

# **Table of contents**

1. [Introduction 2](#_TOC_250009)
2. [Markov Decision Processes 2](#_TOC_250008)
3. Future Cumulative Reward 3
4. Q-Learning 4
   1. [The Q-value 4](#_TOC_250007)
   2. [The Temporal Difference 4](#_TOC_250006)
   3. [The whole Q-Learning process 5](#_TOC_250005)
5. Deep Q-Learning 6
   1. [Q-Learning into Deep Learning 6](#_TOC_250004)
   2. [Experience Replay 6](#_TOC_250003)
   3. [The whole Deep Q-Learning Process 6](#_TOC_250002)
6. Deep Convolutional Q-Learning 8
7. Asynchronous Actor-Critic Agents (A3C) 9
   1. [A3C Intuition 9](#_TOC_250001)
   2. [The whole A3C Process 9](#_TOC_250000)

Conclusion

# **简介**

该手册是人工智能A-Z在线课程的另一资源，其中包含用于实施模型的所有理论。对于希望将所有课程数学都收集在一本结构化手册中的人，以及想深入了解AI方程背后的数学细节的人，这将非常有用。它假定了神经网络的先验知识。

# **马尔可夫决策过程**

要开始AI之旅，我们必须从马尔可夫决策过程开始。马尔可夫决策过程是一个元组 (*S, A, T, R*)，其中：

•S是不同状态的集合。状态可以是编码值的向量或3D输入图像。让我们

以突围游戏为例。在那种情况下会是什么状态？如果状态具有编码值向量的形式，则该向量将包含所有可以描述在时间t处发生的情况的信息，即：球位置的坐标，方向向量的坐标球的运动方向，并且如果每个砖块仍然存在，则它们的一些二进制值可能为1，否则为0。向量中的所有这些值可以编码环境的状态。然后，此向量将成为神经网络的输入，其中输出将成为要播放的动作。并且如果状态是3D输入图像，则输入将恰好是每个时间t的游戏屏幕截图。我们将在“深度卷积Q学习”一节中研究这种情况。

•A是每次t可以播放的不同动作的集合。对于“突破”，有六种可能的操作：向左移三个强度级别，向右移三个强度级别。对于《毁灭战士》，还有更多动作：向前，向后，向左，向右，奔跑，射击...

•T称为过渡规则：

*T* : (*at* ∈ *A, st* ∈ *S, st*+1 ∈ *S*) ›→ P(*st*+1|*st, at*)

其中P（*st*+1 | *st, at*）是在当前状态为 *st* 且所执行的动作为 *at* 的情况下将来状态为 *st*+1 的概率。因此，T是在给定当前状态和在t时刻采取行动的情况下 t +1 时刻未来状态的概率分布。因此，我们可以通过从分布T中抽取随机数来预测未来状态 *st*+1：

*st*+1 ∼ *T* (*at, st, .*)

•R是​​奖励函数：

*R* : (*at* ∈ *A, st* ∈ *S*) ›→ *rt* ∈ R

其中*rt* 是在状态 *st* 中进行动作后获得的奖励。

定义MDP之后，重要的是要提醒它，它依赖于以下假设：

未来状态 *st*+1 的概率仅取决于当前状态 *st* 和当前的动作 *at*，而不取决于先前的任何状态和动作。 那是：

P(*st*+1|*s*0*, a*0*, s*1*, a*1*, ..., st, at*) = P(*st*+1|*st, at*)

