多智能体强化学习最小均值方差公式推导

2024 年 12 月 6 日

王钰

1 公式推导

注：由于不考虑复杂环境，所以没有考虑DQN和VDN中的RNN，也没有考虑CTD(λ)。

1.1 MDP

一个MDP可用一个五元组$(S, A, P^a_{s,s'}, R, \gamma)$表示。其中$S$（state）是状态空间，$A$（action）是动作空间，$P^a_{s,s'}$是在状态$s$下采取动作$a$可以转换到状态$s'$的概率，$R_t$是智能体在$t$时可以获得的奖励，在环境中，智能体每次实际获得的奖励记为r_t，γ是折扣因子。

定义策略为：

$$\pi : P(a|s)$$

策略为从状态s选择动作a的概率。

定义回报（随机变量）为：

$$G^\pi(s_t) = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$$

，

回报为遵从策略π，智能体从状态s_t开始到结束，可以获得的累计折扣奖励。

在MDP中，算法的目标是得到最优的策略π^*，使得回报$G^\pi(s_t)$最大。

1.2 Value Functions

定义状态价值函数$V^{\pi}(s_t) = \mathbb{E}[G^{\pi}(s_t)]$，其中$G^{\pi}(s_t)$是指遵从策略$\pi$，智能体从状态$s_t$开始到结束，可以获得的累计折扣奖励。对状态价值函数可进行进一步推导，得到状态价值函数的贝尔曼方程：

$$\begin{aligned} V^{\pi}(s_t) &= \mathbb{E}[G^{\pi}(s_t)] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)] \\ &= \mathbb{E}[R_{t+1} + \gamma G^{\pi}(s_{t+1})] \\ &= \mathbb{E}[R_{t+1}] + \gamma \mathbb{E}[G^{\pi}(s_{t+1})] \end{aligned}$$

定义状态-动作价值函数$Q^{\pi}(s_t, a_t) = \mathbb{E}[G^{\pi}(s_t, a_t)]$，其中$G^{\pi}(s_t, a_t)$是指智能体从状态$s_t$开始，选取动作$a_t$，然后遵从策略$\pi$直到结束，可以获得的累计折扣奖励。同理，对状态-动作价值函数可进行进一步推导，得到状态-动作价值函数的贝尔曼方程：

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathbb{E}[G^{\pi}(s_t, a_t)] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)] \\ &= \mathbb{E}[R_{t+1} + \gamma G^{\pi}(s_{t+1}, a_{t+1})] \\ &= \mathbb{E}[R_{t+1}] + \gamma \mathbb{E}[G^{\pi}(s_{t+1}, a_{t+1})] \end{aligned}$$

由于价值函数是回报的期望，所以在最优的策略π^*时，有最优状态价值函数$V^{\pi^*}(s_t)$ 和最优动作-状态价值函数$Q^{\pi^*}(s_t, a_t)$。因此，找到使得回报$G^{\pi}(s_t)$最大的最优策略π^*的过程就相当于对价值函数进行优化的过程。

1.3 TD learning

强化学习中，一种更新价值函数的方法是TD学习。

根据状态价值函数的贝尔曼方程，可得：

$$V^{\pi}(s_t) = r_{t+1} + \gamma V^{\pi}(s_{t+1})$$

其中$r_{t+1} + \gamma V^{\pi}(s_{t+1})$被称为TD目标，是即时奖励和下一个状态的价值的无偏估计，用于近似贝尔曼方程中的期望值。

根据TD目标，可定义TD误差δ_{vt}为$r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$。 TD误差项表示实际获得的回报与预期回报之间的差异。

利用TD误差，可以推导出更新当前的状态价值函数的公式：

$$\begin{aligned} V^{\pi}(s_t) &= V^{\pi}(s_t) + \alpha \times \delta_{vt} \\ &= V^{\pi}(s_t) + \alpha(r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)) \end{aligned}$$

其中α是学习率，控制状态价值函数的更新速度。

同理，根据状态-动作价值函数的贝尔曼方程，可得：

$$Q^{\pi}(s_t, a_t) = r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})$$

同理，可以推导出更新当前状态-动作价值函数的公式：

$$Q^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) + \alpha(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t))$$

