MADDPG

林越 2022.7.26

背景

- SARL很成功：应用案例
- MARL很重要：很多实用场景
- （问题）SARL方法在MA场景不直接适用
- （方法）设计了个新算法MADDPG，用CTDE框架
    - CT的时候critic收到别人policy的信息
    - 合作场景和对抗场景都可以用

背景

SARL很成功

- game playing: Atari(2015 nature), alpha GO(2016 nature)
- robotics:
    - 学个policy，图像映射到力矩，控制机械臂(2015)
    - 苏黎世的ANYmal，移动机器人，很多步态都是强化学习学出来的
- data center cooling
    - DeepMind通过RL为Google机房节能（没细看）

背景

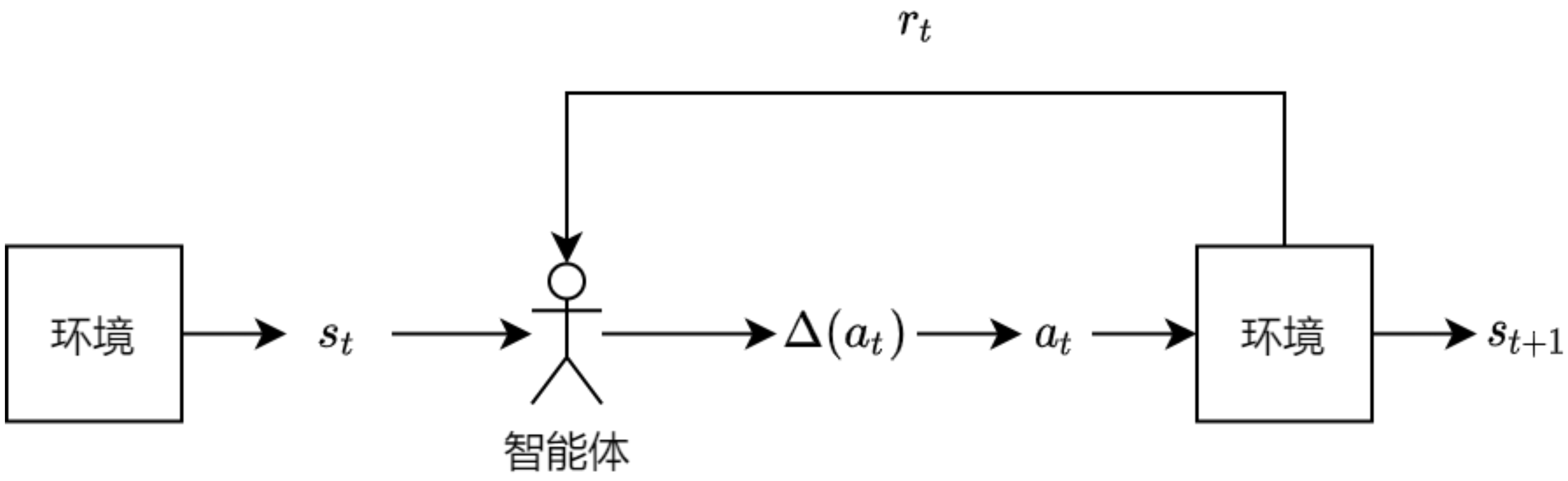
MARL很重要

- multi-robot control：多个移动机器人探索环境(2012)
- the discovery of communication and language
- multiplayer games
- social dilemmas

- hierarchical RL
- self-play

single agent

- 和环境互动，获得奖励
- 改变自己行为，增大自己获得的奖励期望



single agent

- value-based (critic)
  - 自己有个function：输入observation，输出自己在这个obs的收益期望值
    - 优化目标：让这个function输出的值接近真实值（距离小）
    - 随着transition变多，数据也越多；数据来了就迭代更新一次function
    - 分析MDP，可以用bellman equation递推；用incremental方法推真实值
  - 决策也是用的critic
    - 哪个动作的收益期望大就选哪个动作（探索方面可以加epsilon-greedy）

