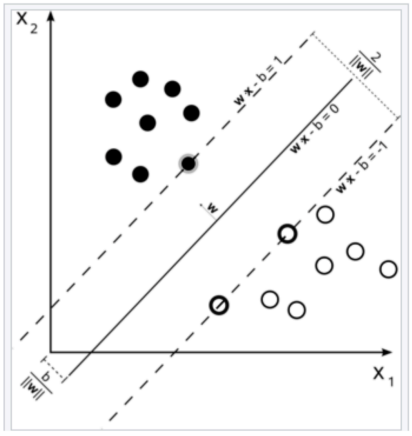
$$
\begin{aligned}
\max_{t,w} \quad & t \\
\text{s.t.} \quad & w^T\mu_E \geq w^T\mu^{(j)} + t,\ j = 0,\ldots,i-1 \\
& ||w||_2 \leq 1
\end{aligned}
$$

该算法可以展示为下图：



2. Maximum Margin Planning

不同于学徒学习，MMP 方法是一种在策略空间上进行结构边际最大化的方法，我们类比 SVM 进行解释：



如图，SVM 的优化目标为

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

假设两个支撑点分别为 x1，x2，函数 f(x)=wx+b，则|f(x1)-f(x2)|\||w||为这两个点在 w 方向的距离 γ。为了使这个距离最大，用学徒学习方法中的符号表示为：

$$\max_{w} \gamma \quad s.t. \frac{w\mu_E - w\mu}{\|w\|} \geq \gamma$$

类比 svm 中几何间隔和函数间隔的区别，我们可以将间隔 γ 写成 γ-\||w||，则：

$$\max_{w} \frac{\gamma^-}{\|w\|} \quad s.t.\ w\mu_E - w\mu \geq \gamma^- \tag{1}$$

在学徒学习的优化公式中 γ-即为 t。由于约束了 w 的范数小于 1，因此优化目标只要求最大化 t 即可。学徒学习通过不断在新的 reward 下正向学习新的策略，然后加入“负例”，而 MMP 方法则是直接在策略空间中进行学习，不用迭代的进行强化学习产生新的策略。

MMP 方法新定义了如下概念：

F 矩阵 d*|X||A|是一个特征基地矩阵，为每一个 state-action 对保存了一个特征向量。

y 是某个 policy 产生的轨迹，也可以用一个矩阵 μ 保存，维度为|X||A|，记录这个 policy 下 state-action 对出现的频率。可以经验的认为是对策略的描述。

f(y)=Fμ，等同于学徒学习方法中的特征期望。

Loss 函数$\mathcal{L}(y, y_i) = \mathcal{L}_i(y) = l_i^T \mu$.定义了 policy y 与产生示例 y_i 的最优 policy 的“距离”。比如可以使用两个 policy 中状态不一致的数量。

知道这些概念后，我们给出 MMP 方法的目标：找到这样的线性回报函数，使得在这个回报函数下最优的 policy$\mu^* = \arg\max_{\mu \in \mathcal{G}_i} w^T F_i \mu$，离产生示例的 policy 距离最近。可以看到，比起学徒学习假设存在一个可以得到专家 policy 的回报函数，这是一个更松弛的假设条件。

为了实现这个目标，MMP 使用了结构最大边际框架，通过使一个正比于距离的间隔，使 yi 优于其他的 policy。在(1)式中，γ-可以简单的设置为 1，这样我

们的优化目标为：

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + 1 \quad \forall\pi$$

由于我们现在有了“距离”的概念，我们认为对于和示例 policy 区别大的 policy，间隔应该更大：

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + m(\pi^*, \pi) \quad \forall\pi$$

最后加上松弛变量，并且推广到多个专家示例（可以来自不同的 MDP）：

$$\min_{w,\zeta_i} \frac{1}{2}\|w\|^2 + \frac{\gamma}{n}\sum_i \beta_i\zeta_i^q$$
$$s.t. \ \forall i \quad w^T F_i\mu_i + \zeta_i \geq \max_{\mu\in\mathcal{G}_i} w^T F_i\mu + l_i^T\mu$$

同时要对 policy μ 施加一个 bellman 流动约束，即某个状态的流入流出的频次应该相等：

$$\sum_{x,a}\mu^{x,a}p_i\left(x'|x,a\right) + s_i^{x'} = \sum_a \mu^{x',a}$$

然后通过一些变换，最终得到一个二次规划的问题，具体的推导过程参见 https://zhuanlan.zhihu.com/p/26766494。关于这两种方法更详细的比较参见《回报函数学习的学徒学习综述》。

