DQN算法

- 挑战一张图解释Q-learning

- 干一件事包括：眼前的瞬时奖励+记忆经验奖励
- 瞬时奖励：做了一个动作就能获得的奖励
- 记忆经验奖励：按照训练时的经验，上一个动作发生后，接下来怎么做才能获得更大的奖励（补刀。。。）
- DQN就是用神经网络来预测了，先来看看Q-learning是咋玩的

哈泡出品 必属精品

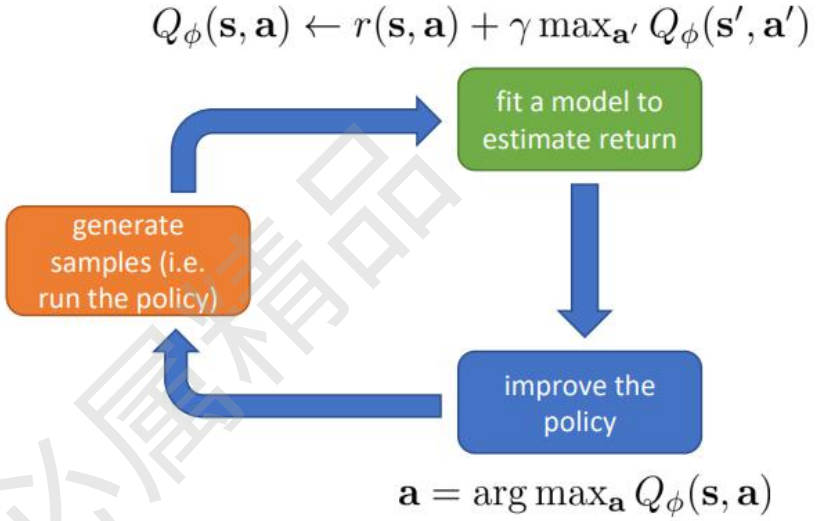
DQN算法

- Q-learning要做什么？ （奖励的期望）

- 收集数据: $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ (就是玩游戏记录)

- 令目标等于： $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$

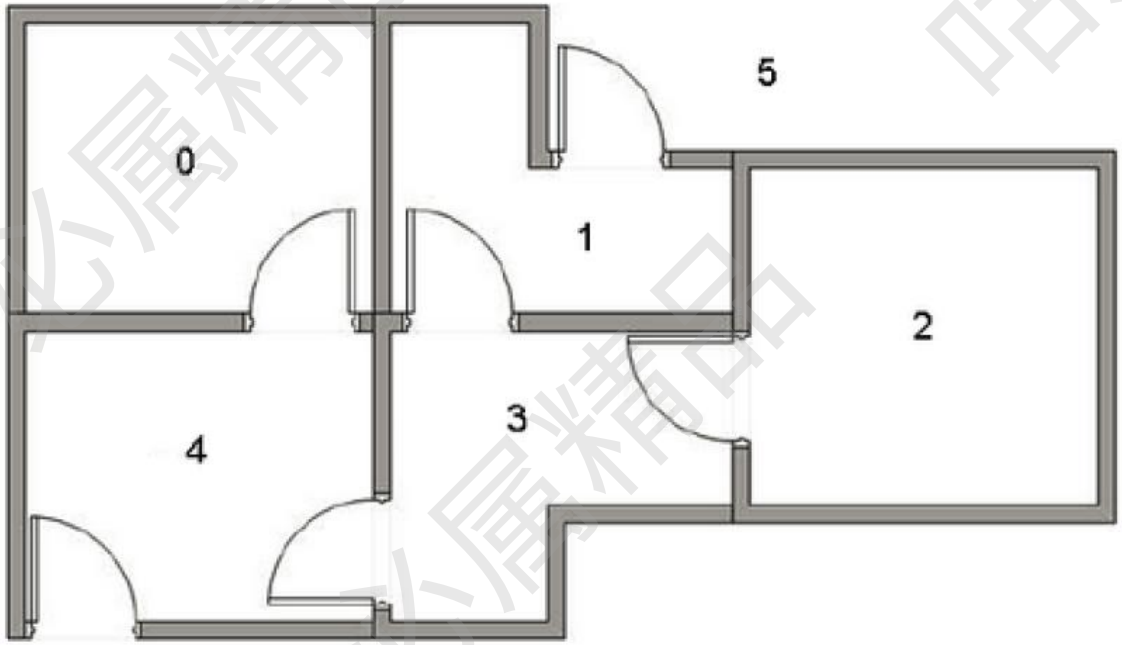
- 目标函数： $\arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$