single agent

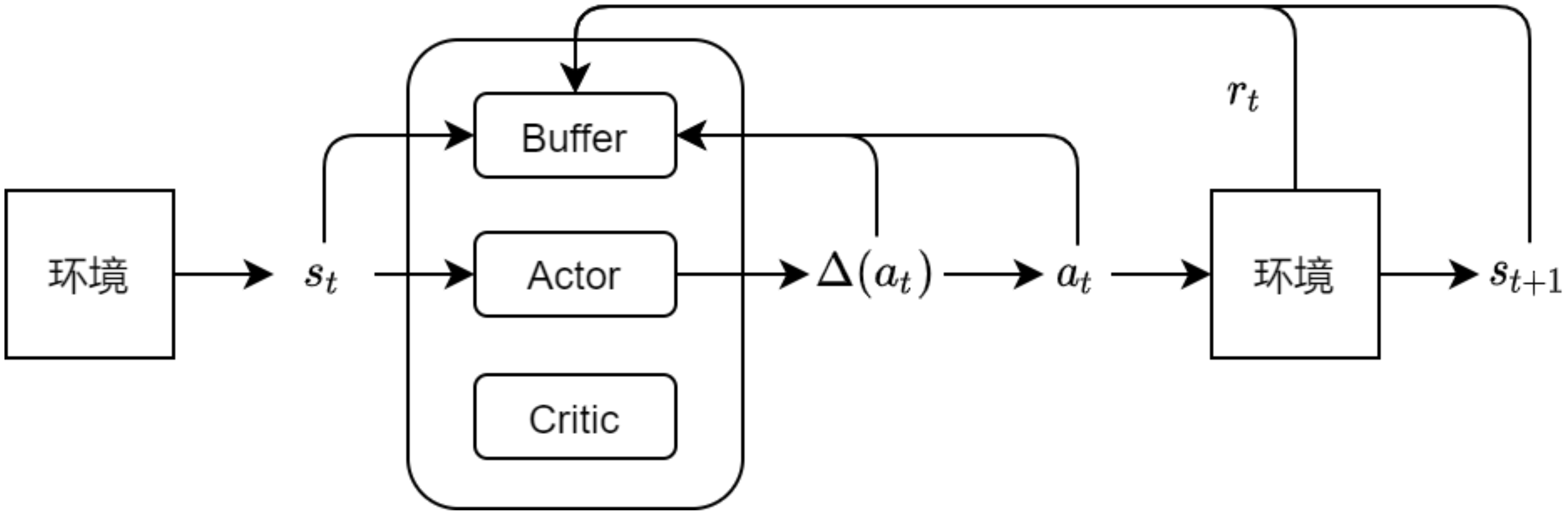
- policy-based (actor / policy gradient)
  - 把RL看成一个优化问题，agent要通过改变policy，来优化自己的收益期望
  - policy被参数化，policy是一个function（在AC里也叫actor）

  - 学习过程是policy function梯度上升过程：求收益期望对policy参数的梯度
  - 然后有policy gradient theorem相关的推导，推出了这个梯度是什么样的