其中α是学习率，控制状态-动作价值函数的更新速度；可定义TD误差δ_{qt}为$r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)$。

1.4 CTD

1.4.1 用于计算方差的价值函数推导

根据定义$V^{\pi}(s_t) = \mathbb{E}[G^{\pi}(s_t)]$，前面的推导可以获得计算在状态$s_t$下的回报的均值。接下来，要推导出可以计算在状态$s_t$下的回报的方差$\bar{V}^{\pi}(s_t)$。

首先，对回报的方差$\mathbb{V}[G^{\pi}(s_t)]$进行推导：

$$
\begin{aligned}
\mathbb{V}[G^\pi(s_t)] &= \mathbb{E}[G^\pi(s_t) - \mathbb{E}[G^\pi(s_t)]]^2 \\
&= \mathbb{E}[r_{t+1} + G^\pi(s_{t+1}) - \mathbb{E}[G^\pi(s_t)]]^2 \\
&= \mathbb{E}[r_{t+1} + G^\pi(s_{t+1}) - V^\pi(s_t)]^2 \\
&= \mathbb{E}[r_{t+1} + G^\pi(s_{t+1}) - V^\pi(s_t) + \gamma V^\pi(s_{t+1}) - \gamma V^\pi(s_{t+1})]^2 \\
&= \mathbb{E}[(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)) + (G^\pi(s_{t+1}) - \gamma V^\pi(s_{t+1}))]^2 \\
&= \mathbb{E}[r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)]^2 + \mathbb{E}[G^\pi(s_{t+1}) - \gamma V^\pi(s_{t+1})]^2 \\
&\quad + 2\mathbb{E}[r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)]\mathbb{E}[G^\pi(s_{t+1}) - \gamma V^\pi(s_{t+1})] \\
&= \mathbb{E}[\delta_{vt}]^2 + \mathbb{E}[G^\pi(s_{t+1}) - V^\pi(s_{t+1}) + (1-\gamma)V^\pi(s_{t+1})]^2 \\
&\quad + 2\mathbb{E}[\delta_{vt}]\mathbb{E}[G^\pi(s_{t+1}) - V^\pi(s_{t+1}) + (1-\gamma)V^\pi(s_{t+1})] \\
&= \mathbb{E}[\delta_{vt}]^2 + \mathbb{E}[G^\pi(s_{t+1}) - V^\pi(s_{t+1})]^2 + \mathbb{E}[(1-\gamma)V^\pi(s_{t+1})]^2 + 2\mathbb{E}[G^\pi(s_{t+1}) - V^\pi(s_{t+1})] \\
&\quad + 2\mathbb{E}[\delta_{vt}]\mathbb{E}[G^\pi(s_{t+1}) - V^\pi(s_{t+1})] + 2\mathbb{E}[\delta_{vt}]\mathbb{E}[(1-\gamma)V^\pi(s_{t+1})] \\
&= \mathbb{E}[\delta_{vt}]^2 + \mathbb{V}[G^\pi(s_{t+1})] + (1-\gamma)V^\pi(s_{t+1})^2 + 0 \\
&\quad + 2\mathbb{E}[\delta_{vt}] \times 0 + 2\mathbb{E}[\delta_{vt}](1-\gamma)V^\pi(s_{t+1}) \\
&= \mathbb{E}[\delta_{vt}]^2 + \mathbb{V}[G^\pi(s_{t+1})] + (1-\gamma)V^\pi(s_{t+1})(2\mathbb{E}[\delta_{vt}] + V^\pi(s_{t+1})) \\
&\approx \mathbb{E}[\delta_{vt}]^2 + \mathbb{V}[G^\pi(s_{t+1})]
\end{aligned}
$$

