Stochastic Processes and Reinforcement Learning

2025-02-05

Pu Fanyi
Nanyang Technological University

Contents

1. Stochastic Process

教材 (Privault, 2013) / 划了重点的教材捏 / Berkeley EECS126

1.1. Markov Chains

1.1.1. Gambling Problems

有 S 块钱，A 有 K 块钱，B 有 $S-K$ 块。每次有 p 的概率 A 从 B 拿走一块，$q=1-p$ 的概率 B 从 A 拿走一块。谁拿到 S 块钱就算赢。

$$X_{n+1}=\begin{cases} X_n+1 \text{ if } p \\ X_n-1 \text{ if } q \end{cases}$$

$f_S(k)$ 表示 A 赢的概率。很显然我们有：

$$f_S(k)=pf_S(k+1)+qf_S(k-1)$$

不难解出

$$f_S(k)=\frac{(p/q)^{S-k}-1}{(p/q)^S-1}$$

$T_{0,S}$ 表示一个游戏啥时候结束，$h_S(k)=\mathbb{E}\left[T_{0,S} \mid X_0=K\right]$。
很显然

$$h_S(k)=1+ph_S(k+1)+qh_S(k-1)$$

这个方程的特解比较难搞，注意到 $p+q=1$，我们可以改写成差分方程：

$$-1=p(h_S(k+1)-h_S(k))+q(h_S(k)-h_S(k-1))$$

观察出方程在 $p\neq q$ 的时候一个特解为 $\frac{k}{q-p}$。
不难解出齐次方程 $h_S(k)=ph_S(k+1)+qh_S(k-1)$ 的解，最终可以得到在 $p\neq q$ 时

$$h_s(k)=\frac{1}{q-p}\left(k-S\cdot\frac{1-(q/p)^k}{1-(q/p)^S}\right)$$

当 $p=q=\frac{1}{2}$ 时，我们有特解 $-k^2$。所以说 $p=q$ 时

$$h_S(k)=k(S-k)$$

1.1.2. Random Walks

Bernoulli Random Walks: X_n 相互独立，其中

$$\begin{cases} \mathbb{P}(X_k=+1)=p \\ \mathbb{P}(X_k=-1)=q \end{cases}$$

$p+q=1$，然后我们定义

$$S_n = \sum_{i=1}^{n} X_i$$

很显然 $\mathbb{P}(S_{2n} = 2k+1) = \mathbb{P}(S_{2n+1} = 2k) = 0$，然后

$$\begin{cases} \mathbb{P}(S_{2n} = 2k) = \binom{2n}{n+k} p^{n+k} q^{n-k} \\ \mathbb{P}(S_{2n+1} = 2k+1) = \binom{2n+1}{n+k+1} p^{n+k+1} q^{n-k} \end{cases}$$

难度在我们怎么计算他啥时候回到 0。我们令

$$T_0^r = \inf\{n \geq 1 : S_n = 0\}$$

表示第一次回到 0 的时间。

然后我们设

$$g(n) = \mathbb{P}(T_0^r = n \mid S_0 = 0)$$

也就是在第 n 步第一次回到 0 的概率。那很显然 $g(2k+1) = 0$。

接下来是一个比较神奇的套路，就是我们假设 $h(n)$ 为第 n 步回到 0 的概率，那我们可以得到一个卷积式子：

$$h(n) = \sum_{k=0}^{n-2} g(n-k) h(k)$$

也就是先走 $n-k$ 步第一次回到 0，然后继续走 k 步回到 0。所以我们现在只要解决 h 就可以了。

我们考虑 $h(n)$ 的生成函数

$$H(s) = \mathbb{E}\left[s^{T_0^r} \cdot \mathbb{1}_{T_0^r < \infty}\right] = \sum_{n=0}^{\infty} h(n) s^n$$

大概推推：

$$\begin{aligned} H(s) &= \sum_{n=0}^{\infty} h(n) s^n \\ &= \sum_{n=0}^{\infty} \binom{2n}{n} p^n q^n s^{2n} \\ &= \sum_{n=0}^{\infty} \frac{(2n)!}{(n!)^2} \left(pqs^2\right)^n \\ &= \sum_{n=0}^{\infty} \frac{\left(\prod_{i=1}^{n} 2i\right)\left(\prod_{i=1}^{n} (2i-1)\right)}{(n!)^2} \left(pqs^2\right)^n \\ &= \sum_{n=0}^{\infty} \frac{\prod_{i=1}^{n} (2i-1)}{n!} \left(2pqs^2\right)^n \end{aligned}$$

$$
\begin{aligned}
&= \sum_{n=0}^{\infty} \frac{(-2)^n \prod_{i=1}^{n} \left(-\frac{1}{2} - (i-1)\right)}{n!} \left(2pqs^2\right)^n \\
&= \sum_{n=0}^{\infty} \frac{\left(-\frac{1}{2}\right)^{\underline{i}}}{n!} \left(-4pqs^2\right)^n \\
&= \left(1 - 4pqs^2\right)^{-\frac{1}{2}}
\end{aligned}
$$

又考虑到：

$$
\begin{aligned}
G(s)H(s) &= \left(\sum_{i=1}^{\infty} s^i g(i)\right)\left(\sum_{j=0}^{\infty} s^j h(j)\right) \\
&= \sum_{i=2}^{\infty}\sum_{j=0}^{\infty} s^{i+j} g(i)h(j) \\
&= \sum_{k=2}^{\infty} s^k \sum_{i=2}^{\infty} g(i)h(k-i) \\
&= \sum_{k=2}^{\infty} s^k h(k) \\
&= -1 + \sum_{k=0}^{\infty} s^k h(k) \\
&= -1 + H(s)
\end{aligned}
$$

于是乎

$$
G(s) = 1 - \frac{1}{H(s)} = 1 - \sqrt{1 - 4pqs^2}
$$

所以

$$
\begin{aligned}
\mathbb{P}(T_0^r = \infty \mid S_0 = 0) &= 1 - \mathbb{P}(T_0^r < \infty \mid S_0 = 0) \\
&= 1 - G(1) = \sqrt{1 - 4pq} \\
&= \sqrt{4p^2 - 4p + 1} \\
&= |2p - 1| \\
&= |p - q|
\end{aligned}
$$

而根据前面的 Gambling Problems，其实我们已经解出当 $k \neq 0$ 时：

$$
\mathbb{P}(T_0^r = \infty \mid S_0 = k) = 1 - \lim_{S \to \infty} f_S(k) = \max\left\{0, 1 - \left(\frac{q}{p}\right)^k\right\}
$$

