A3C算法

- 先来看看AC

  - 还记得我们的老朋友吧：$\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right) \right)$
  - 后续获得的所有奖励：$\sum_{t'=t}^{T} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i)$ 可能并不稳定
  - 引入过baseline：$\left(\sum_{t'=1}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) - b \right)$,并用网络$V^\pi(\mathbf{s}_t)$ 来估计b值
  - 再来回忆下Q：$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta} \left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t \right]$

A3C算法

- 优势函数（Adavantage）

  - 函数表达式：$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t)$

  - 就是在状态s下，选择某一动作有多好，Q相当于咱们得到的；V是期望的（平均）

  - 就好比你现在考试，老师（V）认为你能考100分，其实只考了5分（Q）

  - 如果A值计算是正的，那就说明当前动作执行的挺好，要继续朝这方向干

A3C算法

- AC需要解决的问题

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

- 把这几个当事人都叫来吧: $V^{\pi}(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)]$

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t)$$

- 由于智能体在与环境交互过程中有大量的随机性，所以算的是期望

- 为了计算A，现在出现了Q和V，那我得训练俩网络了
（一个都很难整，现在给我过来俩？）

- 来个近似让问题简单些吧：（现在只需要训练一个网络就够了！）

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^{\pi}(\mathbf{s}_{t+1}) - V^{\pi}(\mathbf{s}_t)$$

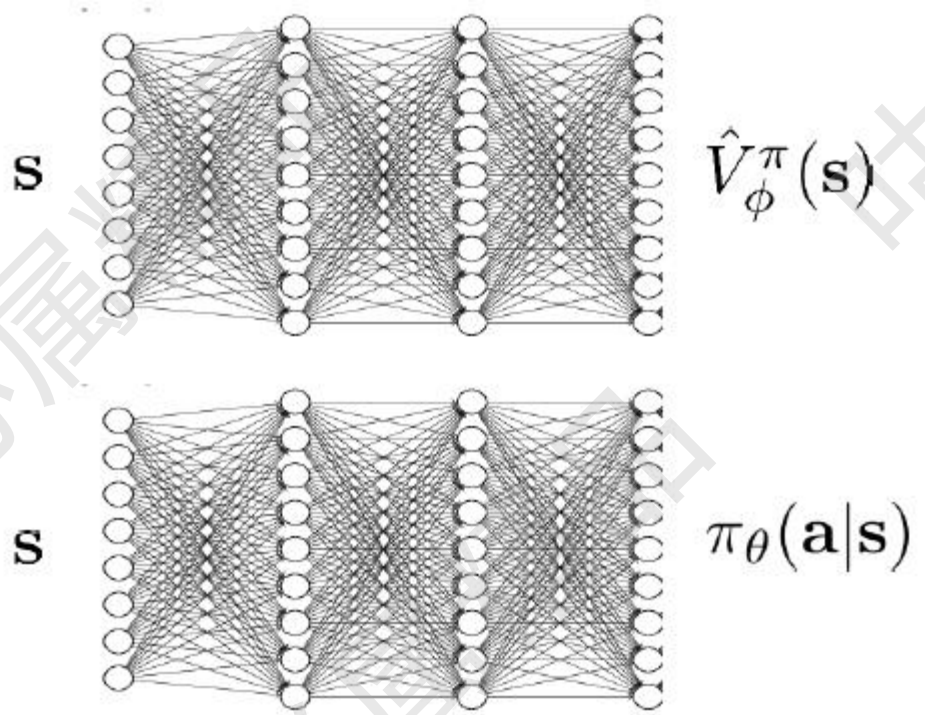
A3C算法

- AC整体流程：

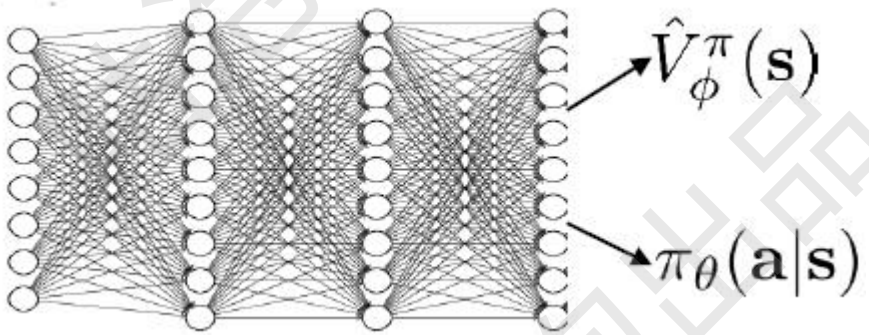
  - 1.获取数据：$\{\mathbf{s}_i, \mathbf{a}_i\}$（不断与环境交互，通过策略 $\pi_\theta(\mathbf{a}|\mathbf{s})$ ）
  - 2.前向传播计算：$A^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + V_\phi^\pi(\mathbf{s}_i') - V_\phi^\pi(\mathbf{s}_i)$
  - 3.计算梯度：$\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) A^\pi(\mathbf{s}_i, \mathbf{a}_i)$
  - 4.更新参数：$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