现在，让我们回顾一下MDP方面的情况。在每一次t：

1. AI观察当前状态 *st*

2. AI发挥作用 *at*

3. AI收到奖励 *rt* = *R*(*at, st*)

4. AI进入以下状态 *st*+1 现在的问题是：

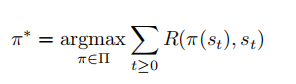
### AI如何知道在每个时间 t 执行哪个动作？

为了回答这个问题，我们需要引入策略功能。策略函数 π 恰好是给定状态 *st*

的函数，它在以下位置返回操作：

*π* : *st* ∈ *S* → *at* ∈ *A*

让我们用Π表示所有策略功能的集合。这样，选择最佳动作就成为优化问题。实际上，归结为找到使累积奖励最大化的最佳策略π∗：



所以现在问题当然变成了：

### 如何找到最佳策略 *π*∗ ?

这就是Q-Learning的作用。但是在开始之前，我们需要了解未来奖励的概念。

# **未来累积奖赏**

让我们考虑游戏的一个情节，即从t = 0（游戏开始）到t = n（游戏结束）。在玩游戏时，大多数时候目标是达到最高分数。因此，为此，我们不应该仅在时间t处考虑奖励，而应该在游戏结束之前（从时间t到时间n）考虑将来的累积奖励。因此，我们需要考虑的不是 *rt* = *R*(*at, st*)，而是：

*Rt* = *R*(*at, st*) + *R*(*at*+1*, st*+1) + *...* + *R*(*an, sn*) = *rt* + *rt*+1 + *...* + *rn*

但是，我们仍然可以改进模型。元素 *rt* ，*rt*+1，...和 *rn* 是我们尝试使用奖励函数R估算的值。因此，当我们在时间t时，对未来的研究越深入，我们对奖励的把握就越不确定在这个未来的时间。换句话说， t' 越大，估计奖励 *rt+*t'的方差越大。因此，要修复这种影响，我们需要对未来的每个奖励进行折现，而我们在未来越远，我们对其进行的折现就越多，因为我们对它的确定性越低。实现此目的的一种方法是获得一些折扣：

Selection_009

其中 *γ* ∈ [0*,* 1]。这样，*t*j 越高， *γtt* 越小，因此 *rt+*t'的折扣就越大。

γ称为折扣因子。 γ越接近0，则AI越会尝试优化当前回报 *rt*。 γ越接近1，则AI旨在优化未来奖励的目标就越多。 这个折扣系数是我们将为 Q-Learning 算法保留的。

# **Q-Learning**

## The Q-value

对于每对动作和状态 (*a, s*)，我们将关联一个数值 Q(a,s)：

*Q* : (*at* ∈ *A, st* ∈ *S*) ›→ *Q*(*at, st*) ∈ R

我们将说 *Q*(*a, s*)是 "在状态s中进行的动作a的Q值"。要了解此目的，我们需要介绍时间差异。

## 时间差异

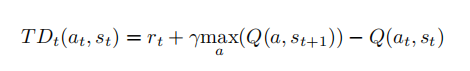
在开始t = 0时，所有Q值都初始化为0。

现在，假设我们处于时间t，处于特定状态 *st*。我们在进行操作，然后获得回报 *rt*。然后我们从 *T* (*at , st , .*)分布中随机抽取，将我们引向下一个状态 *st*+1：

*st*+1 ∼ *T* (*at , st , .*)

现在我们可以介绍时间差异，这是Q-Learning的核心。在时间t处的时间差，用*TDt*(*at, st*)表示，它们之间的差为：

* *rt* + *γ*max(*Q*(*a, st*+1)), *γ* ∈ [0*,* 1]，即通过在状态 *st* 处进行动作获得的报酬 *rt*，加上一个百分比（这是我们先前的折现因子 γ） ）在未来状态 *st*+1 中播放的最佳动作的Q值，
* 和 *Q*(*at, st*)，即处于状态 *st* 的动作的Q值，因此导致：



### 但是这个时间差TDt(***at , st*** ) 的目的是什么？

*TDt*(*at , st*) 就像一个内在的奖励。 AI将以以下方式学习Q值：

* 如果 *TDt*(*at , st*) 高，则AI会收到“很好的惊喜”。
* 如果 *TDt*(*at , st*) 很小，则AI会感到“沮丧”。

然后，在Q-Learning算法的最后下一步中，我们使用时间差来增强从时间t -1到时间t的一对 (action, state)，其公式如下：

Selection_011

从这个角度来看，Q值衡量与动作和状态 (*at, st*) 对相关的惊喜或沮丧的累积。 在惊喜情况下，AI被增强，而在沮丧情况下，AI被削弱。 因此，我们想学习将给AI最大的“惊喜”的Q值。