- 如何知道下一步做什么才能使得奖励更大呢？查表来迭代更新！

DQN算法

- 准备密室逃脱



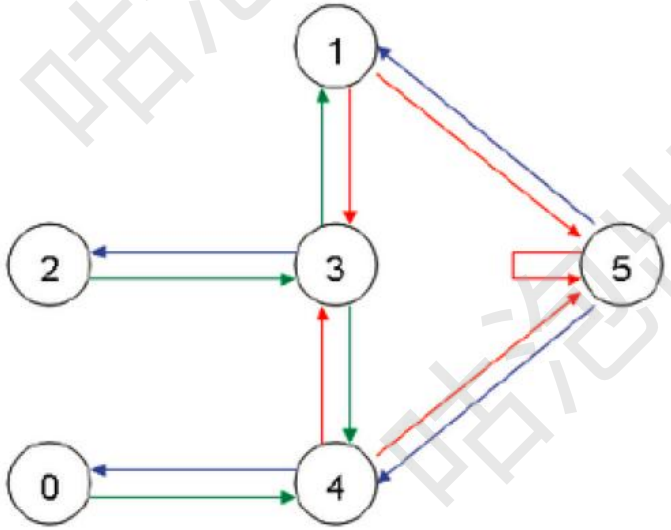
- 状态：0,1,2,3,4,5（其中5是出口）
- 目的就是能逃出去
- 不是每个状态都互通的（2只能到3）
- 迭代让机器人能逃出去

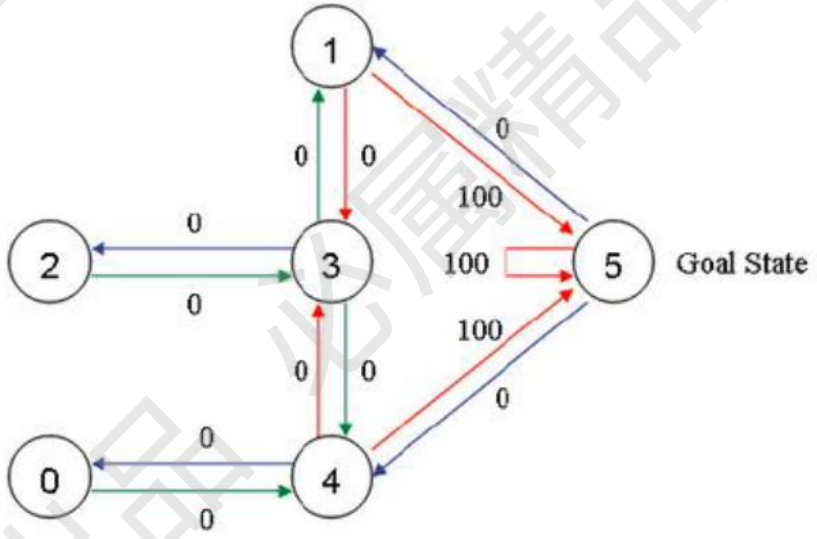
必属精品 咕泡出品

DQN算法

- 密室地图

- 由于5是出口，所以能到5就会获得比较大的奖励，其余均为0





DQN算法

- 初始化Q和R

- Q现在看起来是一个空表，要不断进行填充（行为state，列为action）

$$Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$R=\begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