然后我们来计算 $\mathbb{E}[T_0^r \mid S_0 = 0]$ 的时候会发现如果 $\mathbb{P}(T_0^r \mid S_0 = 0) > 0$ 的时候这个期望肯定是 ∞。而 $\mathbb{P}(T_0^r \mid S_0 = 0)$ 只有在 $p = q = \frac{1}{2}$ 的时候才会为 0。然鹅当 $p = q = \frac{1}{2}$ 时：

$$\mathbb{E}[T_0^r \mid S_0 = 0] = \mathbb{E}\left[T_0^r \cdot \mathbb{1}_{\{T_0^r < \infty\}} \,\middle|\, S_0 = 0\right] = \left.\frac{\partial G}{\partial s}\right|_{s=1} = \infty$$

所以我们不管 p, q 都有：

$$\mathbb{E}[T_0^r \mid S_0 = 0] = \infty$$

关于 first time 的 distribution 的话，我们不难算出

$$\mathbb{P}(T_0^r = 2k \mid S_0 = 0) = \frac{1}{(2k)!} \left.\frac{\partial^{2k} G}{\partial s^{2k}}\right|_{s=0} = \frac{1}{2k-1}\binom{2k}{k}(pq)^k$$

1.1.3. Discrete-Time Markov Chains

Markov property 指的是下一步的 distribution 只跟当前有关：

$$\mathbb{P}(Z_{n+1} = j \mid Z_n = i_n, \cdots, Z_0 = i_0) = \mathbb{P}(Z_{n+1} = j \mid Z_n = i_n)$$

π_n 是行向量，转移方程

$$\pi_{n+1} = \pi_n P$$

首先研究 hitting probabilities。假设状态空间为 $\mathbb{S}$，现在有一个点集 $A \subset \mathbb{S}$ 是吸收点。也就是说 $\forall s \in \mathbb{S}, P_{s,s} = 1$。我们康康从 k 开始被 A 中哪个点吸收的分布：

$$g_l(k) = \mathbb{P}\left(Z_{T_A} = l \,\middle|\, Z_0 = k\right)$$

其中 T_A 表示第一次撞到 A 的概率。

显然

$$g_l(k) = P_{k,l} + \sum_{m \in \mathbb{S} \setminus A} P_{k,m} g_l(m)$$

然后我们研究从一个点开始期望多久被吸收，我们定义：

$$h_A(k) = \mathbb{E}[T_A \mid Z_0 = k]$$

显然

$$h_A(k) = 1 + \sum_{m \in \mathbb{S} \setminus A} P_{k,m} h_A(m)$$

当然很多事会我们会钦定最后 d 个为吸收点，也就是：

$$P = \begin{pmatrix} Q & R \\ 0 & I_d \end{pmatrix}$$

酱一来我们就可以简单地写成

$$h_A = \mathbb{1}_{n-d} + Q h_A$$

当然很多时候我们每个点都是有个 utility 的，也就是说：

$$h_A(k) = \mathbb{E}\left[\sum_{n=0}^{T_A} r(Z_n) \,\middle|\, Z_0 = k\right]$$

那酱的话就把 $\mathbb{1}$ 换成 r 就可以了。

接下来是 return times。我们定义 T_j^r 为第一次到 j 的时间（但不是 Z_0）：

$$T_j^r = \inf\{n \geq 1 : Z_n = j\}$$

然后 $\mu_j(i)$ 表示从 i 开始第一次回到 j 的期望时间：

$$\mu_j(i) = \mathbb{E}\left[T_j^r \,\middle|\, Z_0 = i\right]$$

酱紫 $\mu_i(i)$ 就成功定义了“return times”。

显然：

$$\mu_j(i) = 1 + \sum_{m \in \mathbb{S}} P_{i,m} \mu_j(m)$$

我们接下来定义 $p_{i,j}$ 为从 i 能走到 j 的概率，当然一开始不算：

$$p_{i,j} = \mathbb{P}\left(T_j^r < \infty \,\middle|\, Z_0 = i\right)$$

我们定义 $f_{i,j}^{(n)}$ 为从 i 开始走 n 步恰好第一次走到 j 的概率：

$$f_{i,j}^{(n)} = \mathbb{P}\left(T_j^r = n \,\middle|\, Z_0 = i\right)$$

显然

$$p_{i,j} = \sum_{n=1}^{\infty} f_{i,j}^{(n)}$$

Number of returns 被定义为：

$$R_j = \sum_{n=1}^{\infty} \mathbb{1}_{\{Z_n = j\}}$$

那么 R_j 的分布其实是：

$$\mathbb{P}\left(R_j = m \,\middle|\, Z_0 = i\right) = \begin{cases} 1 - p_{i,j} & \text{if } m = 0 \\ p_{i,j} \cdot p_{j,j}^{m-1} \cdot \left(1 - p_{j,j}\right) & \text{if } m \geq 1 \end{cases}$$

那期望也好算：

$$\mathbb{E}\left[R_j \,\middle|\, Z_0 = i\right] = \sum_{m=1}^{\infty} m \cdot p_{i,j} \cdot p_{j,j}^{m-1} \cdot \left(1 - p_{j,j}\right) = \frac{1}{1 - p_{j,j}}$$

所以说我们得到一个性质就是，如果要

$$\mathbb{E}[R_i \mid Z_0 = i] = \frac{1}{1 - p_{i,i}}$$

这个东西有穷，当且仅当 $p_{i,i} < 1$。

而还有一个意义比较明确而且封闭的式子是：

$$\mathbb{E}[R_j \mid Z_0 = i] = \mathbb{E}\left[\mathbb{1}_{\{X_n = j\}} \,\middle|\, X_0 = i\right] = \sum_{n=1}^{\infty} [P^n]_{i,j} = -\mathbb{1}_{\{i=j\}} + \left[(I - P)^{-1}\right]_{i,j}$$

1.1.4. Branching Processes

一开始我有一个东西，然后没过一段时间这个东西随机分裂成 Y 个一样的东西，然后一直这样分裂下去。

$$X_0 = 1, X_{n+1} = \sum_{k=1}^{X_n} Y_k$$

其中

$$\mathbb{P}(Y < \infty) = \sum_{n \geq 0} \mathbb{P}(Y = n) = 1$$

考虑转移矩阵 P，$P_{i,j}$ 表示 i 个东西分裂成 j 个的概率。显然 $P_{0,0} = 1$，没有东西就无法分裂。$P_{1,j} = \mathbb{P}(Y = j)$，指的是从一个东西分裂开来。

