Policy Gradient Methods for Reinforcement Learning

MARKOV DECISTION PROCESS

强化学习是建立在MDP的设定下的，每个时间点根据当前的状态做决策，然后系统根据特定的决策会有不同的transition matrix (assumed to be stationary)，MDP也叫controlled Markov chain，也是最优控制和stochastic control，以及动态规划里面最基础的模型，这些领域之间都是高度相关的，尤其是动态规划与强化学习。我们这里定义一条轨迹 $\tau = (s_1, a_1, s_2, \dots)$ 是一串状态和动作，那么这个轨迹的概率是：

$$p_\theta(s_1, a_1, \dots, s_T, a_T) = p(s_1) \cdot \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

其中状态转移的pdf是模型决定的，而 $\pi_\theta(a_t|s_t)$ 则是我们的policy，这里的policy是一个stochastic version。policy gradient的思路就是怎么去优化这个pdf，也就是我们的policy，来使得系统有更好的收益：

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \sum_t r(s_t, a_t) = \arg\max_\theta J(\theta)$$

这里的θ是假定我们优化的policy是parameterized的，实际中我们用一个神经网络等各种网络他们都是参数化的模型。有一个地方需要注意的是，这里我们的设定系统是controlled Markov chain，如果我们只能观察到一部分数据(system is partially observable)，那么我们观察到的状态就没有很好地Markov的属性了，因为这时候系统变成了hidden Markov chain,显层之间的transition matrix会受latent layer的影响，最近一直在用这个模型在做状态估计，也非常熟悉。在强化学习里面我们learn的不仅仅是policy，也learn了model(transition matrix)，如果变成部分可观测之后这个transition matrix就不好learn了，他就不是一个stationary的矩阵了，这是一般意义上，但是实际上这个更多的会限制value-based method，对于policy gradient，后面会发现求了一阶导数大概还是可以做的。

POLICY GRADIENT

Policy gradient简单讲就是stochastic gradient descent (ascent, depends on reward or cost)，通过对参数求偏导数然后做gradient descent来优化模型参数：

$$\nabla J_\theta(\theta) = \int \nabla \pi_\theta(\tau) r(\tau) d\tau = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}(\nabla_\theta \log \pi_\theta(\tau) r(\tau))$$

这里做了一个等价变换，把reward写成了关于轨迹τ的期望，这样的好处是写成轨迹的期望之后很自然的就可以用sample的方式通过sample轨迹来对这个期望做估计。这个还可以进一步化简，把这个轨迹的pdf代进去然后得到：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi(\theta)} \left(\sum_t \nabla_\theta \log \pi_\theta(a_t | a_t) \right) \left(\sum_t r(s_t, a_t) \right)$$

中间唯一的化简就是模型的transition matrix并不会根据我们模型的参数改变（并不是不会根据模型改变，是不会根据模型的参数改变），这也是为何policy gradient是一个天然的model-free的算法，然后再看这个结果，这个结果大概是policy gradient最经典的结果了，第一个括号里面是gradient of likelihood也就是说特定轨迹发生概率岁参数变化的变化率，第二个括号是这个轨迹的reward，这样乘起来很自然的如果一条轨迹reward高，并且这个轨迹的gradient of likelihood大，就会得到一个大的gradient，是非常合理的。最简单的REINFOCE算法就是通过sample轨迹，按照这个公式更新gradient，然后再按照gradient更新参数来优化整个模型。这里不去具体讲怎么用sampling的方式估计这个gradient，在reinforcement learning里面有两种方法来估计，一种是Monte Carlo，简言之就是采样，另外一种是Temperal Difference，是更常用的一种可以做online更新的方式，这个和DP中的Bellman equation非常相似了，这里不具体展开，简言之这些expectation都可以通过run model收集数据的方式来估计。

ACTOR-CRITIC

不讲详细只讲大致思路了这里，actor-critic是结合了policy gradient和value-based method的一种方法，出发点仍然是policy gradient，我们最终得到的policy gradient的计算gradient的equation有一个问题，我们直接用这个equation去估计会得到一个方差很大的结果，会导致算法特别不稳定，收敛性极差（也许需要去用stochastic optimization的方法论来分析这个方差对收敛性的影响，我猜），后面有一些列的方法都是想方设法去reduce the variance of the approximation。大致有两个比较常用的思路，我们说过最后的gradient等于gradient of likelihood times the reward，主要就是在这个reward中下文章，一来可以改成reward to go，二来可以给这个reward减去一个baseline，这两个我感觉还是比较直观的。如果我们减去的baseline是value function (of the current policy

parameter)，那么就是大家常说的A2C (advantage actor-critic)，经过这些变换之后，gradient用下面的equation估计：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi(\theta)}\left(\sum_t \nabla_\theta \log \pi_\theta(a_t|a_t)\right)\left(\sum_t A^\pi(s_t,a_t)\right)$$