DQN算法

- 开始迭代

- 假设：初始化状态为1

- 根据右表，action只能选择3和5

$$R=\begin{array}{c|cccccc} \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

Action

- 此时选择3还是5呢？当前的Q为空表，先随机选一个，例如5

- 接下来就到了状态5，此时有三种选择（1,4,5），选哪个呢？

必属精品 咕泡出品

DQN算法

- 开始迭代

  - 按照之前的约定：

$$Q(state, action) = R(state, action) + Gamma * Max[Q(next\ state, all\ actions)]$$

$$Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

  - 其中0.8是折扣因子，相当于经验记忆奖励的权重

  - Q(1,5)这不就来了嘛，5也代表了游戏结束！

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

  - 重新玩，比如在开始的时候再随机选择一个状态

必属精品 咕泡出品

DQN算法

- 开始迭代

  - 再来一次，这回初始化选择到了状态3

  - 此时可以选择action(1,2,4),随机选到了1

  - 下一步在1那可以选择去3或者5：

    Q(3,1)=R(3,1)+0.8max(Q(1,3),Q(1,5))
    = 0+0.8*max(0,100)
    =80

$$R=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} \text{Action} & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

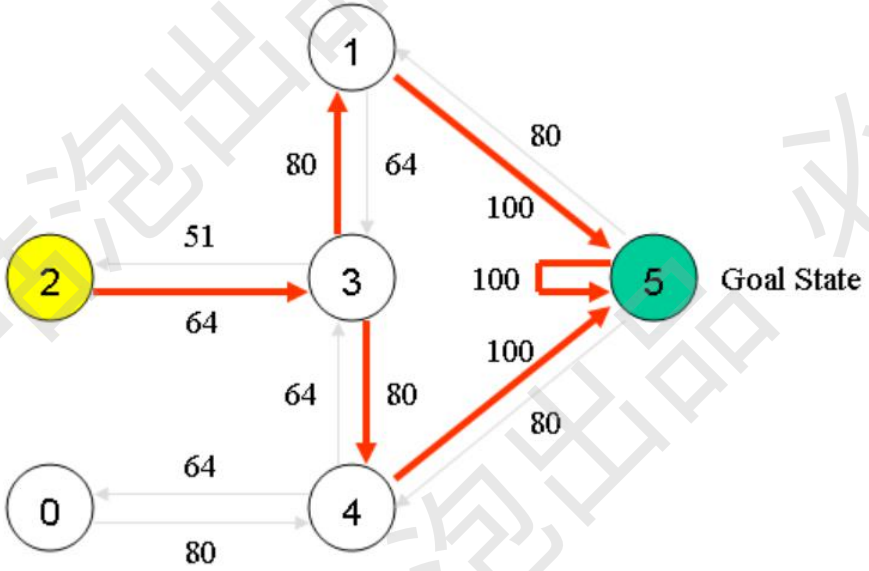
DQN算法

- 开始迭代

- 玩了很多次之后，得到收敛后的模型结果：$Q=$

	0	1	2	3	4	5
0	0	0	0	0	80	0
1	0	0	0	64	0	100
2	0	0	0	64	0	0
3	0	80	51	0	80	0
4	64	0	0	64	0	100
5	0	80	0	0	80	100