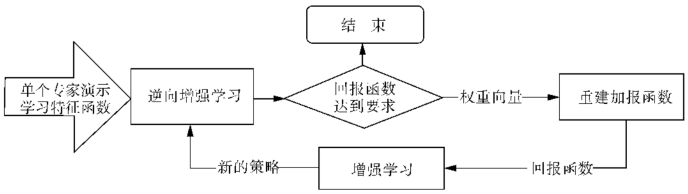


图 1 基于逆向增强学习的学徒学习

Fig. 1 Apprenticeship learning based on IRL

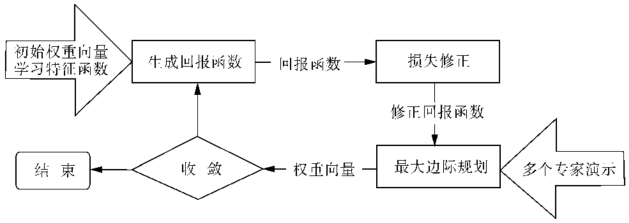


图 2 基于 MMP 的学徒学习

Fig. 2 Apprenticeship learning based on MMP

3. 歧义问题和专家最优假设

这些比较传统的方法都存在的缺点是无法解决歧义问题，具体来说：

1. 对于已知回报函数，我们一定能够得到一个最优 policy。而由于回报函数是未知的，我们需要通过示例反过来学习 reward 函数，**有多个 reward 函数都可以使示例是最优的，比如如果 reward 都是 0，任何的 policy 都是最优的；此外不同的 policy 得到的轨迹的特征期望可能是相同的**。
2. 使用最大边际原则找一个唯一的回报函数，我们可能找不到一个合适的 reward， 使得产生示例的 policy 与其他 policy 间隔最大，同时又是最优 policy。

歧义性问题源自我们假设了专家的行为是最优的。而专家 policy 本身（在任何 reward 下）就不是最优的，而我们通过 reward 函数正向的学习到的 policy 产生的轨迹都是最优的，这时可能需要一些 policy 的混合，才能产生示例中的行为；然而，有多种 policy 的混合方法都能使得特征期望匹配。

二．基于最大熵的逆强化学习

1. 最大熵逆强化学习

复习一下上一节的概念，我们通过学习一个从 state 到 reward 的映射，使得示例优于其他 policy 产生的轨迹，这是逆向学习的步骤；利用学到的 reward，正向学习 policy，最后使得 policy 可以匹配示例，而匹配这个概念则是由 policy 产生的轨迹和示例的特征期望相同得到的。

其中 reward 是定义在 state 特征空间上的函数，而一条轨迹的 reward 则是 reward 函数作用在轨迹特征期望上：

$$\text{reward}(\mathbf{f}_\zeta) = \theta^\top \mathbf{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \mathbf{f}_{s_j}$$

示例的经验特征为：

$$\tilde{\mathbf{f}} = \frac{1}{m} \sum_i \mathbf{f}_{\tilde{\zeta}_i}$$

则我们要学到一个 policy，可以产生满足下面约束的轨迹的分布：

$$\sum_{\text{Path } \zeta_i} P(\zeta_i) \mathbf{f}_{\zeta_i} = \tilde{\mathbf{f}}$$

为了解决歧义性问题，我们引入最大熵方法。最大熵原理是，在所有满足约束的分布里面，选择熵最大的分布。这样模型中除了约束条件是已知的之外，没有做任何主观的假设。如果示例是次优的，则会存在多个轨迹的分布使得特征期望与示例的经验特征期望相匹配，其中的任何分布都可能对一些轨迹存在偏好，而这些偏好是不存在于约束条件中的，根据最大熵原理，我们得到如下约束最优化问题：

$$\begin{aligned} &\max -p \log p \\ s.t. &\sum_{Path\zeta_i} P(\zeta_i) f_{\zeta_i} = \tilde{f} \\ &\Sigma P = 1 \end{aligned}$$

可以得到拉格朗日函数：

$$L = \sum_{\zeta_i} p \log p - \sum_{j=1}^{n} \lambda_j \left(p f_j - \tilde{f}_j \right) - \lambda_0 \left(\Sigma p - 1 \right)$$

令拉格朗日函数对 p 的导数为 0：

$$\frac{\partial L}{\partial p} = \sum_{\zeta_i} \log p + 1 - \sum_{j=1}^{n} \lambda_j f_j - \lambda_0 = 0$$

使 L 最小的 p 满足：

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^\top \mathbf{f}_{s_j}} \tag{2.1}$$

用示例对该模型做最大似然即可（L 是 p 的凸函数，原始问题的最优解和对偶问题的最优解相等，对偶问题是个极大极小问题，求解内部的极小问题得到上式轨迹的分布，这个分布取决于拉格朗日参数，称为最大熵模型；可以证明，最大化对偶函数等价于最大熵模型的最大似然估计）；