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^{\pi_\theta}(s,a)\right]$$

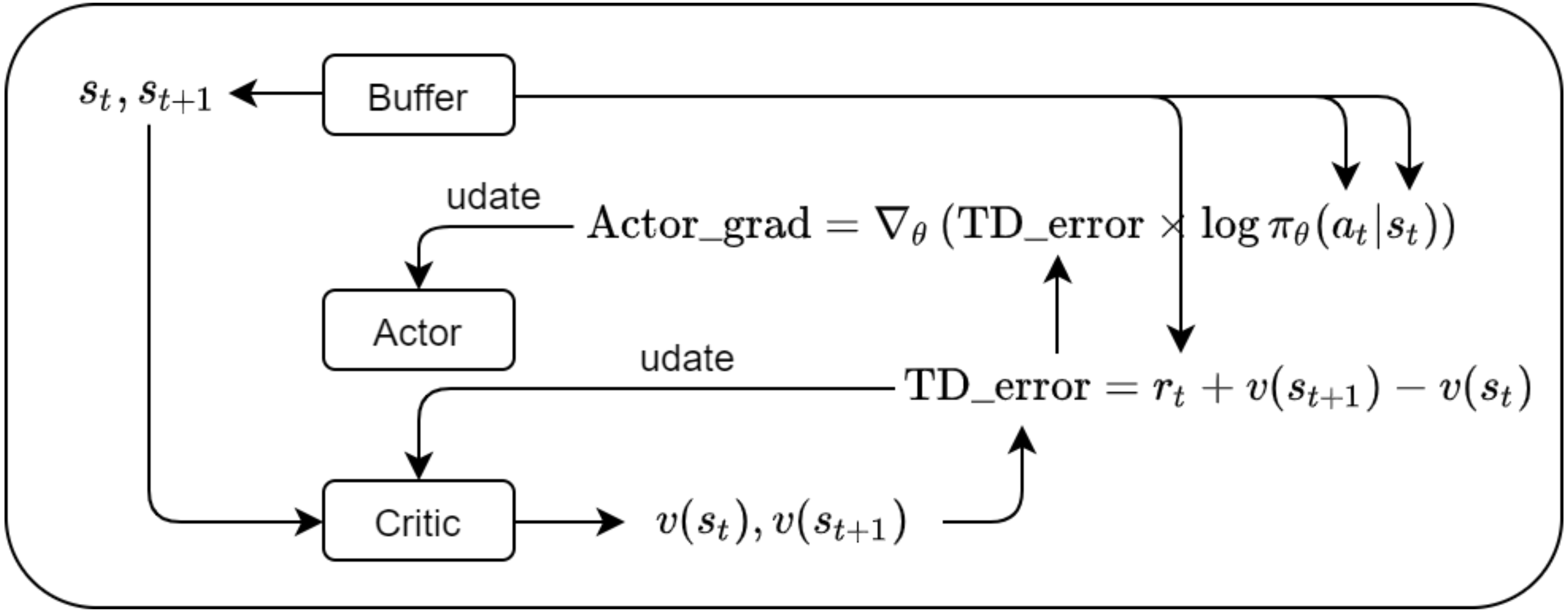
single agent

- actor-critic
  - critic还是评价当前自己在当前state的收益期望，但是不参与决策（不参与execution），只在更新actor时要用到