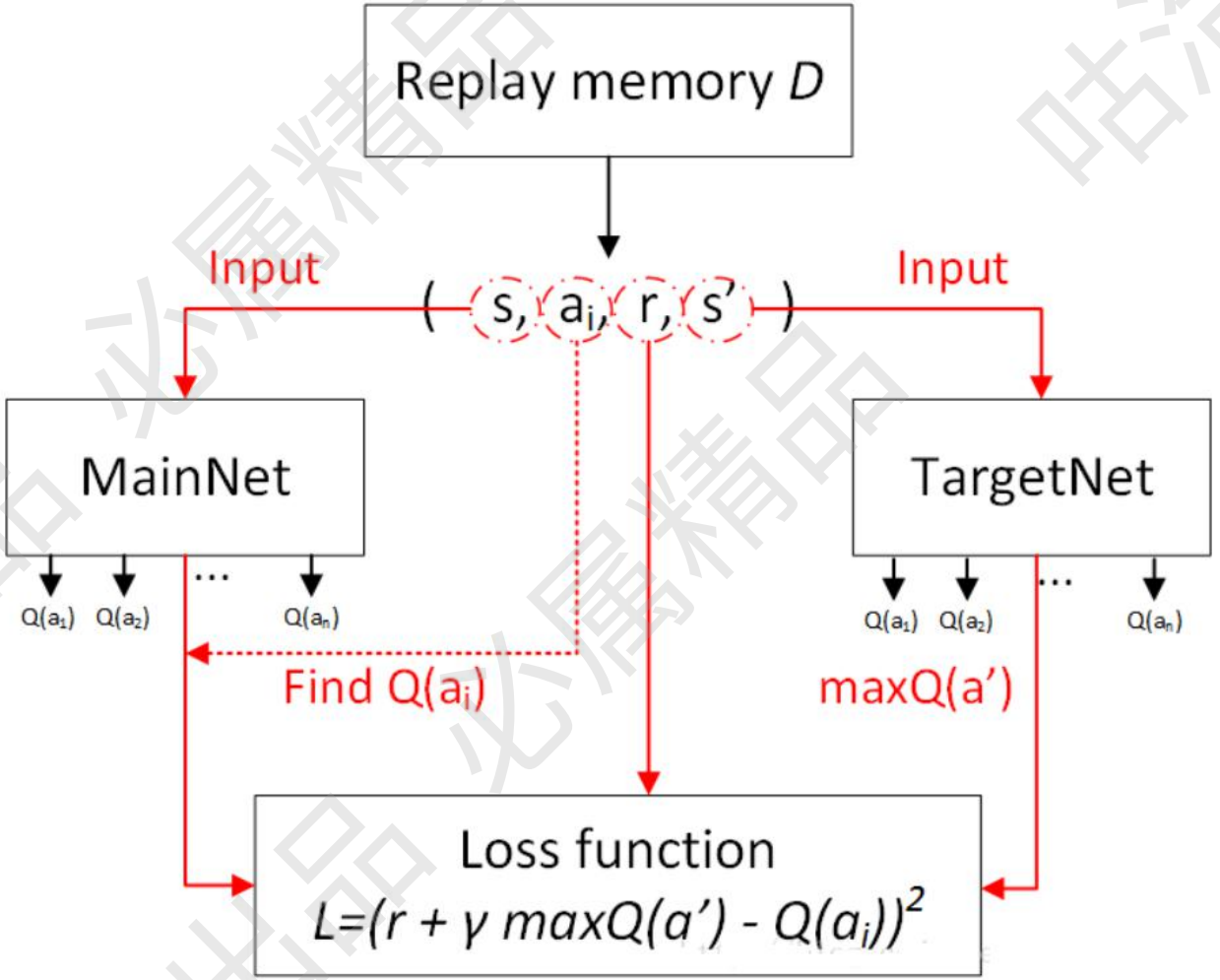
- 有了这个表，密室逃脱就很容易了：



DQN算法

- 什么是DQN呢?