酱紫的话是个树状结构，所以叫 Branching Process。

考虑生成函数 $G_n(s) = \mathbb{E}[s^{X_n} \mid X_0 = 1]$ 表示第 n 代的概率生成函数，我们有：

$$G_{n+1}(s) = G_n(G_1(s)) = G_1(G_n(s))$$

证明的话：

$$\begin{aligned}
G_{n+1}(s) &= \mathbb{E}[s^{X_{n+1}} \mid X_0 = 1] \\
&= \mathbb{E}\left[s^{\sum_{l=1}^{X_n} Y_l} \,\middle|\, X_0 = 1\right] \\
&= \sum_{k=1}^{\infty} \mathbb{E}\left[\prod_{l=1}^{k} s^{Y_l} \,\middle|\, X_n = k\right] \mathbb{P}(X_n = k \mid X_1 = 1) \\
&= \sum_{k=1}^{\infty} \mathbb{E}[s^Y]^k \mathbb{P}(X_n = k \mid X_1 = 1) \\
&= \sum_{k=1}^{\infty} G_1(s)^k \mathbb{P}(X_n = k \mid X_1 = 1) \\
&= G_n(G_1(s))
\end{aligned}$$

于是乎对于 n 轮后的期望值我们有：

$$\mu_n = \mathbb{E}[X_n \mid X_0 = 1]$$

$$
\begin{aligned}
&= \left.\frac{\partial G_n(s)}{\partial s}\right|_{s=1} \\
&= \left.\frac{\partial G_{n-1}(G_1(s))}{\partial G_1(s)} \cdot \frac{\partial G_1(s)}{\partial s}\right|_{s=1} \\
&= \left.\frac{\partial G_{n-1}(G_1(s))}{\partial G_1(s)}\right|_{s=1} \cdot \lim_{s\to 1^-} \left.\frac{\partial G_1(s)}{\partial s}\right|_{s=1} \\
&= \left.\frac{\partial G_{n-1}(s)}{\partial s}\right|_{s=1} \cdot \left.\frac{\partial G_1(s)}{\partial s}\right|_{s=1} \\
&= \mu_{n-1}\mu_1 \\
&= \mu_1^n
\end{aligned}
$$

对于一个 branching process $(X_n)_{n\in\mathbb{N}}$：

- Supercritical: $\mu_1 > 1$，$\mu_n \to \infty$
- Critical: $\mu_1 = 1$，$\mu_n \to \infty$
- Subcritical: $\mu_1 < 1$，$\mu_n \to 0$

同样的，我们考虑方差

$$
\begin{aligned}
\sigma_n^2 &= \mathrm{Var}[X_n \mid X_0 = 1] \\
&= \left.\frac{1}{2}\frac{\partial^2 G_n(s)}{\partial s^2}\right|_{s=1} \\
&= \left.\frac{1}{2}\frac{\partial}{\partial s}\left(\frac{\partial}{\partial s}G_{n-1}(s) \cdot \frac{\partial}{\partial s}G_1(s)\right)\right|_{s=1} \\
&= \left.\left(\frac{\partial^2}{\partial s^2}G_{n-1}(s) \cdot \frac{\partial}{\partial s}G_1(s)\right)\right|_{s=1} + \left.\left(\frac{\partial}{\partial s}G_{n-1}(s) \cdot \frac{\partial^2}{\partial s^2}G_1(s)\right)\right|_{s=1} \\
&= \sigma_{n-1}^2\mu_1 + \mu_{n-1}\sigma_1^2 \\
&= \sigma_{n-1}^2\mu_1 + \mu_1^{n-1}\sigma_1^2 \\
&= \begin{cases} n\sigma_1^2 & \text{if } \mu = 1 \\ \sigma_1^2\mu_1^{n-1}\frac{1-\mu_1^n}{1-\mu_1} & \text{if } \mu \neq 1 \end{cases}
\end{aligned}
$$

接下来我们要研究的是，X_n 能延续多久，也就是“time to extinction”。我们定义

$$
T_0 = \inf\{n \geq 0 : X_n = 0\}
$$

以及最终的 extinction probability

$$
\alpha_k = \mathbb{P}(T_0 < \infty \mid X_0 = k)
$$

首先显然我们有：

$$
\mathbb{P}(T_0 = n \mid X_0 = 1) = \mathbb{P}(X_n = 0 \mid X_0 = 1) - \mathbb{P}(X_{n-1} = 0 \mid X_0 = 1)
$$

$$
\begin{aligned}
&= G_n(0) - G_{n-1}(0) \\
&= G_1(G_{n-1}(0)) - G_{n-1}(0)
\end{aligned}
$$

而我们来考虑 α_k，首先显然这 k 个是独立的：

$$\alpha_k = \alpha_1^k$$

而我们考虑了 α_1：

$$
\begin{aligned}
\alpha_1 &= \lim_{n\to\infty} \mathbb{P}(T_0 < n \mid X_0 = 1) \\
&= \lim_{n\to\infty} \mathbb{P}(X_n = 0 \mid X_0 = 1) \\
&= \lim_{n\to\infty} G_n(0)
\end{aligned}
$$

另一个神奇的性质是 α_1 一定是方程 $G_1(\alpha) = \alpha$ 的解：

$$
\begin{aligned}
\alpha_1 &= \sum_{k=0}^{\infty} \alpha_k \mathbb{P}(X_1 = k \mid X_0 = 1) \\
&= \sum_{k=0}^{\infty} \alpha_1^k \mathbb{P}(X_1 = k \mid X_0 = 1) \\
&= G_1(\alpha_1)
\end{aligned}
$$

当然考虑到

$$\alpha = G_1(\alpha) = G_1(G_1(\alpha)) = \cdots$$

所以

$$\forall k \in \mathbb{N}, \alpha = G_k(\alpha)$$

我们可以进一步证明 α_1 一定是方程 $G_1(\alpha) = \alpha$ 最小的正解。

首先因为 $\frac{\partial}{\partial s} G_1(s) > 0$，这个函数肯定是递增的，所以说对于任意 k，G_k 肯定是递增的。

而考虑到上面已经论证过只要满足 $\alpha = G_1(\alpha)$，肯定对于任意 k，能满足 $\alpha = G_k(\alpha)$，于是乎：

$$\alpha_1 = \lim_{n\to\infty} G_n(0) \le \lim_{n\to\infty} G_n(\alpha) = \alpha$$

也就是说对于任意符合条件的 α，他肯定是大于等于 α_1 的。于是乎 α_1 一定是最小的正解。

1.1.5. Continuous-Time Markov Chains

首先是 Poisson Process。就是隔一段时间往上 $+1$。N_t 表示 t 时刻是多少，其中 $N_0 = 0$。我们定义 T_k 为第一次到达 k 的时间。

$$N_t = \sum_{k\ge1} k \cdot \mathbb{1}_{t\in[T_{k-1}, T_k)} = \sum_{k\ge1} \mathbb{1}_{t\in[T_{k-1}, \infty)}$$

我们需要这种过程满足两个性质：

1. Independence of increments: 对于任意的 $0 \le t_1 < t_2 < \cdots < t_n$，$N_{t_1} - N_{t_0}, N_{t_2} - N_{t_1}, \cdots, N_{t_n} - N_{t_{n-1}}$ 是相互独立的。
2. Stationarity of increments: $N_{t+s} - N_s \sim N_t$。