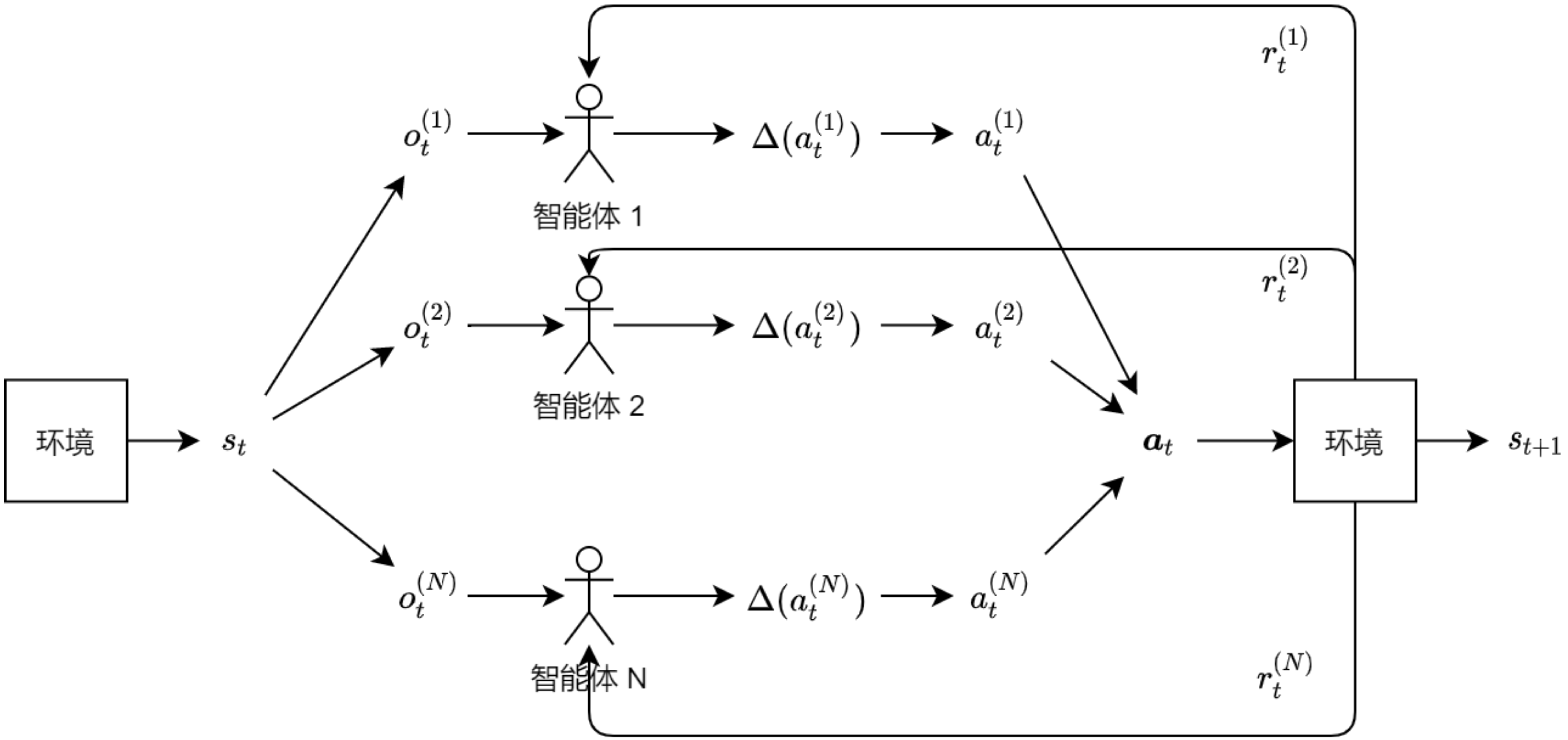
single agent

- actor-critic
    - critic还是评价当前自己在当前state的收益期望，但是不参与决策（不参与execution），只在更新actor时要用到



从SARL到MARL

- 每个agent都用个SARL算法来控制
- 各自学各自的，不管别人，这个叫independent



MARL

independent

- independent Q-learning
  - independent且每个SARL都是Q- learning
  - 引用的论文：M.Tan.Multi-agent reinforcement learning: **Independent vs. cooperative** agents. ICML. 1993.
    - 其实IQL只是个benchmark，这篇里面还有其他加了机制**来合作**的实验
      - sharing sensation
      - sharing episodes
      - sharing learned policy

MARL

independent方法的问题

- 如果每个agent的SARL都是用的value-based方法
  - 对于每个agent的SARL来说，环境都是non-stationary的
    - 就算我的策略没变、选的动作也一样，别人的策略/动作变了，我获得的奖励也不同，下一个观测也会变
    - 相当于每个人把别人看成环境的一部分
    - 不满足Markov的假设，不能保证收敛
  - experience replay buffer也用不了；选了同样的动作，结果去哪的概率不确定
    - $P(s'|s, a, \boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_N) \neq P(s'|s, a, \boldsymbol{\pi}'_1, ..., \boldsymbol{\pi}'_N)$ when any $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i$.

MARL

independent方法的问题

- 如果每个agent的SARL都是用的policy-based方法
    - 在SARL中，policy gradient方法，对gradient的估计，方差本来就大
    - 在MARL中这个现象加剧了
    - 奖励是很多人的动作一起决定的，但是各自优化时只考虑自己动作的影响
    - 他们提了个proposition，然后有个小证明