- 1.先玩一阵，得到记录数据
- 2.从记录中取一个batch
- 3.target与main相同网络结构
- 4.损失函数就是回归问题



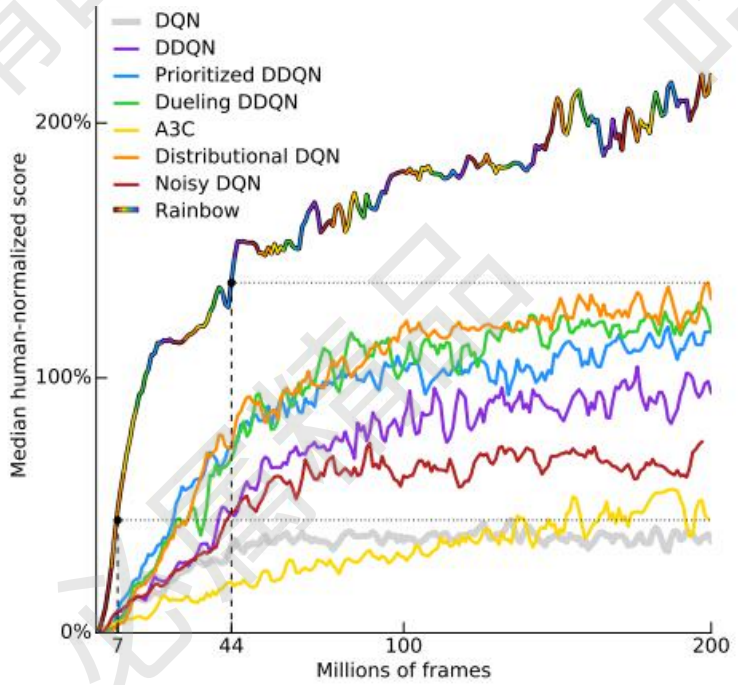
DQN算法

- 什么是DQN呢?
  - 又到了算账时间，状态真的可以穷举吗？如果是游戏画面，那像素点太多了！
  - Q(s,a)这回不能用表格了，得用神经网络来做了！
  - 数据如何获取呢？构建一个replay buffers，用的时候去里面取一个batch就行
  - 其实就是off policy策略，代码构建还是比较容易的

必属精品 咕泡出品

DQN算法

- 再整个彩虹
  - 其实就是DQN基础上还能再改进很多
  - 咱们挑几个重点的来唠唠
  - Rainbow其实就是把各种套路全都整上了
  - （Double-DQN，Dueling-DQN，MultiStep）