2.1 是定义在确定性 MDP 下的模型，直观上看，这个模型说明了有相同回报的轨迹出现的概率相同，高回报的轨迹出现的概率指数级的增长；

对于 MDP 具有随机性的情况，路径的分布可以近似为：

$$\frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta,T)} \prod_{s_{t+1},a_t,s_{t\in\zeta}} P_T(s_{t+1}|a_t,s_t)$$

这个模型与正向的加入熵正则的强化学习中，轨迹的概率是相同的：

$$p(\tau|\mathcal{O}_{1:T}) = \frac{p(\tau,\mathcal{O}_{1:T})}{p(\mathcal{O}_{1:T})} \propto p(\tau)\prod_t \exp(r(\mathbf{s}_t,\mathbf{a}_t)) = p(\tau)\exp\left(\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right) \tag{2.2}$$

但是在正向学习过程中，reward 是已知的（或者可以从环境中获得），我们根据 reward 学习 policy：

$$\pi_Q^{\mathcal{B}}(a \mid s) = \overline{\pi}(a \mid s)\exp((Q(s,a) - V_Q(s))/\tau)$$

这里 Q 即为上式中 reward 之和。

在逆向学习过程中，该形式是一个取决于参数 θ 的模型，我们不知道 reward，但是我们有一些示例轨迹。我们认为这些示例的行为是近似最优的，专家想要表现的最优，但是有概率产生次优的行为。我们利用这些示例轨迹对模型做最大似然，来学习 reward 的参数。

现在我们具体来看最大似然的步骤是如何计算梯度的，如果有 N 个示例样本，则最大化对数似然函数：

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

轨迹分布代入(2.2)，则得到：

$$\max_\psi \frac{1}{N}\sum_{i=1}^N r_\psi(\tau_i) - \log Z$$

$$Z = \int p(\tau)\exp(r_\psi(\tau))d\tau$$

Z 是 partition function，是一个和参数 ψ 有关的函数。整个式子对参数求梯度得到：

$$\nabla_\psi \mathcal{L} = \frac{1}{N}\sum_{i=1}^N \nabla_\psi r_\psi(\tau_i) - \underbrace{\frac{1}{Z}\int p(\tau)\exp(r_\psi(\tau))\nabla_\psi r_\psi(\tau)d\tau}_{p(\tau|\mathcal{O}_{1:T},\psi)}$$

如果我们把第一项也当做期望的话，最终就得到了下式：

$$\nabla_\psi \mathcal{L} = E_{\tau\sim\pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - E_{\tau\sim p(\tau|\mathcal{O}_{1:T},\psi)}[\nabla_\psi r_\psi(\tau)]$$

第一部分是对于示例（假设来自于最优 policy）的回报关于 ψ 的梯度，第二部分是 ψ 确定的 reward 下，最优 policy 产生的轨迹的回报关于 ψ 的梯度，也就是是说，我们更新梯度使得示例的回报更高，同时使得我们在训练的 policy 回报更低。当 reward 是个线性函数时，∇r 正是轨迹的特征，当这个梯度为 0 时，意味着示例的经验特征匹配了轨迹的特征期望。

将轨迹按照时间拆开：

$$\mathbf{E}_{\tau\sim p(\tau|\mathcal{O}_{1:T},\psi)}\left[\nabla_\psi \sum_{t=1}^{T} r_\psi(\mathbf{s}_t,\mathbf{a}_t)\right]=\sum_{t=1}^{T}\mathbf{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim p(\mathbf{s}_t,\mathbf{a}_t|\mathcal{O}_{1:T},\psi)}[\nabla_\psi r_\psi(\mathbf{s}_t,\mathbf{a}_t)]$$

其中状态-动作分布为：

$$p(\mathbf{s}_t,\mathbf{a}_t|\mathcal{O}_{1:T},\psi)=p(\mathbf{a}_t|\mathbf{s}_t,\mathcal{O}_{1:T},\psi)p(\mathbf{s}_t|\mathcal{O}_{1:T},\psi)$$

$$p(\mathbf{s}_t,\mathbf{a}_t|\mathcal{O}_{1:T},\psi)\propto\beta(\mathbf{s}_t,\mathbf{a}_t)\alpha(\mathbf{s}_t) \tag{2.3}$$

整个训练过程就是训练 reward—在 reward 下用动态规划学习软最优 policy 的过程，这个模型存在的问题是，每当参数 ψ 更新一次，都要重新用前向后向算法进行正向的强化学习过程。