那其实很显然这就是一个 poison distribution 嘛：

$$\mathbb{P}(N_t - N_s = k) = e^{-\lambda(t-s)} \frac{(\lambda(t-s))^k}{k!}$$

其中

$$\lambda = \lim_{h \to 0^+} \frac{1}{h} \mathbb{P}(N_h = 1)$$

然后还有就是 T_1 个 exp distribution 有关，T_n 跟 gamma distribution 有关。

$$T_n \sim \Gamma(n, \lambda) : f_{T_n}(t) = \lambda^n e^{-\lambda t} \frac{(\lambda t)^{n-1}}{\Gamma(n)}$$

然后我们把他一般化一下，得到 continuous-time Markov chain。其实核心也就是 “Memoryless”。

只不过这次的转移矩阵是一个关于时间的函数了：

$$P_{i,j}(t) = \mathbb{P}(Z_{s+t} = j \mid Z_s = i)$$

而显然 $P(0) = I$。

很显然的性质是：

$$P(s+t) = P(s)P(t) = P(t)P(s)$$

学习 Poisson process，我们来研究类似 λ 的一个跟“平均”有关的东西，我们考虑：

$$Q = \lim_{t \to 0^+} \frac{1}{t}(P(t) - P(0)) = \frac{\partial}{\partial t} P(t) \bigg|_{t=0}$$

而其实我们有（这个叫“backward Kolmogorov equation”）：

$$\begin{aligned}
\frac{\partial}{\partial t} P(t) &= \lim_{h \to 0^+} \frac{1}{h}(P(t+h) - P(t)) \\
&= \lim_{h \to 0^+} \frac{1}{h}(P(h)P(t) - P(t)) \\
&= \lim_{h \to 0^+} \frac{1}{h}(P(h) - I)P(t) \\
&= QP(t)
\end{aligned}$$

当然同理我们还可以得到（这个叫“forward Kolmogorov equation”）：

$$\frac{\partial}{\partial t} P(t) = P(t)Q$$

解这个微分方程我们有：

$$P(t) = e^{Qt} = \sum_{n=0}^{\infty} \frac{t^n}{n!} Q^n = I + \sum_{n=1}^{\infty} \frac{t^n}{n!} Q^n$$

我们记 $\lambda_{i,j} = Q_{i,j}$，其实这个 $\lambda_{i,j}$ 就和 Poisson process 那个一样了：

$$P(h) = I + hQ + \mathcal{O}(h^2)$$

也就是：

$$\mathbb{P}(X_{x+h} = j \,|\, X_t = i) = P_{i,j}(h) = \begin{cases} \lambda_{i,j} h + \mathcal{O}(h^2) & \text{if } i \neq j \\ 1 + \lambda_{i,i} h + \mathcal{O}(h^2) & \text{if } i = j \end{cases}$$

根据这个，我们对于 $i \neq j$，如果我们已经知道它下一步会到 j。我们定义停留在 i 的时间为 $\tau_{i,j}$，我们可以有：

$$\mathbb{P}\big(\tau_{i,j} > t \,\big|\, i \to j\big) = e^{-\lambda_{i,j} t}$$

以及

$$\mathbb{E}\big[\tau_{i,j} \,\big|\, i \to j\big] = \int_0^{\infty} t \lambda_{i,j} e^{-\lambda_{i,j} t} \mathrm{d}t = \frac{1}{\lambda_{i,j}}$$

当然另外还有一个性质是：

$$\sum_{j \in \mathbb{S}} \lambda_{i,j} = 0$$

就是每个点要么出去要么留在原地嘛。

然后我们考虑 τ_i，也就是停留在 i 的时间：

$$\tau_i = \min_{j \neq i} \tau_{i,j}$$

我们可以计算概率：

$$\begin{aligned} \mathbb{P}(\tau_i > t) &= \mathbb{P}\left(\min_{j \neq i} \tau_{i,j}\right) \\ &= \prod_{j \neq i} \mathbb{P}\big(\tau_{i,j} > t\big) \\ &= \exp\left(-\sum_{j \neq i} \lambda_{i,j} t\right) \\ &= e^{\lambda_{i,i} t} \end{aligned}$$

于是乎

$$\mathbb{E}[\tau_i] = \frac{1}{\sum_{j \neq i} \lambda_{i,j}} = -\frac{1}{\lambda_{i,i}}$$

1.2. Martingales

$$\mathbb{E}[Z_{n+1} \,|\, \mathcal{F}_n] = Z_n$$

坑，待填

2. Reinforcement Learning

[Berkeley CS285](#) / [Stanford CS224R](#) / [Yezhen 学长推荐嘟](#) / [Li Bo 学长推荐嘟](#)

2.1. Definition

我们把 $\langle s_t, a_t \rangle$ 打包起来，其实他就构成了一个 Markov chain:

$$p_\theta(\tau) = p_\theta(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) \cdot \mathbb{P}(s_{t+1} \mid s_t, a_t)$$

对于 learning objective：

$$\theta^* = \arg\max_\theta \mathcal{J}(\theta) = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[r(\tau)]$$

我们定义我现在在 s_t，做了 action a_t，然后按照 π 所得到的期望 reward 为

$$\mathcal{Q}^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{(s_{t'}, a_{t'}) \sim \pi}[r(s_{t'}, a_{t'}) \mid s_t, a_t]$$

然后我们定义 $\mathcal{V}$ 表示现在我在 s_t 的时候遵循 π 所得到的期望 reward：

$$\mathcal{V}^\pi(s_t) = \sum_{t'=t}^{T} \mathbb{E}_{(s_{t'}, a_{t'}) \sim \pi}[r(s_{t'}, a_{t'}) | s_t] = \mathbb{E}_{a_t \sim \pi(a_t | s_t)}[\mathcal{Q}^\pi(s_t, a_t)]$$

2.2. Types of Algorithms

Policy Gradients 跟往常一样，就是求导然后去做优化
Value-based 去直接计算最优解的 $\mathcal{Q}$ 和 $\mathcal{V}$，然后通过这个推出 π
Actor-critic 通过计算当前 $\hat{\theta}$ 的 $\hat{\mathcal{Q}}$ 和 $\hat{\mathcal{V}}$，然后通过这个去进行优化，得到最终的 π
Model-based RL 使用模型来代表各种参数，然后进行优化

2.3. Policy Gradient Algorithms

2.3.1. Direct policy differentiation

Direct policy differentiation (REINFORCE Algorithm (Williams, 1992)):

$$\begin{aligned}
\nabla_\theta \mathcal{J}(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) \, \mathrm{d}\tau \\
&= \mathbb{E}_{\tau \sim \pi_\theta(r)}[r(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau)] \\
&= \mathbb{E}_{\tau \sim \pi_\theta(r)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]
\end{aligned}$$