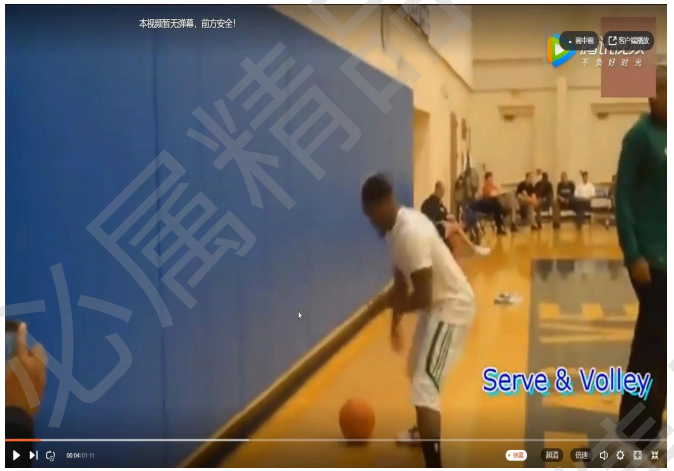
必属精品 咕泡出品

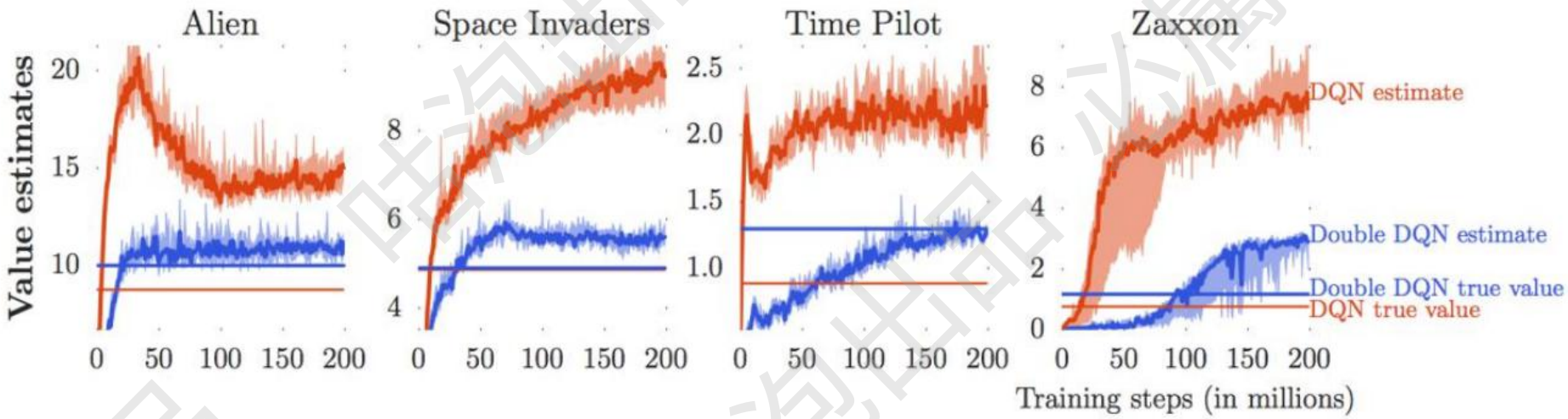
DQN算法

- Double-DQN

  - 挑战一张图解释Double-DQN：

  - 咱们的网络会高估自己！



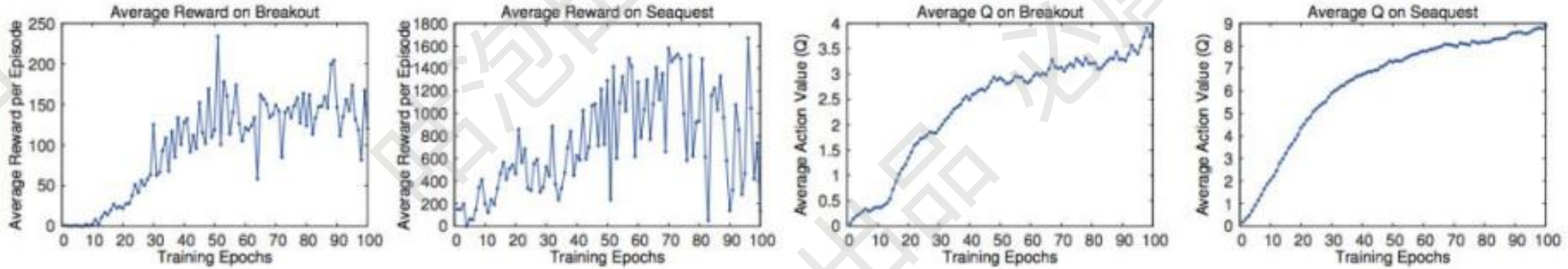


DQN算法

- Double-DQN

  - 游戏玩着玩着咋还飘上了呢！随着游戏的进行，期望的Q会越大越大！

  - 这并不利于网络训练，学习得脚踏实力的！



DQN算法

- Double-DQN

- targetValue：$y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$

这兄弟会惹麻烦

- 如何解决问题呢？来个双重保险吧！
  （其中A还是DQN，额外引入B让他时刻提醒A低调点）

$$Q_{\phi_A}(\mathbf{s}, \mathbf{a}) \leftarrow r + \gamma Q_{\phi_B}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_{\phi_A}(\mathbf{s}', \mathbf{a}'))$$

必属精品 咕泡出品

DQN算法

- Dueling-DQN

- 挑战一张图解释Dueling-DQN：
- 童年回忆来了，去A网吧被你老爸一顿揍之后是不是去B网吧也相同的结果呢？
- 能不能让网络有点举一反三的能力呢？而不是一次就更新当前的结果
- 这件事怎么做呢？做起来比解释简单多了，一个偏置项就够了！