2. guided cost learning---基于重要性采样的方法

最大熵逆强化学习要求 MDP 已知，并且 state，action 有限。如何将其推广到更一般化的问题中呢？

计算 partition function 要求我们可以得到当前 reward 下最优 soft policy 得到所有轨迹的分布。Guided cost learning 提出了一种基于采样的方法，就算我们不知道系统的动态，但是可以用学习到的策略去采样轨迹。具体来说，在逆向学习的部分使用基于采样的最大熵逆最优控制，而采样的分布并不是当前 reward 下的最优 policy（无模型的最优控制计算量要远远大于有模型的，如果 reward 的参数走一个梯度步，就要重新学习 policy，代价非常高），而是一个不断用当前的 reward 学习去接近 $p(\tau)\exp\left(\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right)$ 的分布（而不是学习到最优）。

a) 基于采样的逆最优控制

$$\begin{aligned}L&=\frac{1}{N}\sum_{i=1}^{N}r_\psi(\tau_i)-\log\int\exp(r_\psi(\tau))d\tau\\&=\frac{1}{N}\sum_{i=1}^{N}r_\psi(\tau_i)-\log\frac{1}{M}\sum_{j=1}^{M}\frac{\exp(r_\psi(\tau_j))}{q(\tau_j)}\end{aligned} \tag{2.4}$$

将上式中 $\frac{\exp(r_\psi(\tau_j))}{q(\tau_j)}$ 记为 w_j，则由该目标得到梯度：

$$\nabla_\psi\mathcal{L}\approx\frac{1}{N}\sum_{i=1}^{N}\nabla_\psi r_\psi(\tau_i)-\frac{1}{\sum_j w_j}\sum_{j=1}^{M}w_j\nabla_\psi r_\psi(\tau_j)$$

为了计算梯度，需要计算 w_j，当系统具有随机性时：

$$w_j=\frac{p(\mathbf{s}_1)\prod_t p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\exp(r_\psi(\mathbf{s}_t,\mathbf{a}_t))}{p(\mathbf{s}_1)\prod_t p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t)}=\frac{\exp(\sum_t r_\psi(\mathbf{s}_t,\mathbf{a}_t))}{\prod_t\pi(\mathbf{a}_t|\mathbf{s}_t)}$$

当系统不具有随机性时，相当于 p(s'|s,a)=1。则得到同样的结果。

b) 通过策略优化提升采样

理论上采样可以来自任何分布，11 年有篇工作 relative entropy inverse reinforcement learning 使用了均匀分布。但我们知道用重要性采样估计 f(x)在 p(x)

下的期望，最好的重要性采样分布是 $q(x) \propto |f(x)|p(x)$，估计 partition function 最好的策略是 $\pi(\tau) \propto \exp(r_\psi(\tau))$，也就是最大熵最优策略；这意味着如果 q(τ)可以更多地覆盖回报高的区域，则对 partition function 的评估的偏差更小，所以这个采样分布可以利用当前 reward function 不断进行提升。

具体来说，我们要找到这样的 q(τ)使得下式最小：

$$KL(q(\tau)||\exp(r_\psi(\tau))) = E_q[-r_\psi(\tau)] - H(\tau)$$

而这正是最大熵强化学习的目标，当该算法收敛时，q(τ)正比于 $\exp(r_\psi(\tau))$。guided cost 的意思就是引导采样向 reward 更高的区域。

正向的强化学习方法见第三部分，soft q learning 和 policy gradient 都是可以的。由于本文的目的是未知系统动态的逆最优控制，所以选择一个基于采样进行策略优化的方法即可。

c) 重要性权重

本文使用了 NN 来建立 reward function，一般来说，当参数维度很高的时候我们会考虑使用随机梯度下降，但是对 partition function，如果从样本中取一个小的 batch，发现会 unbounded，这是因为相当于只从 q(τ)采样了一个很小的样本，这些样本的 reward 可能非常小，导致(2.4)分子部分几乎为 0。

本文实验发现，如果从示例中采样一些样本加入(2.4)式中第二部分的采样，效果会好很多，也可以使用随机梯度方法。此外每次更新 policy 后，并不抛弃掉原来的采样，而是保留每一次的采样。此时计算重要性权重不能仅仅使用当前 policy 来计算，而是使用一个融合的分布。

如果从多个分布中采样，去估计这些采样下 f(x)的值，那么可以构建一个统一的分布，在这个分布下 f(x)期望是与之前的估计值是一致的：

$$E[f(\tau)] \approx \frac{1}{M} \sum_{\tau_j} \frac{1}{\frac{1}{k}\sum_\kappa q_\kappa(\tau_j)} f(\tau_j)$$

最后，loss 改为如下形式：

$$\mathcal{L}_{\text{IOC}}(\theta) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} z_j \exp(-c_\theta(\tau_j))$$