A3C算法

- AC算法细节



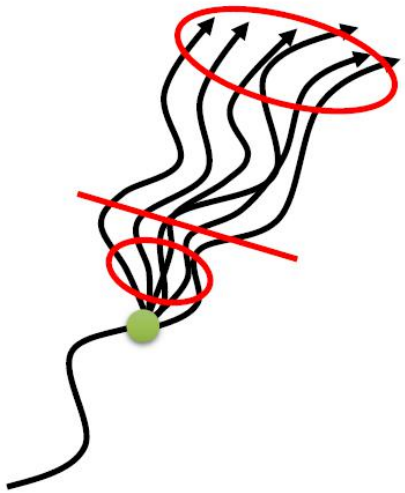
- 是不是得训练两个网络呢?
- 分别完成其对应的任务
- 但是它俩好像都是根据状态来预测结果
- 共享一下吧： $\mathbf{s}$



A3C算法

- AC算法细节

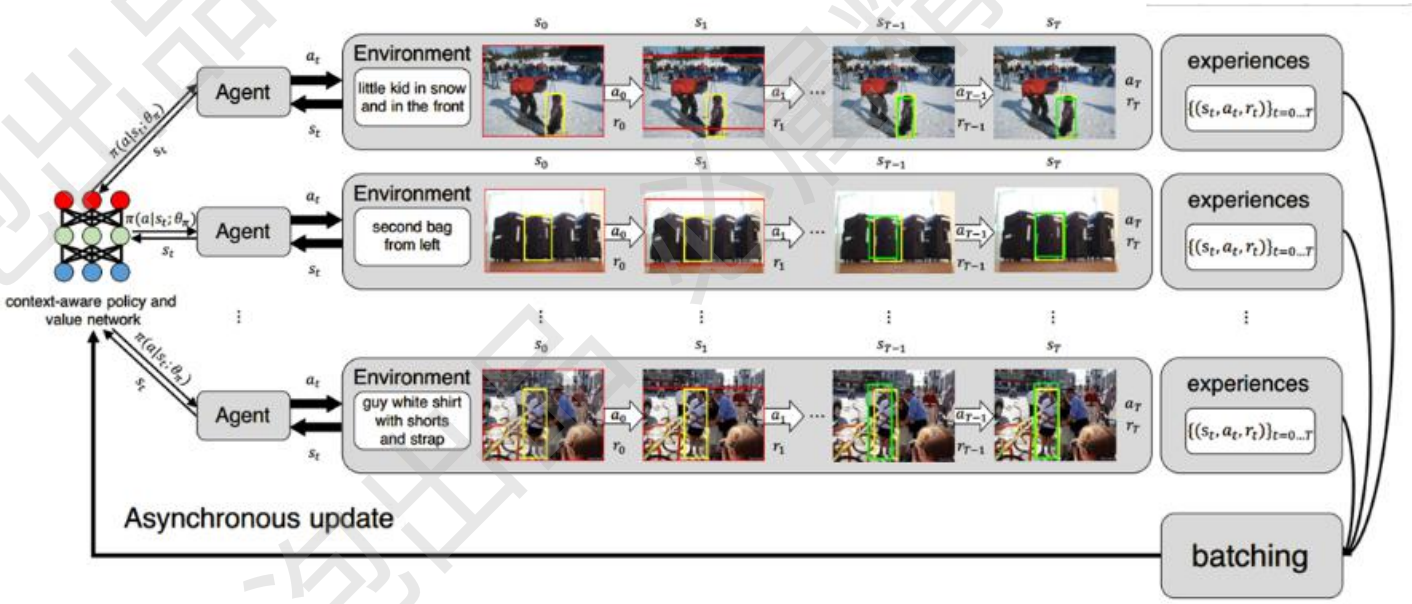
  - n-step: 只算一步会不会有点简单呢？让它眼光更长一些
  - 实际计算公式：$\hat{A}_n^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_{\phi}^{\pi}(\mathbf{s}_t) + \gamma^n \hat{V}_{\phi}^{\pi}(\mathbf{s}_{t+n})$
  - 还引入了折扣系数，越往后的情况影响力稍微有所降低
  - n值通常情况下也不同太大，越大的话variance也会越大