MARL

independent policy-based方法，方差很大

- N个人，每个人动作为0或1
- 所有人选择相同的动作，则奖励每人1，否则奖励每人0
- 每个人开始都是0.5的概率选择动作1，记为θ_i
- 每个人都用的policy gradient方法
- 则（要证明）：

$$P(\langle \hat{\nabla} J, \nabla J \rangle > 0) \propto (0.5)^N$$

MARL

independent policy-based方法，方差很大

- N个人，每个人动作为0或1
- 所有人选择相同的动作，则奖励每人1，否则奖励每人0
- 每个人开始都是0.5的概率选择动作1，记为θ_i
- 每个人都用的policy gradient方法

$$P(a_i) = {\theta_i}^{a_i}(1-\theta_i)^{1-a_i}$$

$$\log P(a_i) = a_i \log \theta_i + (1-a_i)\log(1-\theta_i)$$

MARL

independent policy-based方法，方差很大

$$P(a_i) = {\theta_i}^{a_i}(1-\theta_i)^{1-a_i}$$

$$\log P(a_i) = a_i \log \theta_i + (1-a_i)\log(1-\theta_i)$$

The policy gradient estimator (from a single sample) is:

$$\begin{aligned}
\frac{\hat{\partial}}{\partial \theta_i} J &= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} \log P(a_1, \ldots, a_N) \\
&= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} \sum_i a_i \log \theta_i + (1-a_i)\log(1-\theta_i) \\
&= R(a_1, \ldots, a_N) \frac{\partial}{\partial \theta_i} (a_i \log \theta_i + (1-a_i)\log(1-\theta_i)) \\
&= R(a_1, \ldots, a_N) \left(\frac{a_i}{\theta_i} - \frac{1-a_i}{1-\theta_i} \right)
\end{aligned}$$

MARL

independent policy-based方法，方差很大

The policy gradient estimator (from a single sample) is:

$$\begin{aligned}
\frac{\hat{\partial}}{\partial\theta_i}J &= R(a_1,\ldots,a_N)\frac{\partial}{\partial\theta_i}\log P(a_1,\ldots,a_N) \\
&= R(a_1,\ldots,a_N)\frac{\partial}{\partial\theta_i}\sum_i a_i\log\theta_i + (1-a_i)\log(1-\theta_i) \\
&= R(a_1,\ldots,a_N)\frac{\partial}{\partial\theta_i}(a_i\log\theta_i + (1-a_i)\log(1-\theta_i)) \\
&= R(a_1,\ldots,a_N)\left(\frac{a_i}{\theta_i} - \frac{1-a_i}{1-\theta_i}\right)
\end{aligned}$$

For $\theta_i = 0.5$ we have:

$$\frac{\hat{\partial}}{\partial\theta_i}J = R(a_1,\ldots,a_N)\,(2a_i - 1)$$

MARL

independent policy-based方法，方差很大

And the expected reward can be calculated as:

$$\mathbb{E}(R) = \sum_{a_1,\ldots,a_N} R(a_1,\ldots,a_N)(0.5)^N$$

Consider the case where $R(a_1,\ldots,a_N) = \mathbf{1}_{a_1=\cdots=a_N=1}$. Then

$$\mathbb{E}(R) = (0.5)^N$$

and

$$\mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J) = \frac{\partial}{\partial\theta_i}J = (0.5)^N$$

MARL

independent policy-based方法，方差很大

The variance of a single sample of the gradient is then:

$$\mathbb{V}(\frac{\hat{\partial}}{\partial\theta_i}J) = \mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J^2) - \mathbb{E}(\frac{\hat{\partial}}{\partial\theta_i}J)^2 = (0.5)^N - (0.5)^{2N}$$

What is the probability of taking a step in the right direction? We can look at $P(\langle\hat{\nabla}J, \nabla J\rangle > 0)$. We have:

$$\langle\hat{\nabla}J, \nabla J\rangle = \sum_i \frac{\hat{\partial}}{\partial\theta_i}J \times (0.5)^N = (0.5)^N \sum_i \frac{\hat{\partial}}{\partial\theta_i}J,$$

so $P(\langle\hat{\nabla}J, \nabla J\rangle > 0) = (0.5)^N$. Thus, as the number of agents increases, the probability of taking a gradient step in the right direction decreases exponentially. □