其中 $z_j = [\frac{1}{k}\sum_\kappa q_\kappa(\tau_j)]^{-1}$

对于前面每次迭代时得到的采样，保留 policy 每次的参数即可，而对于示例的分布，本文是建立了一个高斯分布，经验的计算示例中的均值和方差。最后算法如下：

Algorithm 1 Guided cost learning

1: Initialize $q_k(\tau)$ as either a random initial controller or from demonstrations
2: **for** iteration $i = 1$ to I **do**
3: Generate samples $\mathcal{D}_{\text{traj}}$ from $q_k(\tau)$
4: Append samples: $\mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \mathcal{D}_{\text{traj}}$
5: Use $\mathcal{D}_{\text{samp}}$ to update cost c_θ using Algorithm 2
6: Update $q_k(\tau)$ using $\mathcal{D}_{\text{traj}}$ and the method from (Levine & Abbeel, 2014) to obtain $q_{k+1}(\tau)$
7: **end for**
8: **return** optimized cost parameters θ and trajectory distribution $q(\tau)$

Algorithm 2 Nonlinear IOC with stochastic gradients

1: **for** iteration $k = 1$ to K **do**
2: Sample demonstration batch $\hat{\mathcal{D}}_{\text{demo}} \subset \mathcal{D}_{\text{demo}}$
3: Sample background batch $\hat{\mathcal{D}}_{\text{samp}} \subset \mathcal{D}_{\text{samp}}$
4: Append demonstration batch to sample batch: $\hat{\mathcal{D}}_{\text{samp}} \leftarrow \hat{\mathcal{D}}_{\text{demo}} \cup \hat{\mathcal{D}}_{\text{samp}}$
5: Estimate $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$ using $\hat{\mathcal{D}}_{\text{demo}}$ and $\hat{\mathcal{D}}_{\text{samp}}$
6: Update parameters θ using gradient $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$
7: **end for**
8: **return** optimized cost parameters θ

3. connection between GAN, IRL and energy-based models

a) 最大熵 IRL 与能量模型

基于最大熵的 IRL 实际上是一个 energy-based model，这种模型为每个数据点 x 关联一个能量函数 $E_\theta(x)$，将整个数据建模为一个 boltzmann 分布：

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp(-E_\theta(\mathbf{x}))$$

参数 θ 一般通过对数据集最大似然得到，这样优化的难点在于，要评估 partition function Z 我们需要在整个 x 的空间做求和或者求积分，这在 x 的维度很高的时候是很困难的。一个常见的评估 Z 的方法是从 $p_\theta(x)$中得到采样。在 IRL 中，数据是在以 r_θ 为 reward 的 MDP 中产生的轨迹 τ，利用 $p_\theta(\tau)$我们可以很方便的求某个轨迹的概率密度，但是如果要用这个分布求积分（原始的最大熵 IRL）或者对其进行采样，我们转而学习在这个 MDP 下的最优策略，采样时可用这个策略和环境互动采样到 τ。

b) Guided cost learning 等价于特定形式的 GAN

GCL 同时训练一个 reward function 和一个用来产生采样的 policy，两部分的 loss 分别为：

$$\mathcal{L}_{\text{sampler}}(q) = \mathbb{E}_{\tau\sim q}[c_\theta(\tau)] + \mathbb{E}_{\tau\sim q}[\log q(\tau)]$$

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2}\tilde{p}(\tau)+\frac{1}{2}q(\tau)}\right]\right)$$

记 $\mu = \frac{1}{2}p + \frac{1}{2}q$，在 GCL 中用 μ 进行重要性采样。$\tilde{p}(\tau)$是对示例分布粗略的评估，GCL 是直接假设示例服从高斯分布，然后经验的统计均值和方差。而这里我们对 GCL 进行一个小的修改：选择使用当前的 $p_\theta(\tau)$作为对示例分布的估计，注意这不同于 q(τ)，q(τ)来自一个学习中的用来生成采样的 policy，学习的目标是 $KL(q(\tau)||p_\theta(\tau))$。

GAN 的判别器相当于一个分类器，损失是交叉熵误差：

$$\mathcal{L}_{\text{discriminator}}(D) = \mathbb{E}_{\mathbf{x}\sim p}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim G}[-\log(1-D(\mathbf{x}))]$$

生成器的原始目标是判别器损失的负。

这里对 GAN 进行了两个修改：

1. 修改生成器的 loss，由两部分组成：

$$\mathcal{L}_{\text{generator}}(G) = \mathbb{E}_{\mathbf{x}\sim G}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim G}[\log(1-D(\mathbf{x}))].$$

2. 用一个特定形式的判别器

我们知道，当固定生成器（相当于固定生成分布 q(τ)），最优判别器为：

$$D^*(\tau) = \frac{p(\tau)}{p(\tau)+q(\tau)}$$

我们用这个形式来估计真实分布：

$$D_\theta(\tau) = \frac{\tilde{p}_\theta(\tau)}{\tilde{p}_\theta(\tau) + q(\tau)}$$

为了和最大熵 IRL 产生联系，同样用 boltzmann 分布来建模真实数据分布；

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\frac{1}{Z}\exp(-c_\theta(\tau)) + q(\tau)}$$