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

也就是说，我们在更新policy，估计policy gradient的时候，同时也要去算value function，这个value function就是所谓之critic，policy是actor，大致actor-critic就是演员在演，下面的critic在打分的过程。

ADVANCED POLICY GRADIENT

这里建议参考两个参考文献，主要看一下第一个TRPO吧。

1. Schulman, John, et al. "Trust region policy optimization." International conference on machine learning. 2015.
2. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

值得注意的是我们这里看到的policy gradient好像与传统dynamic programming里面的policy iteration 之间有千丝万缕的联系，我们考虑不同参数之间expected reward的差值，可以推导出：

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

外面的期望是根据新的轨迹followed distribution given θ'，里面的advantage是θ下的，然后我们把这个整个轨迹的期望拆解成每个状态下的概率乘上我们做决策的概率（概率不能这么拆解，期望可以，本质上就是summation换顺序）然后再用一步importance sampling，把policy的pdf换掉，从θ换成θ'，这里稍微有点绕，结果是：

$$\mathbb{E}_{\tau\sim p_{\theta'}}\left(\sum_t \gamma^t A^{\pi_\theta}(s_t,a_t)\right) = \mathbb{E}_{s_t\sim p'_\theta(s_t)}\left(\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left(\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\sum_t \gamma^t A^{\pi_\theta}(s_t,a_t)\right)\right)$$

可以看到里面是对policy取得期望，外面是对state取得期望，然后里面的期望我们通过importance sampling做了一步变换，外面还保留了原来的base，这里就是policy iteration的结论，我们只要保证这个期望最后整体是大于0的，就可以保证$J(\theta') - J(\theta) \geq 0$，也就是说我们新的参数提升了系统的表现，通过这个公式来看，我们很容易做到，只要我们让新的参数下的policy $\pi_{\theta'}()$ 产生一个大于零的expected advantage就好了，比如最常用的$a' = \arg\max A^{\pi_\theta}(s_t, a_t)$，就可以for sure的产生一个至少不会更差的policy，也即policy gradient的方法论。

但是这个equation在实际中并不能操作，因为我们sample的过程中或者估计的过程中，用的是θ下的轨迹，处理方式是直接用θ下面每个状态的分布概率去替代θ'下的分布概率，这个估计的误差可以被bound住只要我们保证这两个参数下的policy之间的difference被一个ϵ bound住：$\sup_s d_{TV}(\pi_{\theta'}(\cdot \mid s), \pi_\theta(\cdot \mid s)) \leq \epsilon$，这里的$d_{TV}$ 是distribution distance measure中的total variation，就是1范数的积分（除以2），f-divergence的一种，可以被KL-divergence bound住，这样我们会有：

$$\mathbb{E}_{\tau \sim p_{\theta'}}\left(\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right) \geq \mathbb{E}_{s_t \sim p_\theta(s_t)}\left(\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left(\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right)\right) - \sum_t 2\epsilon t C$$

顺着这个思路且把total variation换成KL divergence，在TRPO的paper中，他们证明了下面的policy可以有monotonic increase guarantee：

$$\pi_{i+1} = \arg\max_\pi \mathbb{E}_{s_t \sim p_\theta(s_t)}\left(\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left(\frac{\pi(a_t|s_t)}{\pi_\theta(a_t|s_t)} \sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right)\right) - \left(\frac{2\epsilon\gamma}{1-\gamma^2}\right) \sup_s d_{KL}(\pi_i, \pi)$$

where $\epsilon = \max_s \max_a A^\pi(s, a)$. 大致的形式是一个期望（用于估计新参数下轨迹的期望）加上一个penalty项，这个penalty项penalize了新参数和原参数下policy distribution的距离。值得注意的是第一项的linear approximation是和policy gradient的结果是一样的，也即所谓policy gradient as policy iteration。

这个公式有理论保证，但是somehow在实际中用起来并不是很方便，于是乎在实际中有两种操作方式，一种是把penalty项改成contraint，也就成了TRPO， trust region policy optimization:

$$\max_\pi \mathbb{E}_{s_t \sim p_\theta(s_t)}\left(\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left(\frac{\pi(a_t|s_t)}{\pi_\theta(a_t|s_t)} \sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right)\right)$$

$$s.t. \quad d_{KL}(\pi_i, \pi) \leq \epsilon$$

这个关于KL divergence的限制条件还可以用Fisher Information matrix来做二阶近似。除了TRPO，还有Proximal Policy Optimization (PPO),并没有把penalty 放到contraint里面，而是动态的调整penalty的权重以及come up with了一些surrogate function 替代前面的objective function。

这些都是大致讲下故事吧，有一些细节并没有认真看，后面可能用一下PPO或者TRPO做一些具体的应用，然后有些细节可能会被逼着重新回头去看。