（论文里这里每行推导都可以细讲）。最后，为了推导出关于回报的方差的贝尔曼方差，这里令$\gamma \approx 1$，所以，这里可以认为$(1-\gamma)V^\pi(s_{t+1})(2\mathbb{E}[\delta_{vt}]+V^\pi(s_{t+1}))$是一个远小于$\mathbb{E}[\delta_{vt}]^2 + \mathbb{V}[G^\pi(s_{t+1})]$的数，因此在最终的公式中，忽视了$(1-\gamma)V^\pi(s_{t+1})(2\mathbb{E}[\delta_{vt}] + V^\pi(s_{t+1}))$这项，最终得到回报的方差的贝尔曼方程。

回顾TD学习，用于计算回报的均值的状态价值函数的贝尔曼方程为：$V^\pi(s_t) = r_{t+1} + \gamma V^\pi(s_{t+1})$；用于计算回报的均值的状态价值函数的更新公式为：$V^\pi(s_t) = V^\pi(s_t) + \alpha(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$。因此，根据以上关于$\mathbb{V}[G^\pi(s_t)]$的推导，可得出用于计算回报的方差的状态价值函数$\bar{V}^\pi(s_t)$的更新公式为：

$$
\bar{V}^\pi(s_t) = \bar{V}^\pi(s_t) + \bar{\alpha}(\delta_{vt}^2 + \bar{V}^\pi(s_{t+1}) - \bar{V}^\pi(s_t))
$$

在这里，我们采用了TD学习中的常用技术，用智能体与环境交互中

实际计算的δ_{vt}^2来代替$\mathbb{E}[\delta_{vt}]^2$。类似的，可以得到用于计算回报的方差的状态-动作价值函数$\bar{Q}^\pi(s_t, a_t)$的更新公式为：

$$\bar{Q}^\pi(s_t, a_t) = \bar{Q}^\pi(s_t, a_t) + \bar{\alpha}(\delta_{qt}^2 + \bar{Q}^\pi(s_{t+1}, a_{t+1}) - \bar{Q}^\pi(s_t, a_t))$$

1.4.2 级联时序差分的动作选择策略

这里的策略是ϵ-贪心策略的改进版本。在所有动作集合中，我们认为使得回报的均值和方差的线性组合最大的动作是最优动作。记最优动作为a^*，则：

$$a^* = \arg\max_a (Q(s, a) - \zeta\sqrt{\bar{Q}(s, a)})$$

其中最优动作a^*即为使$Q(s, a) - \zeta\sqrt{\bar{Q}(s, a)}$最小的动作。其中将均值和方差的线性组合作为优化目标，ζ是一个大于0的数，ζ越大，表示智能体更倾向于选择带来的回报的随机性更小的动作。

在我们的策略中，设置参数$\epsilon \in (0, 1)$；智能体有ϵ的概率选择最有动作，有$1 - \epsilon$的概率随机选择一个可行的动作，这样就可以保证探索与利用的平衡。更进一步，在训练中，可以随着训练轮数的增加逐步增大ϵ的值，在训练完成后，将ϵ的值设置为很接近1的数。

1.5 CTD+DQN

1.5.1 基于DNN的时序差分

在状态-动作空间过大时，维护一张Q表需要大量内存；因此，引入神经网络（Neural Network, NN）来拟合状态-动作价值函数。也就是说，让当前环境s_t作为NN的输入，NN输出所有的动作的对应的状态-动作价值函数。设神经网络的参数为θ_e，则引入NN对状态-动作价值函数进行拟合可以表示为$Q^\pi(s_t, a_t, \theta_e)$。

根据TD公式，可以得到在NN情况下的TD误差为：$r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}, \theta_e) - Q^\pi(s_t, a_t, \theta_e)$。

在时序差分中，TD误差是需要被优化的项，即让每次智能体与环境交互的TD误差尽可能小；这与NN的损失函数的设计十分相似。因此，为了优化网络参数θ_e，可以将TD误差作为网络的损失函数。然而，在训练过程

中，网络的参数的更新会导致$Q^{\pi}(s_{t+1}, a_{t+1}, \theta_e)$和$Q^{\pi}(s_t, a_t, \theta_e)$的值同时变化，这样会导致最终网络预测的Q值越来越大。因此，采用两个相同架构的网络分别预测当前和下一步的状态-动作价值函数。这里把预测当前状态-动作价值函数的网络称作预测网络，参数为θ_e；预测下一步状态-动作价值函数的网络称作目标网络，参数为θ_e^-。在更新参数时，只更新预测网络的参数，特定步数后，再把预测网络的参数拷贝给目标网络。