此时判别器是一个关于 θ 的函数，q(τ)是预先计算好的；可以看到，该形式很像一个二分类，输入 τ，先经过网络 C_θ，然后将输出减掉数值 log q(τ)，然后再传到一个 sigmoid 层，sigmoid 有 logZ 的偏置。用原来的判别器 loss 优化 θ，当 $\frac{1}{Z}\exp(-c_\theta(\tau)) = p(\tau)$时，判别器达到最优。

修改了 GAN 和 IRL 模型后，证明这两者的等价性：
判别器的损失为：

$$\begin{aligned}\mathcal{L}_{\text{discriminator}}(D_\theta) &= \mathbb{E}_{\tau\sim p}[-\log D_\theta(\tau)] + \mathbb{E}_{\tau\sim q}[-\log(1 - D_\theta(\tau))] \\ &= \mathbb{E}_{\tau\sim p}\left[-\log \frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\frac{1}{Z}\exp(-c_\theta(\tau)) + q(\tau)}\right] + \mathbb{E}_{\tau\sim q}\left[-\log \frac{q(\tau)}{\frac{1}{Z}\exp(-c_\theta(\tau)) + q(\tau)}\right]\end{aligned}$$

最大熵 IRL 中，log 似然为：

$$\begin{aligned}\mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim \frac{1}{2}p+\frac{1}{2}q}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)}\right]\right) \\ &= \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim \mu}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2Z}\exp(-c_\theta(\tau)) + \frac{1}{2}q(\tau)}\right]\right)\end{aligned}$$

1. 能够 minimize 判别器 loss 的 Z 的值，正是对 partition function 的重要性采样评估：

$$\begin{aligned}&\partial_Z \mathcal{L}_{\text{discriminator}}(D_\theta) = 0 \\ &\frac{1}{Z} = \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z^2}\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)}\right] \\ &Z = \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)}\right].\end{aligned}$$

2. 两个 loss 对参数 θ 求梯度是相同的

$$\partial_\theta \mathcal{L}_{\text{discriminator}}(D_\theta) = \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] - \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z}\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\tilde{\mu}(\tau)}\right]$$

$$\begin{aligned}\partial_\theta \mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] + \partial_\theta \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)}\right]\right) \\ &= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] + \left(\mathbb{E}_{\tau\sim\mu}\left[\frac{-\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\tilde{\mu}(\tau)}\right] \Big/ \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)}\right]\right) \\ &= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] - \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z}\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\tilde{\mu}(\tau)}\right] \\ &= \partial_\theta \mathcal{L}_{\text{discriminator}}(D_\theta).\end{aligned}$$

注意在最大熵 IRL 中，μ~(τ)是用来计算重要性采样的权重的，在求梯度时不将其当做关于 θ 的函数。

3. 训练生成器相当于优化产生采样 policy

$$\begin{aligned}\mathcal{L}_{\text{generator}}(q) &= \mathbb{E}_{\tau\sim q}[\log(1-D(\tau)) - \log(D(\tau))] \\ &= \mathbb{E}_{\tau\sim q}\left[\log\frac{q(\tau)}{\widetilde{\mu}(\tau)} - \log\frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right] \\ &= \mathbb{E}_{\tau\sim q}[\log q(\tau) + \log Z + c_\theta(\tau)] \\ &= \log Z + \mathbb{E}_{\tau\sim q}[c_\theta(\tau)] + \mathbb{E}_{\tau\sim q}[\log q(\tau)] = \log Z + \mathcal{L}_{\text{sampler}}(q)\end{aligned}$$

其中 log Z 是关于 θ（判别器的参数）的函数，在训练生成器时是固定的，则只剩下第二项。

所以，经过简单的修改后，我们发现，更新判别器相当于更新 reward function；更新生成器具体是在更新产生采样的 policy，产生的采样用来估计 partition function。

三．Energy-based reinforcement learning

基于能量的强化学习，也称为熵正则的强化学习。在常规的强化学习中，我们的目标是最大化 total reward：

$$\pi^*_{\text{std}} = \arg\max_\pi \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{3.1}$$

而如果加入一个正则项：$\mathrm{KL}_t = D_{\mathrm{KL}}\left[\pi(\cdot\mid s_t)\parallel\overline{\pi}(\cdot\mid s_t)\right]$，则我们得到了； 熵正则回报：$\sum_{t=0}^{\infty}\gamma^t(r_t - \tau\,\mathrm{KL}_t)$。τ 是一个温度系数，权衡两部分的重要性；kl 散度中的第二项是一个“参考”的分布，常选择均匀分布，则减小 kl 散度可以拆分为增大 π(a|s)的熵和减小和均匀分布的交叉熵，这样我们找到回报最大的轨迹，同时鼓励更多的探索。有些文献中，该目标也可能只包含 π(a|s)的熵：

$$\pi^*_{\text{MaxEnt}} = \arg\max_\pi \textstyle\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\cdot\mid\mathbf{s}_t))\right] \tag{3.2}$$