MARL

independent policy-based方法，方差很大

- 文章里是这么说的：
    - 虽然这是个小的构造出来的例子，但是有助于说明（it serves to illustrate that）有一些简单环境，随着agent数量变多，independent方法用policy based会变得很难
    - 特别是在奖励稀疏的时候

MARL

针对问题他们提出了MADDPG

- （训练时）每个人各自更新critic，所以每个人环境是non-stationary的
    - 那把**每个人的critic**的输入加上其他人的动作，每个人看到的env就稳定了
- （训练时）不能用experience replay，因为

$$P(s'|s,a,\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_N) \neq P(s'|s,a,\boldsymbol{\pi}'_1,...,\boldsymbol{\pi}'_N) \text{ when any } \boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i.$$

- 那加上其他所有人的动作后，这两个就相等了

A primary motivation behind MADDPG is that, if we know the actions taken by all agents, the environment is stationary even as the policies change, since $P(s'|s,a_1,...,a_N,\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_N) = P(s'|s,a_1,...,a_N) = P(s'|s,a_1,...,a_N,\boldsymbol{\pi}'_1,...,\boldsymbol{\pi}'_N)$ for any $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i$. This is not the case if we do not explicitly condition on the actions of other agents, as done for most traditional RL methods.

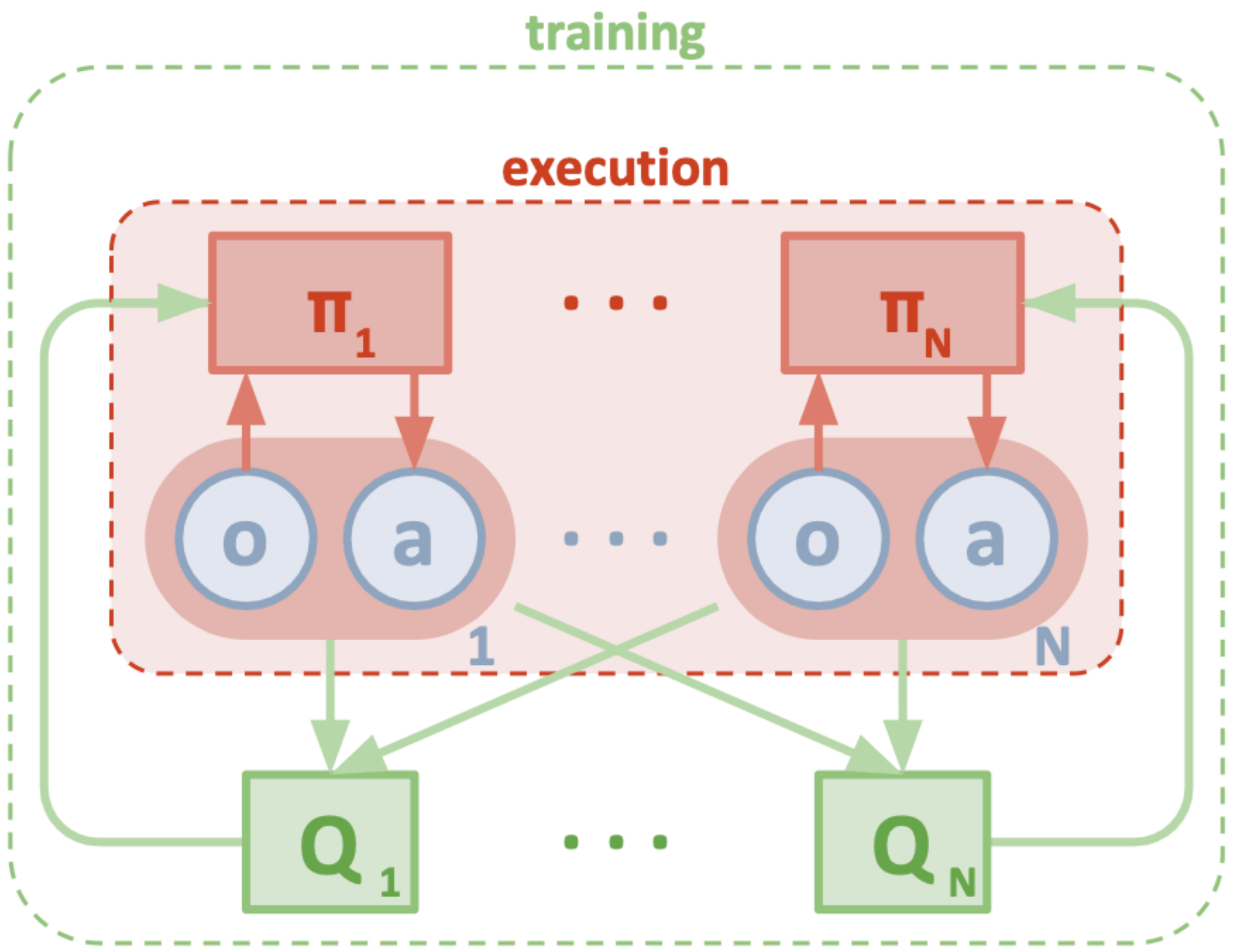
MARL

针对问题他们提出了MADDPG

- （训练时）每个人各自更新actor时梯度项的Q只有考虑自己的动作，估计梯度的方差会很大
    - 这个Q要是也考虑了别人的动作，方差就下来了（也是更准确了）

MADDPG

针对问题他们提出了MADDPG



- 所以他们的方法就在independent方法上改的
- 每个agent的SARL是用的DDPG，deep deterministic policy gradient
    - 每人一个actor，每人一个critic，不共享参数

- 额外的机制是，每个人的critic要求输入**所有人的观测和动作**
    - 也就是说更新的时候他们的critic也要用到**所有人的观测和动作**
    - 他们把这叫centralized critic

MADDPG

针对问题他们提出了MADDPG

- MADDPG中actor的更新：policy gradient
    - Q是centralized，要所有人的观测值和所有人的动作
    - 策略项还是只用自己的，毕竟对别人的策略求梯度也是0（加log是相加）