为了用批次训练加快训练速度，采用经验回放的方法。首先构建一个经验池$U = u_1, u_2, \cdots, u_n$，经验池中的每一条经验是智能体与环境的一次交互，即$u_t = (s_t, a_t, r_{t+1}, s_{t+1})$。智能体与环境的每次交互都会被放到经验池中，然后从经验池中随机抽取一批次的数据，对网络进行训练。因此，最终网络的损失函数为：

$$\mathcal{L} = \mathbb{E}_{(s_t, a_t, r_{t+1}, s_{t+1}) \in U}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}, \theta_e^-) - Q^{\pi}(s_t, a_t, \theta_e)]^2$$

根据损失函数，选取适当的优化器和学习率，即可对网络进行训练。

1.5.2 基于DNN的级联时序差分

上一部分介绍了引入NN拟合智能体获得回报的期望的状态-动作价值函数，现在介绍引入NN拟合智能体获得回报的方差的状态-动作价值函数。设预测智能体获得的回报的方差的网络参数为θ_v，预测的方差为$\bar{Q}^{\pi}(s_t, a_t, \theta_v)$则根据之前推导的级联时序差分，关于方差的误差为：

$$(r_{t+1}+\gamma Q^{\pi}(s_{t+1}, a_{t+1}, \theta_e)-Q^{\pi}(s_t, a_t, \theta_e))^2+\gamma\bar{Q}^{\pi}(s_{t+1}, a_{t+1}, \theta_v)-\bar{Q}^{\pi}(s_t, a_t, \theta_v)$$

同理，采用两个相同架构的预测网络和目标网络来对当前和下一状态的方差的状态-动作价值函数进行预测。可推导出网络的损失函数为：

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{(s_t, a_t, r_{t+1}, s_{t+1}) \in U}[&(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}, \theta_e^-) - Q^{\pi}(s_t, a_t, \theta_e))^2 \\ &+\gamma\bar{Q}^{\pi}(s_{t+1}, a_{t+1}, \theta_v) - \bar{Q}^{\pi}(s_t, a_t, \theta_v)]^2\end{aligned}$$

根据以上推导的损失函数，即可对网络进行训练。

1.6 DEC-POMDP

这部分讲分布式部分可观测马尔可夫决策过程，主要是各种定义。暂忽略，到时候论文中再写。

1.7 完全合作式MARL（CTD+VDN）

在完全合作式MARL中，所有智能体获得的奖励是一致的，即智能体合作完成团队目标时，所有智能体都会获得相同的奖励，也叫做团队奖励。假设有n个智能体，则团队的回报的期望的价值函数为：

$Q_{total}((o_1, o_2, \cdots, o_n), (a_1, a_2, \cdots, a_n), (\theta_{e,1}, \theta_{e,2}, \cdots, \theta_{e,n}))$。

智能体i的价值函数为：$Q_i(o_i, a_i, \theta_{e,i})$。

根据VDN，团队价值函数可以近似分解为各个智能体的价值函数之和，即：

$$Q_{total} \approx \sum_{i=1}^{n} Q_i$$

同理，团队的回报的方差的价值函数为：

$\bar{Q}_{total}((o_1, o_2, \cdots, o_n), (a_1, a_2, \cdots, a_n), (\theta_{v,1}, \theta_{v,2}, \cdots, \theta_{v,n}))$。

智能体i的方差价值函数为：$\bar{Q}_i(o_i, a_i, \theta_{e,i})$。

则团队方差价值函数可以近似分解为各个智能体的方差价值函数之和，即：

$$\bar{Q}_{total} \approx \sum_{i=1}^{n} \bar{Q}_i$$

因此，团队的价值函数可以分解为个体的价值函数之和，把团队奖励看作每个智能体的奖励，就可以对每个智能体进行训练，从而得到每个智能体的最优策略。