所以说 $r(\tau) \cdot \nabla_\theta \log \pi_\theta(r)$ 是 $\nabla_\theta \mathcal{J}(\theta)$ 的无偏估计，可惜这玩意儿的 variance 很高。

2.3.2. Reduce Variance

仔细观察这个式子，其实这玩意儿是 MLE 那个梯度对 $r(s_t, a_t)$ 加权了：

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{r\sim\pi_\theta(\tau)}\left[\sum_{t=0}^{T} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

当前我们是有：

$$\Psi_t = \sum_{t'=0}^{T} r(s_{t'}, a_{t'})$$

2.3.3. Don't Let the Past Distract You

一种简单的方法来减小 varience，是我们令

$$\Psi_t = \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

因为其实对于 a_t 来说，他做啥对于 t 之前的 reward 来说是不具有参考价值的。因此我们主要考虑后面的 reward。这玩意儿直觉上挺清楚的，但数学上想了半天才想明白为啥是对的。主要参考了这篇文章 (Achiam, 2018)。

证明的不造为啥让我想起了 MLE。主要用到的就是一个叫做 EGLP lemma 的东西 (其实好像用这个 lemma 需要积分和导数的可交换性，貌似这个在这本书 (Hogg et al., 2013) 里写挺详细的)：

$$\begin{aligned}
\mathbb{E}_{x\sim\mathbb{P}_\theta}[\nabla_\theta \log \mathbb{P}_\theta(x)] &= \int \mathbb{P}(x) \nabla_\theta \log \mathbb{P}_\theta(x)\, \mathrm{d}x \\
&= \int \mathbb{P}(x) \left(\frac{\nabla_\theta \mathbb{P}(x)}{\mathbb{P}(x)}\right) \mathrm{d}x \\
&= \nabla_\theta \int \mathbb{P}(x)\, \mathrm{d}x = 0
\end{aligned}$$

其实跟 MLE 是一样的嘛：

$$\mathbb{E}\left[\frac{\partial}{\partial\theta} \log \mathcal{L}(x \mid \theta)\right] = 0$$

其实我们是要证明嘟是：

$$\mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=0}^{T} \sum_{t'<t} r(s_{t'}, a_{t'}) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] = 0$$

也就是要证明当 $t' < t$ 这个时候：

$$\mathbb{E}_{s_t, a_t, s_{t'}, a_{t'} \sim \pi_\theta}[r(s_{t'}, a_{t'}) \nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$$

那么中心思想其实就是咋来区分 $t' < t$ 捏，我们考虑 $t' < t$ 是先 reward，再选择：

$$\mathbb{E}_{s_{t'}, a_{t'} \sim \pi_\theta}\left[r(s_{t'}, a_{t'}) \cdot \mathbb{E}_{s_t, a_t \sim \pi_\theta(\cdot \mid s_{t'}, a_{t'})}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid s_{t'}, a_{t'}]\right]$$

其实也就是当 $r(s_{t'}, a_{t'})$ 不依赖于 s_t, a_t 的时候，本身这个 $\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)]$ 他就是 0。

所以说最终结果是整个期望 0。

2.3.4. Introducing Baselines

另一个优化是我们考虑加入 baseline。这个直觉就更对了。就是我们考虑把 $r(s,a)$ 替换成 $r(s,a)-b$。因为 reward 这种东西，大家一起加多少减多少肯定都是无所谓的。

当然从数学上来讲也是 EGLP 用用易证的，这里就不多写了。但是我们 baseline 设多少最好呢？从直觉上来讲，是这个 b 让整个 r 尽量居中。接下来我们从数学上进行考虑。

我们考虑

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)\cdot(r(\tau)-b)]$$

的方差

$$\begin{aligned}\sigma^2 &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[(\nabla_\theta \log \pi_\theta(\tau)\cdot(r(\tau)-b))^2\right] - \left(\mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)\cdot(r(\tau)-b)]\right)^2 \\ &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[(\nabla_\theta \log \pi_\theta(\tau)\cdot(r(\tau)-b))^2\right] - \left(\mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)\cdot r(\tau)]\right)^2\end{aligned}$$

我们解

$$\frac{\partial}{\partial b}\sigma^2 = \frac{\partial}{\partial b}\mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[(\nabla_\theta \log \pi_\theta(\tau)\cdot(r(\tau)-b))^2\right] = 0$$

可以得到

$$b = \frac{\mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[(\nabla_\theta \log \pi_\theta(\tau))^2\cdot r(\tau)\right]}{\mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[(\nabla_\theta \log \pi_\theta(\tau))^2\right]}$$

这啥捏，这其实是 reward 的加权期望。

但其实这个 baseline 挺难算的，所以我们通常不会用这个最优的 baseline。而是去找一个相对比较好的。

2.3.5. Off-Policy Policy Gradients

之前我们做的都是 on-policy 的。但真实在训练的时候，我们很难做到稍稍改一点 θ，就重新生成一堆新的 τ。这样是非常 inefficient 的。

所以现在可能的问题是，我们没有关于 $\tau\sim\pi_\theta$ 的数据，但是我们可能有一个其他的 distribution，通过这个 distribution 来 sample 出的数据。也就是 $\tau\sim\overline{\pi}$。

我们需要使用的一个 trick 叫做 importance sampling：

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \int p(x)f(x)\,\mathrm{d}x = \int q(x)\frac{p(x)}{q(x)}f(x)\,\mathrm{d}x = \mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]$$

所以说我们的 RL objective 可以改成：

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau\sim\overline{\pi}(\tau)}\left[\frac{\pi_\theta(\tau)}{\overline{\pi}(\tau)}r(\tau)\right]$$

$$
\begin{aligned}
&= \mathbb{E}_{\tau \sim \overline{\pi}(\tau)}\left[\frac{p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)}{p(s_1)\prod_{t=1}^{T}\overline{\pi}(a_t|s_t)p(s_{t+1}|s_t,a_t)}r(\tau)\right] \\
&= \mathbb{E}_{r \sim \overline{\pi}(\tau)}\left[r(\tau)\prod_{t=1}^{T}\frac{\pi_\theta(a_t|s_t)}{\overline{\pi}(a_t|s_t)}\right]
\end{aligned}
$$

所以说我们可以推导梯度

$$
\begin{aligned}
\nabla_\theta \mathcal{J}(\theta) &= \mathbb{E}_{\tau\sim\overline{\pi}}\left[\frac{\pi_\theta(\tau)}{\overline{\pi}(\tau)}\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right] \\
&= \mathbb{E}_{r\sim\overline{\pi}}\left[\left(\prod_{t=1}^{T}\frac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right)\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\sum_{t=1}^{T} r(s_t,a_t)\right)\right] \\
&= \mathbb{E}_{r\sim\overline{\pi}}\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\prod_{t'=1}^{t}\frac{\pi_\theta(a_{t'} \mid s_{t'})}{\overline{\pi}(a_{t'} \mid s_{t'})}\right)\left(\sum_{t'=t}^{T} r(s_{t'},a_{t'})\left(\prod_{t''=t}^{t'}\frac{\pi_\theta(a_{t''} \mid s_{t''})}{\overline{\pi}(a_{t''} \mid s_{t''})}\right)\right)\right]
\end{aligned}
$$