    - $\mathbf{x} = (o_1, ..., o_N)$，也可以再加点别的state里的东西

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\boldsymbol{\mu}}, a_i\sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)].$$

MADDPG

针对问题他们提出了MADDPG

- MADDPG中actor的更新：policy gradient
  - Q是centralized，要所有人的观测值和所有人的动作
  - 策略项还是只用自己的，毕竟对别人的策略求梯度也是0（加log是相加）
  - deterministic policy, continuous policy

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^{\boldsymbol{\mu}}, a_i \sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)].$$

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i = \boldsymbol{\mu}_i(o_i)}]$$

The objective function to optimize for is listed as follows:

$$J(\theta) = \int_{\mathcal{S}} \rho^{\mu}(s) Q(s, \mu_{\theta}(s)) ds$$

Deterministic policy gradient theorem: Now it is the time to compute the gradient! According to the chain rule, we first take the gradient of Q w.r.t. the action a and then take the gradient of the deterministic policy function μ w.r.t. θ:

$$\begin{aligned}
\nabla_{\theta} J(\theta) &= \int_{\mathcal{S}} \rho^{\mu}(s) \nabla_a Q^{\mu}(s, a) \nabla_{\theta} \mu_{\theta}(s) \rvert_{a=\mu_{\theta}(s)} ds \\
&= \mathbb{E}_{s \sim \rho^{\mu}} [\nabla_a Q^{\mu}(s, a) \nabla_{\theta} \mu_{\theta}(s) \rvert_{a=\mu_{\theta}(s)}]
\end{aligned}$$

MADDPG

针对问题他们提出了MADDPG

- MADDPG中actor的更新：policy gradient
  - Q是centralized，要所有人的观测值和所有人的动作
  - 策略项还是只用自己的，毕竟对别人的策略求梯度也是0（加log是相加）
  - deterministic policy, continuous policy

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\boldsymbol{\mu}}, a_i \sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)].$$