必属精品 咕泡出品

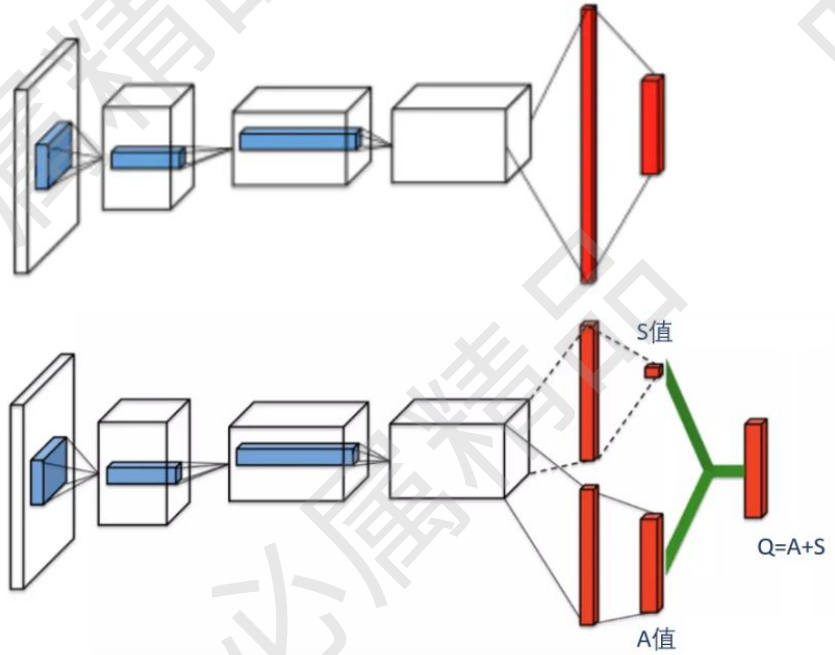
DQN算法

- Dueling-DQN

  - 网络架构中多了一块，输出俩货：

  - 多了一个V值（可以当做偏置项）

  - Q(s,a)=A(s,a)+V(s)，举一反三！



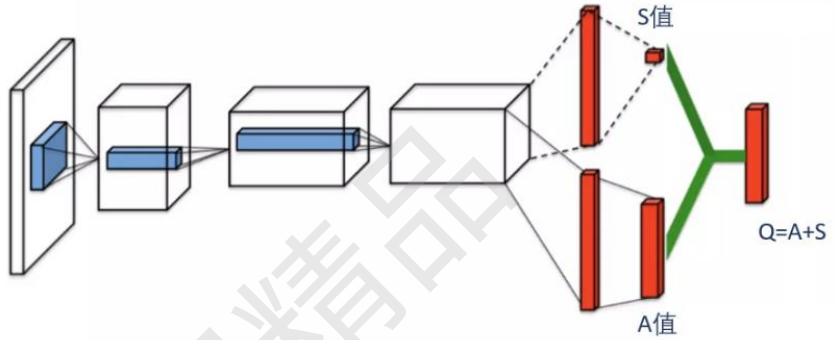
  - 其实就是让网络能同时更新相同state下不同action可能导致的结果
    （一般迭代的时候就更新当前状态下，某一个action的结果）
    （当你老爸很生气的时候，你去A,B,C,D...哪个网吧的结果都差不多）

必属精品 咕泡出品

DQN算法

Dueling-DQN

- 如何让网络知道调整V(s)而不是A(s,a)呢?
- 让A中所有列的值加起来恒为0(限制条件)