2.4. Actor Critic Methods

2.4.1. General Idea

我们回到式子

$$
\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\Psi_{i,t}\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})
$$

我们观察这个 Ψ_t（先不考虑 baseline）：

$$
\Psi_t = \sum_{t=1}^{T} r(s_t,a_t)
$$

我们发现一件事情，就是他其实是在估计 $\mathcal{Q}^\pi(s_t,a_t)$。也就是我做了这个 policy，会得到一个什么样的结果。也就是其实这个和是 $\mathcal{Q}^\pi(s_t,a_t)$ 的一个无偏估计。

于是我们可以尝试 $\Psi_t = \mathcal{Q}^\pi(s_t,a_t)$。我的理解是，这样做可以更稳定，从而降低方差。

然后我们考虑 baseline。我们大概这么想，就是某个操作比我现在的效果好，那么 Ψ_t 要大于 0，这样子让他有梯度往下优化。如果比我现在的 π 还要垃圾，那不应该优化。

所以我们考虑 $b = \mathcal{V}^\pi(s_t)$，于是我们定义：

$$
\mathcal{A}^\pi(s_t,a_t) = \mathcal{Q}^\pi(s_t,a_t) - \mathcal{V}^\pi(s_t)
$$

也就是做了这个操作，能优化多少。

然后

$$
\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathcal{A}^\pi(s_t,a_t)\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})
$$

在 $\mathcal{A}^\pi(s_t,a_t)$ 无偏的情况下，这东西是无偏的。那我们考虑怎么算这个 $\mathcal{A}^\pi(s_t,a_t)$。

我们考虑

$$\mathcal{Q}^{\pi}(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)}[\mathcal{V}^{\pi}(s_{t+1})]$$

所以说

$$\mathcal{A}^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim p(s_t, a_t)}[r(s_t, a_t) + \mathcal{V}^{\pi}(s_{t+1}) - \mathcal{V}^{\pi}(s_t)]$$

也就是说 $r(s_t, a_t) + \mathcal{V}^{\pi}(s_{t+1}) - \mathcal{V}^{\pi}(s_t)$ 他其实是对 $\mathcal{A}^{\pi}(s_t, a_t)$ 的一个无偏估计。

也就是我们现在的问题来到了怎么搞这个 $\mathcal{V}^{\pi}$，最直接的方法是大力去做，然后求平均。

当然，更成熟的想法是我们可以训一个模型 ϕ 去预测这个额 $\mathcal{V}^{\pi}$。我们考虑 loss，一种直接的方法是：

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{\mathcal{V}}_{\phi}^{\pi}(s_{i,t}) - \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right\|^2$$

但是我们考虑有没有更稳定一些的方法，就是我们考虑用 $r(s_t, a_t) + \hat{\mathcal{V}}_{\phi}^{\pi}(s_{t+1})$ 来代替整个求和，这样其实我们会发现就是要 minimize $\mathcal{A}_{\phi}^{\pi}$：

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{\mathcal{V}}_{\phi}^{\pi}(s_{i,t}) - r(s_{i,t}) - \hat{\mathcal{V}}_{\phi}^{\pi}(s_{i,t+1}) \right\|^2$$

2.4.2. Introducing Discount Factors

有时候我们会考虑 $T \to \infty$ 的情况，我们就为了让 reward 是有穷的，我们会考虑增加一个 factor。这个其实是比较直觉的，因为现在给你一块钱和以后给你一块钱，肯定选马上要。

$$r(\tau) = \sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)$$

那么我们就需要稍稍改一改式子：

$$\mathcal{A}^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim p(s_t, a_t)}[r(s_t, a_t) + \gamma \mathcal{V}^{\pi}(s_{t+1}) - \mathcal{V}^{\pi}(s_t)]$$

这时候其实我们在估计导数的时候是有点问题的：

$$\begin{aligned}
\nabla_{\theta} \mathcal{J}(\theta) &\approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \gamma^{t-1} r(s_{i,t}, a_{i,t}) \right) \left(\sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t) \right) \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} \gamma^{t-1} \nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid s_{i,t}) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{i,t}, a_{i,t}) \right)
\end{aligned}$$

(Thomas, 2014) 专门讨论了这个问题，有空填坑。

2.4.3. Implementation Details

在真正写代码的时候，肯定不能直接用 $\mathcal{J}(\theta)$，因为否则我们这些求导的计算就白做了。我们考虑定义一个看似无意义的函数

$$\tilde{\mathcal{J}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \hat{\mathcal{A}}_\phi(s_t, a_t) \log \pi_\theta(s_t, a_t)$$

因为 $\hat{\mathcal{A}}_\phi(s_t, a_t)$ 是和 θ 无关的的常数，所以其实这个 $\nabla_\theta \tilde{\mathcal{J}}(\theta)$ 就是我们想要的 $\nabla_\theta \mathcal{J}(\theta)$。也就是说，我们其实是欺骗了自动求导程序去快速的求导。

也就是说我们要训练两个网络 θ 和 ϕ，使得

$$\begin{aligned} \phi &= \arg\min_\phi \mathbb{E}_{s_{t+1} \sim p(s_t, a_t)} \left[\frac{1}{2} \left\| r(s_t, a_t) + \gamma \hat{\mathcal{V}}_\phi(s_{t+1}) - \hat{\mathcal{V}}_\phi(s_t) \right\|^2 \right] \\ \theta &= \arg\max_\theta \mathbb{E}_{(s_t, a_t) \sim \pi_\theta} \left[\hat{\mathcal{A}}_\phi(s_t, a_t) \log \pi_\theta(a_t \mid s_t) \right] \end{aligned}$$

2.5. Generalized Advantage Estimation

(Schulman et al., 2015a)

我们考虑

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_\theta} [\Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)]$$

其中如果 $\Psi_t = \mathcal{A}^{\pi_\theta, \gamma}(s_t, a_t)$ 的话，是对的。那我们有没有啥其他更好的 Ψ_t 是对的呢？我们考虑把这一类函数 $\hat{\mathcal{A}}_t$ 叫做 γ-just 的。

首先很显然 $r(s_t, a_t) + \gamma \mathcal{V}(s_t) - V(s_{t+1})$ 肯定是 γ-just 的，因为这玩意儿

2.6. Trust Region Policy Optimization

(Schulman et al., 2015b) / tutorial (Achiam, 2018)

$$\begin{aligned} \theta_{k+1} &= \arg\max_\theta \mathcal{L}_{\theta_k}(\theta) \\ &\text{s.t. } \bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta) \le \delta \end{aligned}$$