A3C算法

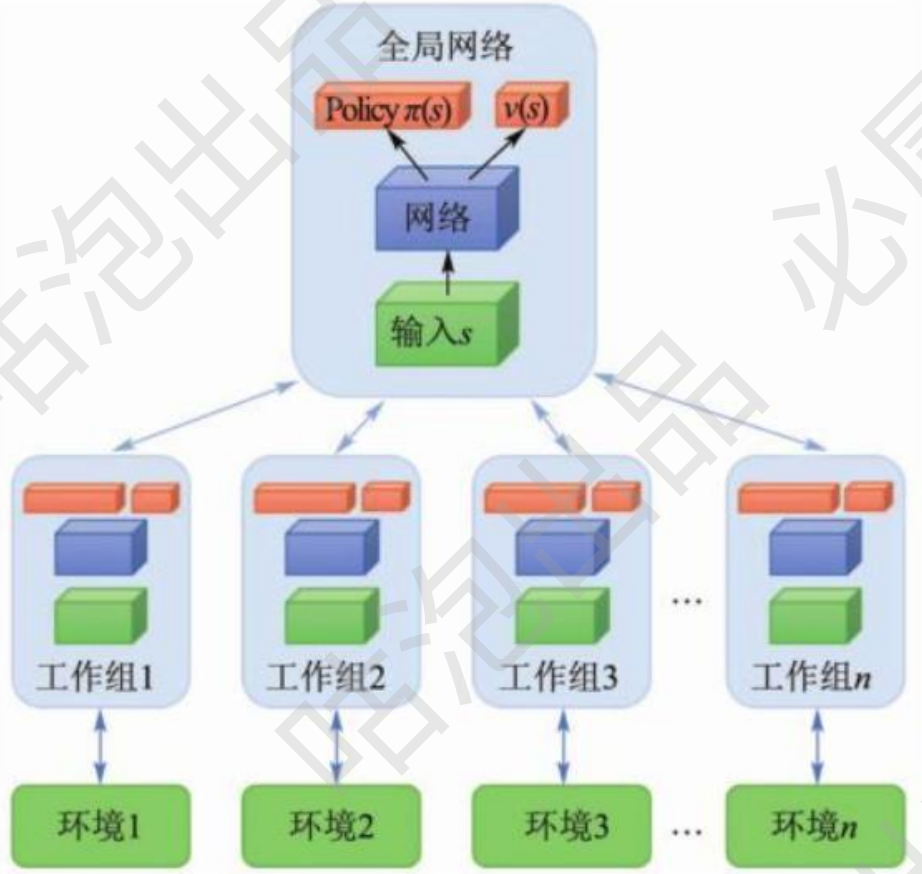
- A3C整体架构

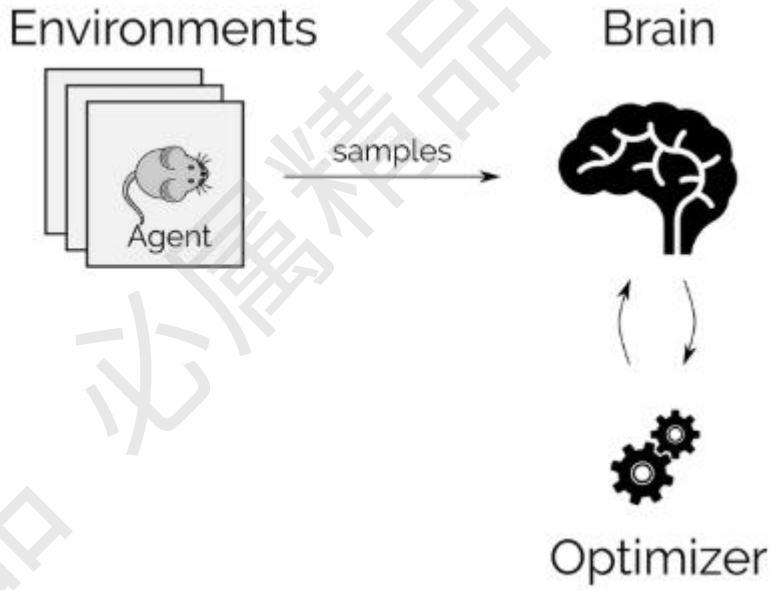
- 如果只用一个智能体不断与环境交互得到数据， 会有什么问题吗?
- 样本之间的相关性会较大，违背了机器学习的本质（独立同分布）
- 多个智能体（多线程）
- 每一个都自己去玩（单独）



A3C算法

- A3C整体架构





A3C算法

- 损失函数：

  - 策略损失（Policy）：$-J(\pi_{\theta'}) = -\mathop{\mathbb{E}}_{\tau\sim\pi_{\theta'}}\sum_{t=0}^{T} log\pi_{\theta'}(a_t|s_t)A$ （起决策的网络）
  - Value网络损失：$L(\theta_v) = \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta}}\left[\frac{1}{2}\sum_{t=0}^{T}(R(\tau) - V(s_t;\theta_v))^2\right]$ （预期与实际的差异）
  - 熵：$-H(\pi_{\theta'}(a_t|s_t))$ （熵越大表示各种行为可能性都能有一些，别太绝对）
  - 整体损失函数：$Total_{loss} = Policy_{loss} + \alpha * Value_{loss} + \beta * Entropy_{loss}$