- A(s,a)，此时如果只更改一个值，满足不了条件，那只能调V(s)了。

2	4	1	2
3	-1	-1	-4
-5	-3	0	2

必属精品 哈泡出品

DQN算法

- MultiStep-DQN

  - 挑战一句话解释MultiStep，曾经看到过这样一句话：
  - 为什么宁肯去打工也不愿意学习？生活的苦是被动的，学习的苦是主动的
  - 能不能将眼界放远一些呢？不光要看眼前（下一步），也要看（下N步）结果
  - 其实MultiStep就是计算Q值的时候选择多个时间步

咕泡出品 必属精品

DQN算法

- MultiStep-DQN

  - 只看下一步的情况：$y_{j,t} = r_{j,t} + \gamma \max_{\mathbf{a}_{j,t+1}} Q_{\phi'}(\mathbf{s}_{j,t+1}, \mathbf{a}_{j,t+1})$

  - 下N步的情况：$y_{j,t} = \sum_{t'=t}^{t'+N-1} r_{j,t'} + \gamma^N \max_{\mathbf{a}_{j,t+N}} Q_{\phi'}(\mathbf{s}_{j,t+N}, \mathbf{a}_{j,t+N})$

  - 其实就像梯度下降中的，随机/批量，这个就是小批量

必属精品 哈泡出品

DQN算法

- continuous actions

  - 离散值很好搞定（上，下，左，右）连续值怎么办？（力度，角度等）
  - 如何来求解呢：$y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$
  - 看起来就是一个求极值问题，常用方法可以采样，梯度上升
  - 采样：$\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \approx \max\{Q(\mathbf{s}, \mathbf{a}_1), \ldots, Q(\mathbf{s}, \mathbf{a}_N)\}$

    （遍历尽可能多的情况，得到一个最大值就得了）

咕泡出品 必属精品

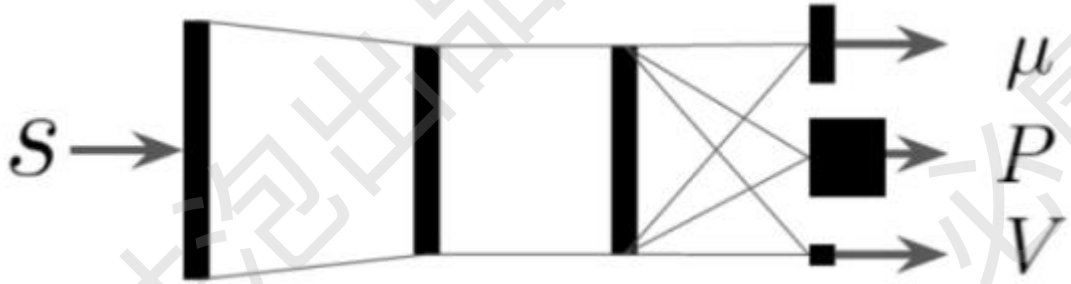
DQN算法

- continuous actions

- 优化求解：$y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$

- 就是在内层再训练一个网络来解这个梯度上升问题，够麻烦。。。

- 更强的一招：



- 重新定义Q网络，输出三个结果，分别是向量，矩阵，值

必属精品 咕泡出品

DQN算法

- continuous actions

  - 新的Q：$Q_\phi(\mathbf{s},\mathbf{a}) = -\frac{1}{2}(\mathbf{a}-\mu_\phi(\mathbf{s}))^T P_\phi(\mathbf{s})(\mathbf{a}-\mu_\phi(\mathbf{s})) + V_\phi(\mathbf{s})$

  - 由这三兄弟组成了，为啥要凑成这个样子呢？怎么找到合适的action呢？

  - 这兄弟 $(\mathbf{a}-\mu_\phi(\mathbf{s}))^T P_\phi(\mathbf{s})(\mathbf{a}-\mu_\phi(\mathbf{s}))$ 恒为正，所以action= $\mu_\phi(\mathbf{s})$

  - 此时代入Q(s,a)即可求出：$\max_{\mathbf{a}} Q_\phi(\mathbf{s},\mathbf{a}) = V_\phi(\mathbf{s})$

咕泡出品 必属精品