其中

$$\begin{aligned} \mathcal{L}_{\theta_k}(\theta) &= \mathbb{E}_{(s,a) \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \hat{\mathcal{A}}^{\pi_{\theta_k}}(s, a) \right] \\ \bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta) &= \mathbb{E}_{s \sim \pi_{\theta_k}} \left[\mathbb{D}_{\mathrm{KL}} \left(\pi_{\theta_k}(\cdot \mid s) \,\middle\|\, \pi_\theta(\cdot \mid s) \right) \right] \end{aligned}$$

2.6.1. Kullback–Leibler Divergence

学过一遍忘一遍 😅 再记一下千万别忘了。

$\mathbb{D}_{\mathrm{KL}}(p \parallel q)$ 衡量的是，我以为这个 distribution 是 q，但是其实 distribution 是 q，然后我记录信息时浪费了多少。前面那个是真实分布前面那个是真实分布前面那个是真实分布啊啊啊啊啊啊啊啊啊啊啊

但是对比了半天发现用 $\bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta)$ 还是用 $\bar{\mathbb{D}}_{\mathrm{KL}}(\theta \parallel \theta_k)$ 没啥区别？我看 deepseek math 那篇文章甚至写反了。受不了了亏我研究了那么久为啥要用 $\bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta)$。

所以说

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}(p \parallel q) &= \mathcal{H}(p,q) - \mathcal{H}(p) \\ &= \mathbb{E}_{x\sim p(x)}[-\log q(x)] - \mathbb{E}_{x\sim p(x)}[-\log p(x)] \\ &= \mathbb{E}_{x\sim p(x)}\left[\log \frac{p(x)}{q(x)}\right]\end{aligned}$$

那为啥说其实是差不多的呢？看到这篇文章我总算是明白了

2.7. Proximal Policy Optimization

PPO Paper (Schulman et al., 2017) / OpenAI Tutorial (Achiam, 2018) / Hugging Face Tutorial (Simonini & Sanseviero, 2023)

TRPO 这么复杂，主要是因为他的限制和 $\mathcal{L}$ 是分开的。文章提出了两种方法将限制加入到 $\mathcal{L}$ 中：PPO-Penalty 和 PPO-Clip。

2.7.1. PPO-Clip

我们对式子略加修改：

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{(s,a)\sim\theta_k}\left[\min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\hat{\mathcal{A}}(s,a), \operatorname{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1-\epsilon, 1+\epsilon\right)\hat{\mathcal{A}}(s,a)\right)\right]$$

其中

$$\operatorname{clip}(x,l,r) = \begin{cases} l \text{ if } x < l \\ x \text{ if } l \le x \le r \\ r \text{ if } x > r \end{cases}$$

大概感性理解是这样的，正常情况下的 $\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\hat{\mathcal{A}}(s,a)$，如果我们发现 $\hat{\mathcal{A}}$ 很大，那我们考虑去增大这个 π_θ，如果很小，那么我们考虑去减小 π_θ。但是当 $\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}$ 太大或太小，意味着可能优化过多了。这时候我们就需要选择把这个权重 clip 掉。继续增大或减小 π_θ 将不再能够优化 $\mathcal{L}$。

2.7.2. PPO-Penalty

非常直觉的式子：

$$\theta_{k+1} = \arg\min_{\theta} \mathbb{E}_{(s,a)\sim\theta_k}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\hat{\mathcal{A}}(s,a) - \beta \cdot \bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta)\right]$$

但是捏，这样做肯定是不好的。因为你肯定是要优化前面那部分的嘛。那咋搞呢，就是我们考虑让 β 动起来。就是我们跑一段时间，就去康康这个

$$d = \hat{\mathbb{E}}_t\left[\bar{\mathbb{D}}_{\mathrm{KL}}(\theta_k \parallel \theta)\right]$$

直觉上就是这玩意儿如果大了，就说明太远了，那我们得稍微调高一下 β；否则的话，我们需要做更细致的优化，那么我们得把 β 搞低。

所以说我们可以定一个阈值 d_{targ}

$$\beta \leftarrow \begin{cases} \frac{\beta}{2} & \text{if } d < \frac{d_{\text{targ}}}{1.5} \\ \beta & \text{if } \frac{d_{\text{targ}}}{1.5} \le d \le d_{\text{targ}} \times 1.5 \\ 2\beta & \text{if } d > d_{\text{targ}} \times 1.5 \end{cases}$$

2.7.3. Learning to summarize from human feedback

paper (Ouyang et al., 2022a) / demo

TL;DR Dataset(Völske et al., 2017): 一个 summarize 任务，主要是要对 reddit 的 summarize。

Reward model: $r_\phi(x, y)$ 只对最终的结果给 reward。所以说其实 γ 就没啥意义了，我们直接大力 $\gamma = 1$。Reward model 的数据是一大堆 preference data $\langle x, y_0, y_1 \rangle$，其中 y_0 更被喜欢。

$$\mathcal{L}(\phi) = -\mathbb{E}_{(x,y_0,y_1)\sim\mathcal{D}}\left[\log \sigma\left(r_\phi(x, y_0) - r_\phi(x, y_1)\right)\right]$$

最终我们训练模型的时候，是用 PPO-Penalty，也就是

$$R(x, y) = r(x, y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

然后我们就发现为啥这个 $\mathbb{E}_{y\sim\pi_\theta}\left[\log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right]$ 外面这个期望咋被吞掉了。思考了一下大概知道为啥了，就是好像两个确实是等价的。因为其实我们有个隐形条件，就是 $y \sim \pi_\theta$，也就是说，其实这个 R 应该是：

$$\begin{aligned} R(\theta) &= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}\left[r(x, y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right] \\ &= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}[r(x, y)] - \beta \cdot \mathbb{E}_{x\sim\mathcal{D}}[\mathbb{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})] \end{aligned}$$

对于这个 reward model，文中没有提及，但是联想到有篇文章叫做 Bradley-Terry Model (Bradley & Terry, 1952)。这个在 DPO 那篇文章里提了很多。这个 notes 写的感觉非常好。

n 个球队，每个球队有个 strength β_i，i 跟 j 打的赢率

$$p_{ij} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}$$

但是有时候 (i, j) 并不是可交换嘟，所以说我们定义

$$p_{ij} = \frac{e^{\alpha + \beta_i - \beta_j}}{1 + e^{\alpha + \beta_i - \beta_j}}$$

当然咱这儿没有 α。其实这个 r 预测的就是这个 β 嘛。

2.7.4. InstructGPT

paper (Ouyang et al., 2022b)