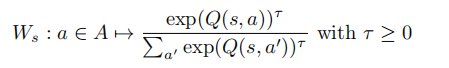
因此，决定要执行哪个动作主要取决于Q值 Q(*at , st*)。 如果动作在 *at* 在状态 *st* 下玩游戏与较高的Q值 Q(a , *st*) 相关联，则AI有较高的倾向选择 a。 另一方面，如果在状态s中进行的动作a与较小的Q值 Q(*at , st*) 相关联，则AI较小的倾向选择 *at*。

有几种选择最佳动作的方法。 首先，当处于特定状态 *st* 时，我们可以简单地选择一个具有最大 Q(a , *st*) 的a：

Selection_012

但是经验表明，这不是最佳选择。 更好的解决方案是softmax方法。

softmax方法包括为每个状态s考虑以下分布：



然后，通过从该分布中随机抽取来选择要播放的动作：

a ∼ Ws(.)

## 整个Q-Learning过程

让我们总结一下整个Q学习过程的不同步骤：

初始化：

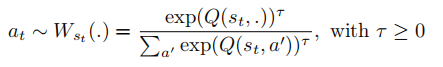
对于所有一对动作a和状态s，Q值均初始化为0：

∀*a* ∈ *A, s* ∈ *S, Q*0(*a, s*) = 0

我们从初始状态s0开始。 我们进行一个随机动作，并到达第一状态s1。

在每次t≥1时：

1.我们在处进行操作，其中处是来自Ws分布的随机抽奖：

**

2.我们得到的回报是 *rt* = *R*(*at, st*)

3.我们进入下一个状态 *st*+1*，*其中 *st*+1 是从 T (*at , st , .*)分布中随机抽取的：

*st*+1 ∼ *T* (*at, st, .*)

4.我们计算时间差

Selection_019

5.我们更新Q值：

*Selection_020*

# **深度Q学习**

## Q-深度学习

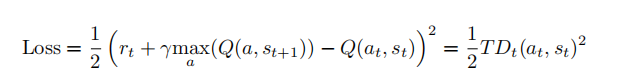
深度Q学习包括将Q学习与人工神经网络相结合。输入是编码的向量，每个向量定义环境的状态。这些输入将进入人工神经网络，其中输出是要播放的动作。更确切地说，假设游戏有n种可能的动作，神经网络的输出层由n种输出神经元组成，每个输出神经元对应于当前状态下执行的每个动作的Q值。然后，执行的动作是与输出神经元关联的具有最高Q值（argmax）的动作，或者是通过softmax方法返回的动作。而且由于Q值是实数，因此使我们的神经网络成为回归的ANN。

因此，在游戏的每个状态下：

•预测是Q值 Q(*at , st*)，其中a由argmax或softmax选择，

•目标是 Selection_021

•损失误差是时间差的平方：

**

然后，将该丢失错误反向传播到网络中，并根据权重对错误的贡献程度来更新权重。

## 体验重播

我们注意到，到目前为止，我们仅考虑了从一个状态 *st* 到下一个状态 *st*+1 的过渡。问题在于，大多数情况下， *st* 与 *st*+1 非常相关。因此，网络学习不多。如果我们考虑最后一个m个过渡（其中m是一个大数字），而不是仅考虑上一个过渡，则可以改进此方法。这最后m个转换包称为“体验重播”。然后，从此“体验重播”中，我们随机进行一些过渡，以进行更新。

## 整个深度Q学习过程

让我们总结一下整个深度Q学习过程的不同步骤：

### 初始化：

对于所有一对动作a和状态s，Q值均初始化为0：

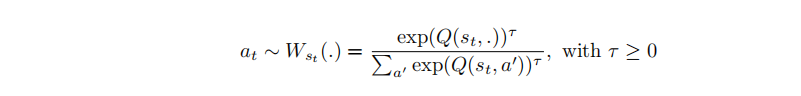
∀*a* ∈ *A, s* ∈ *S, Q*0(*a, s*) = 0

体验重播被初始化为空列表M

我们从初始状态s0开始。我们进行一个随机动作，并到达第一状态 s1

在每次t≥1时：

1.我们在处进行操作，其中处是来自Ws分布的随机抽奖：

**

2.我们得到的回报是 *rt* = *R*(*at, st*)