本文的推导主要按照第一种定义，但是要替换也是非常容易的。

优化这个目标一般有两种方法，第一种是间接的方法，先设置一个 action value function q_θ，然后再基于 q_θ 定义一个 policy；第二种则是直接对策略进行随机梯度上升。 这两种方法事实上是等价的，我们先介绍第一种：

a) 在 kl 惩罚项下定义 state、action value：

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t(r_t - \tau\,\mathrm{KL}_t)\,\middle|\, s_0 = s\right] \tag{3.3}$$

$$Q_\pi(s,a) = \mathbb{E}\left[r_0 + \sum_{t=1}^{\infty}\gamma^t(r_t - \tau\,\mathrm{KL}_t)\,\middle|\, s_0 = s, a_0 = a\right] \tag{3.4}$$

注意到 q value 不包含第一个 kl 惩罚项，因为这一项不依赖于 a_0；则 state value 比 q 在 action 下的期望还要多一个惩罚项：

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}\left[Q_\pi(s,a)\right] - \tau\,\mathrm{KL}(s) \tag{3.5}$$

则优化目标(3.2)可以等价的写成 max $V_\pi(s_0)$，最优 policy 是可以使 $V_\pi(s_0)$最大

的 policy，其中 s_0 是 MDP 的初始状态或者初始状态的集合。

b) 利用 value 进行 policy improvement：

基于已有 policy 的 value 值，定义一个新的 policy：

$$\pi'(\cdot|s) = arg\max_{\pi} V_{\pi}(s) \tag{3.6}$$

则可以得到

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

对所有 state 都成立，则可保证 π′对 π 是有提升的。

$$\begin{aligned}\pi_Q^{\mathcal{B}}(\cdot \mid s) &= \arg\max_{\pi}\{\mathbb{E}_{a\sim\pi}\left[Q(s,a)\right] - \tau D_{\mathrm{KL}}\left[\pi \parallel \overline{\pi}\right](s)\} \\ &= \overline{\pi}(a \mid s)\exp(Q(s,a)/\tau)/\underbrace{\mathbb{E}_{a'\sim\overline{\pi}}\left[\exp(Q(s,a')/\tau)\right]}_{\text{normalizing constant}}.\end{aligned} \tag{3.8}$$

我们将(3.8)的 1 式与常见的 greedy policy 对比：

$$\pi'(s) \doteq \arg\max_{a} q_{\pi}(s,a)$$

greedy policy 是一个确定性的 policy，而 1 式中的 policy 是具有随机性的，要在该 policy 下使 Q 的期望最大，同时鼓励熵增。2 式可以通过使 1 式对 π(a|s)的导数为 0 得出。我们看到这个分布时服从 boltzmann 分布的，分母部分是一个只与 state 有关的函数。根据这一部分，我们定义一个新的 state value 值：

$$V_Q(s) = \tau\log\mathbb{E}_{a'\sim\overline{\pi}}\left[\exp(Q(s,a')/\tau)\right] \tag{3.9}$$

在这个定义下，还存在：

$$V_Q(s) = \mathbb{E}_{a\sim\pi_Q^{\mathcal{B}}(s)}\left[Q(s,a)\right] - \tau D_{\mathrm{KL}}\left[\pi_Q^{\mathcal{B}} \parallel \overline{\pi}\right](s) \tag{3.10}$$

也就是说这个 value 值代表在 boltzmann 策略 π_Q 下，对 state s 的熵正则回报的评估。

则最优策略可以重新写作以下形式：

$$\pi_Q^{\mathcal{B}}(a \mid s) = \overline{\pi}(a \mid s)\exp((Q(s,a) - V_Q(s))/\tau) \tag{3.11}$$

观察该式，这个 policy 是一个能量模型，其中 Q(s,a)是负的能量，$V_Q(s)$是 log partition function。如果定义 advantage 函数：

$A_Q(s,a) = Q(s,a) - V_Q(s)$

则最优 policy 还可以写成：

$$\frac{\pi_Q^{\mathcal{B}}(a \mid s)}{\overline{\pi}(a \mid s)} = \exp(A_Q(s,a)/\tau)$$

c) soft value iteration

定义了 policy 后，我们需要讨论如何学习 value 值，已知：

$$Q_{\pi}(s,a) = \mathbb{E}\left[r_0 + \sum_{t=1}^{\infty}\gamma^t(r_t - \tau\,\mathrm{KL}_t)\,\middle|\, s_0 = s, a_0 = a\right]$$