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}]$$

MADDPG

针对问题他们提出了MADDPG

- MADDPG中critic的更新
    - 要知道别人的policy，要访问别人的target actor

Here the experience replay buffer $\mathcal{D}$ contains the tuples $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, r_1, \ldots, r_N)$, recording experiences of all agents. The centralized action-value function $Q_i^{\boldsymbol{\mu}}$ is updated as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j' = \boldsymbol{\mu}_j'(o_j)}, \tag{6}$$

MADDPG

针对问题他们提出了MADDPG

- MADDPG中critic的更新
    - 要知道别人的policy，要访问别人的target actor
        - 可以每个人多加个神经网络来估计别人的policy，这样就不用访问了

Here the experience replay buffer $\mathcal{D}$ contains the tuples $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, r_1, \ldots, r_N)$, recording experiences of all agents. The centralized action-value function $Q_i^{\boldsymbol{\mu}}$ is updated as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j' = \boldsymbol{\mu}_j'(o_j)}, \tag{6}$$

MADDPG

针对问题他们提出了MADDPG

4.2 Inferring Policies of Other Agents

To remove the assumption of knowing other agents' policies, as required in Eq. 6, each agent i can additionally maintain an approximation $\hat{\boldsymbol{\mu}}_{\phi_i^j}$ (where ϕ are the parameters of the approximation; henceforth $\hat{\boldsymbol{\mu}}_i^j$) to the true policy of agent j, $\boldsymbol{\mu}_j$. This approximate policy is learned by maximizing the log probability of agent j's actions, with an entropy regularizer:

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j,a_j}\left[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)\right], \tag{7}$$

where H is the entropy of the policy distribution. With the approximate policies, y in Eq. 6 can be replaced by an approximate value $\hat{y}$ calculated as follows:

$$\hat{y} = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', \hat{\boldsymbol{\mu}}_i'^1(o_1), \ldots, \boldsymbol{\mu}_i'(o_i), \ldots, \hat{\boldsymbol{\mu}}_i'^N(o_N)), \tag{8}$$

MADDPG

针对问题他们提出了MADDPG

- 这个估计别人policy的神经网络，目标是模拟别人的actor
  - 输入别人的obs，用别人的actor会才养出一个action
  - 现在这个神经网络输入别人的obs，要最大化输出别人输出的action的概率
  - 加熵是加探索
- 他们说这是可以online的：更新critic前，从buffer里拿出最新的sample来更新

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j, a_j}\left[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)\right]$$

MADDPG

他们还提出了policy ensembles

- 他们说环境的non-stationary在competitive setting下很明显
    - competitive setting下，每个人的policy都是overfitting对手的行为的
    - 对手变了个策略，这样的策略就很容易失败了
- 他们提出要训练个有K个子策略(sub-policy)的集合
    - 每个episode，每个agent随机选某个sub-policy来执行（uniform）
    - $\boldsymbol{\mu}_i$是agent i的子策略集合
    - $\boldsymbol{\mu}_{\theta_i^{(k)}}$是agent i的第k个子策略，也记为$\boldsymbol{\mu}_i^{(k)}$

MADDPG

他们还提出了policy ensembles

- 每个人的ensemble的优化目标：

$$J_e(\boldsymbol{\mu}_i) = \mathbb{E}_{k\sim\mathrm{unif}(1,K), s\sim p^{\boldsymbol{\mu}}, a\sim\boldsymbol{\mu}_i^{(k)}}\left[R_i(s,a)\right]$$

- 每人有k个buffer，其中每个buffer都存一个sub-policy跑出来的东西
    - 因为每个episode的sub-policy是随机选出来的

$$\nabla_{\theta_i^{(k)}} J_e(\boldsymbol{\mu}_i) = \frac{1}{K}\mathbb{E}_{\mathbf{x},a\sim\mathcal{D}_i^{(k)}}\left[\nabla_{\theta_i^{(k)}}\boldsymbol{\mu}_i^{(k)}(a_i|o_i)\nabla_{a_i}Q^{\boldsymbol{\mu}_i}(\mathbf{x},a_1,\ldots,a_N)\Big|_{a_i=\boldsymbol{\mu}_i^{(k)}(o_i)}\right]$$