3.我们进入下一个状态 *st*+1*，*其中 *st*+1是从 *T* (*at, st, .*)分布中随机抽取的：

*st*+1 ∼ *T* (*at, st, .*)

4.我们在M中附加过渡 (*st, at, rt, st*+1)

5.我们随机抽取 B ⊂ M 个转换。 对于随机批次B的每个转换 (*stB , atB , rtB , stB* +1)

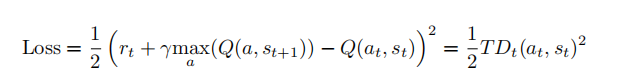
•我们得到的预测*：*

*Q*(*stB , atB* )

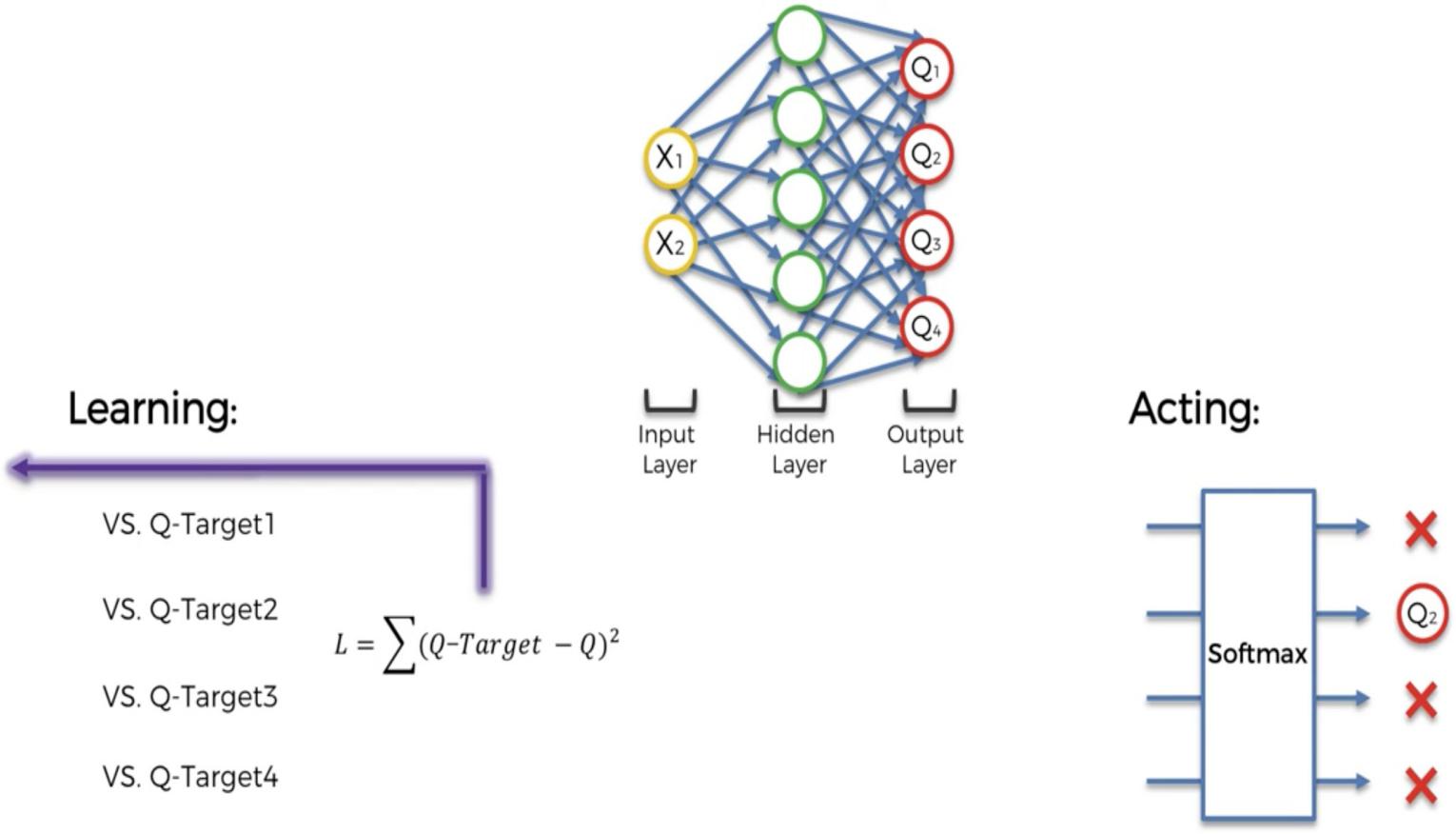
•我们达到目标：

*rtB* + *γ*max(*Q*(*a, stB* +1))