对于 reward model，他把两个改成了 K 个。就是他直接让排 K 个 y。这样的话

$$\mathcal{L}(\phi) = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x, y_w \succ y_l) \sim \mathcal{D}} \left[\log \sigma\left(r_\phi(x, y_w) - r_\phi(x, y_l)\right)\right]$$

然后 RL 部分，

2.8. Direct Preference Optimization

paper (Rafailov et al., 2024) / HF Docs / HF Tutorial

2.8.1. Key Ideas

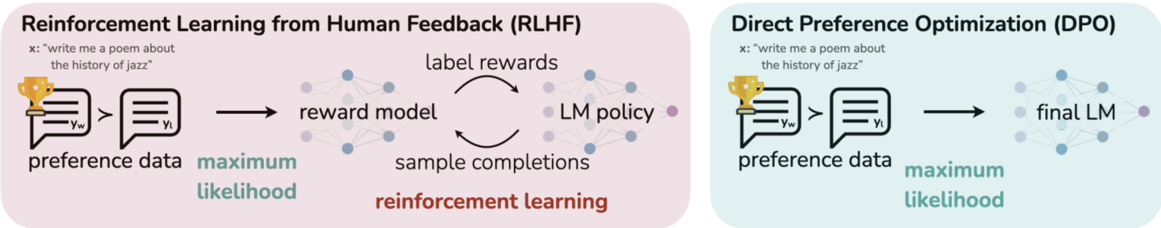

Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

我的大概猜测是，作者主要有两个 key observation：

1. Preference data 本身可以体现一种 reward function（BT model）
2. π 可以一一对应到 r，使得这个 r 能够训练出的最优 policy 是 π

整条线大概是 $\pi \leftrightarrow r \to p_\succ \to \mathcal{L}$。

也就是说，我们其实可以通过调整 π 来保证这个与 π 一一对应的 r 去吻合数据。

具体地，对于我们的 learning objective：

$$\arg\max_\theta \mathbb{E}[]$$

2.8.2. Paper Details

2.8.3. Other Details

In eq 12,

$$\begin{aligned}
&\min_\pi \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y \mid x)} \left[\log\left(\frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right) - \frac{1}{\beta} r(x, y)\right] \\
&= \min_\pi \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y \mid x)} \left[\log\left(\frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)}\right)\right] \\
&= \min_\pi \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y \mid x)} \left[\log\left(\frac{\pi(y \mid x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)}\right) - \log Z(x)\right]
\end{aligned}$$

While

$$Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

比较有意思的点是，他把外面的 $\frac{1}{\beta} r(x, y)$ 强行塞进了 $\log$ 里面来构造一个新的 policy：

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

酱紫的话，里面的 $\log\left(\frac{\pi(y \mid x)}{\pi^*(y \mid x)}\right)$ 就构成了一个新的 $\mathbb{D}_{\text{KL}}$

$$\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y \mid x)} \left[\log\left(\frac{\pi(y \mid x)}{\pi^*(y \mid x)}\right)\right] = \mathbb{D}_{\text{KL}}(\pi \parallel \pi^*)$$

2.9. Reward Hacking

https://lilianweng.github.io/posts/2024-11-28-reward-hacking/

2.10. Group Relative Policy Optimization

Deepseek Math (Shao et al., 2024) / Deepseek R1 (Guo et al., 2025)

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{\|y_i\|} \sum_{t=1}^{\|y_i\|} \left(f\left(\frac{\pi_\theta(y_{i,t} \mid q, y_{i,1\ldots t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid q, y_{i,1\ldots t})}, \hat{A}_{i,t}\right) - \beta \cdot \mathbb{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})\right)\right]$$

其中

$$f\left(w, \hat{A}_{i,t}\right) = \min\left\{w\hat{A}_{i,t}, \text{clip}(w, 1-\varepsilon, 1+\varepsilon)\hat{A}_{i,t}\right\}$$

欸为啥看着这么像 offline learning 捏。冷静分析发现他式子里居然有 $\pi_{\theta_{\text{old}}}$ 和 π_{ref} 两个东西。所以他其实是

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \eta \nabla_\theta \mathcal{J}(\theta, \theta_{\text{old}})$$

酱紫一个 update。而初始模型是 θ_{ref}。这俩确实应该是不一样的。

欸那为啥就他要这么写呢？所以他好像想 fix 两个事情：

1. $\theta_{\text{old}} \to \theta_{\text{new}}$ 也就是单步不要变太多，这个用 clip 维护
2. $\theta_{\text{ref}} \to \theta_{\text{new}}$ 也就是全局不要变太多，这个用 KL 维护

我超那这个 clip 的效果和直接修改学习率会有啥影响吗？其实有点没想明白 qaq 我大概毛猜猜好像是确实 clip 会好一点，就是因为他对微小的变化察觉好像确实会明显一点。因为调 learning rate 的话，你这个小的梯度一乘好像确实就丢了。但其实他现在就是对大的变化做了个 clip，所以细微的差异还是会进行一定修改的。

2.11. Paper Reading

一些读了但是没有归类的论文

2.11.1. A Comparative Study of Foundation Model Post-training

paper (Chu et al., 2025) / twitter

Bibliography

Achiam, J. (2018). *Spinning Up in Deep Reinforcement Learning*.

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, *39*(3/4), 324–345.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., & Ma, Y. (2025). SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. *Arxiv Preprint Arxiv:2501.17161*.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., & others. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Arxiv Preprint Arxiv:2501.12948*.

Hogg, R. V., McKean, J. W., Craig, A. T., & others. (2013). *Introduction to mathematical statistics*. Pearson Education India.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., & others. (2022a). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, *35*, 27730–27744.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R. (2022b). Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems: Vol. 35. Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

Privault, N. (2013). Understanding markov chains. *Examples and Applications, Publisher Springer-Verlag Singapore*, *357*, 358.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., & Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, *36*.

Schulman, J., Levine, S., Moritz, P., Jordan, M., & Abbeel, P. (2015b). Trust region policy optimization. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015a). High-dimensional continuous control using generalized advantage estimation. *Arxiv Preprint Arxiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *Arxiv Preprint Arxiv:1707.06347*.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., & others. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Arxiv Preprint Arxiv:2402.03300*.

Simonini, T., & Sanseviero, O. (2023,). *The Hugging Face Deep Reinforcement Learning Class*. GitHub.

Thomas, P. (2014). Bias in natural actor-critic algorithms. *International Conference on Machine Learning*, 441–448.

Völske, M., Potthast, M., Syed, S., & Stein, B. (2017). TL;DR: Mining Reddit to Learn Automatic Summarization. In L. Wang, J. C. K. Cheung, G. Carenini, & F. Liu (Eds.), *Proceedings of the Workshop on New Frontiers in Summarization: Proceedings of the Workshop on New Frontiers in Summarization*. https://doi.org/10.18653/v1/W17-4508

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.