根据(3.10)发现第二项刚好是 $V_Q(s')$，则我们得到了 soft bellman equation：

$$
\begin{aligned}
Q_{\text{soft}}^{\pi}(\mathbf{s},\mathbf{a}) &= r(\mathbf{s},\mathbf{a}) + \gamma\,\mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}}}\left[\mathcal{H}(\pi(\cdot\,|\mathbf{s}')) + \mathbb{E}_{\mathbf{a}'\sim\pi(\cdot\,|\mathbf{s}')}\left[Q_{\text{soft}}^{\pi}(\mathbf{s}',\mathbf{a}')\right]\right] \\
&= r(\mathbf{s},\mathbf{a}) + \gamma\,\mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}}}\left[V_{\text{soft}}^{\pi}(\mathbf{s}')\right].
\end{aligned}
\tag{3.12}
$$

结合(3.9)，我们得到了 soft value iteration：

$$
\begin{aligned}
Q_{\text{soft}}(\mathbf{s}_t,\mathbf{a}_t) &\leftarrow r_t + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p_{\mathbf{s}}}\left[V_{\text{soft}}(\mathbf{s}_{t+1})\right],\ \forall \mathbf{s}_t,\mathbf{a}_t \\
V_{\text{soft}}(\mathbf{s}_t) &\leftarrow \alpha\log\int_{\mathcal{A}}\exp\left(\frac{1}{\alpha}Q_{\text{soft}}(\mathbf{s}_t,\mathbf{a}')\right)d\mathbf{a}',\ \forall \mathbf{s}_t
\end{aligned}
\tag{3.13}
$$

对比常规的 value iteration，发现第二项由单纯的 max Q(s,a)操作变成了一个 soft max，soft 体现在如果某个 action 的 $Q_{soft}(s,a)$远大于其他 action，则右侧将趋近于 max Q(s,a)。温度参数是用来调节软/硬的程度，α→0，则右侧变成一个“硬”的 max。

d) soft q learning

在上一小节中，我们得到了一个基于动态规划的强化学习过程，但这只适用于环境动态已知，状态、动作空间有限的情况。

$$
\begin{aligned}
Q_{\text{soft}}(\mathbf{s}_t,\mathbf{a}_t) &\leftarrow r_t + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p_{\mathbf{s}}}\left[V_{\text{soft}}(\mathbf{s}_{t+1})\right],\ \forall \mathbf{s}_t,\mathbf{a}_t \\
V_{\text{soft}}(\mathbf{s}_t) &\leftarrow \alpha\log\int_{\mathcal{A}}\exp\left(\frac{1}{\alpha}Q_{\text{soft}}(\mathbf{s}_t,\mathbf{a}')\right)d\mathbf{a}',\ \forall \mathbf{s}_t
\end{aligned}
$$

而在 1. 环境未知：此时只能使用基于采样的方法（MC\TD），Q learning 中使用的是后者，仍然基于 bellman equation，但是此时将期望换为了采样。2. 状态空间连续，不可能对所有的 state 都评估 value 值，此时可以用一个函数来学习 q value，用 target 和 $Q_\theta(s,a)$的 l2 loss 作为目标：

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

target 包含 reward 和采样到的 next state 的 soft state value。

$$
\hat{Q}_{\text{soft}}^{\bar{\theta}}(\mathbf{s}_t,\mathbf{a}_t) = r_t + \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p_{\mathbf{s}}}[V_{\text{soft}}^{\bar{\theta}}(\mathbf{s}_{t+1})]
$$

此时还存在一个问题是，计算 2 式涉及到一个对 action 空间的求和，这计算量是很大的，而若 action 空间连续，需要对其求积分，则无法计算。可以通过重要性采样的方法：

$$
V_{\text{soft}}^{\theta}(\mathbf{s}_t) = \alpha\log\mathbb{E}_{q_{\mathbf{a}'}}\left[\frac{\exp\left(\frac{1}{\alpha}Q_{\text{soft}}^{\theta}(\mathbf{s}_t,\mathbf{a}')\right)}{q_{\mathbf{a}'}(\mathbf{a}')}\right]
$$

分母中的 q 可以是任意分布，但是这个分布如果更多的覆盖$\exp\left(\frac{1}{\alpha}Q_{\text{soft}}^{\theta}(\mathbf{s}_t,\mathbf{a}_t)\right)$高的区域，则估计会更加准确。所以可以学习一个 policy 的函数（action 有限时，π 是可以直接求得的），目标如下：

$$
\mathrm{D}_{\mathrm{KL}}\left(\pi^{\phi}(\cdot\,|\mathbf{s}_t)\,\middle\|\,\exp\left(\frac{1}{\alpha}\left(Q_{\text{soft}}^{\theta}(\mathbf{s}_t,\cdot) - V_{\text{soft}}^{\theta}\right)\right)\right)
$$

用当前的 policy 作为 q 进行采样。