•我们得到损失：

**

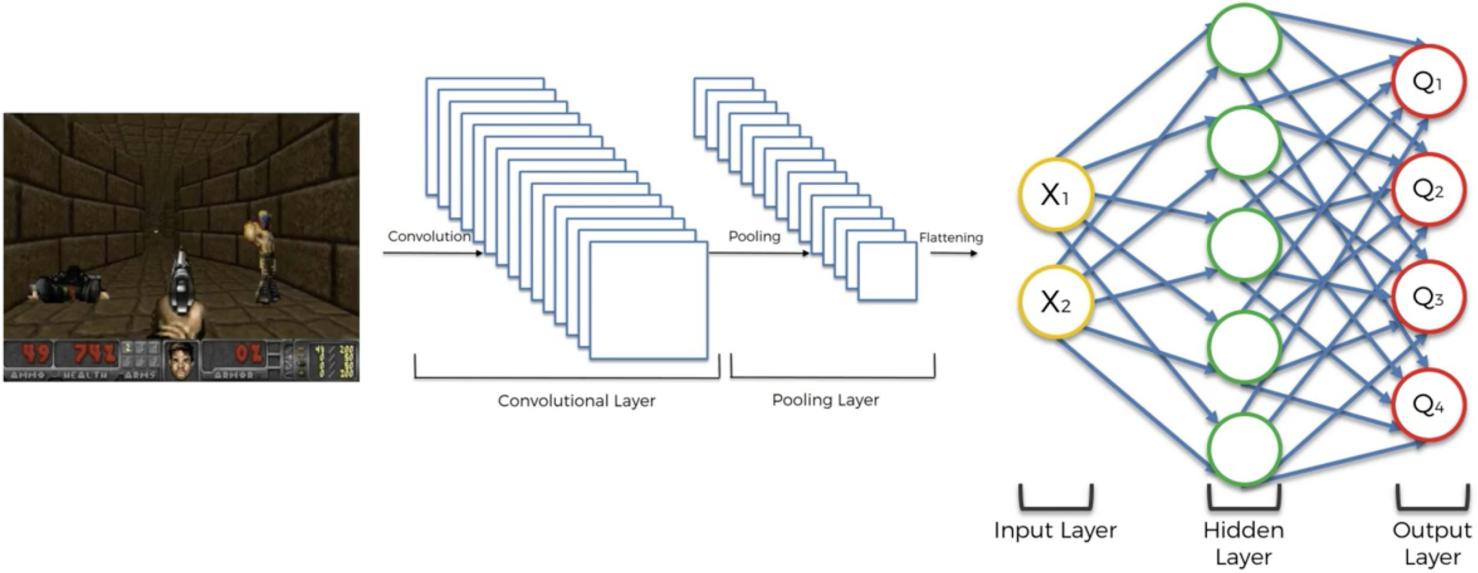
•我们反向传播该损失误差，并根据权重对误差的影响来更新权重。



# **深度卷积Q学习**

在上一节中，我们的输入是定义环境状态的矢量编码值。但是，由于编码的矢量无法保留图像的空间结构，因此这不是描述状态的最佳形式。空间结构确实很重要，因为它为我们提供了更多信息来预测下一个状态，而预测下一个状态对于我们的AI知道什么是正确的下一个动作当然必不可少。因此，我们需要保留空间结构，并且要做到这一点，我们的输入必须是3D图像（像素阵列为2D，颜色还要附加一个维）。在这种情况下，输入只是屏幕本身的图像，就像人类在玩游戏时所看到的一样。按照这种类比，AI就像人类一样：在玩游戏时观察屏幕的输入图像，输入图像进入卷积神经网络（人类的大脑），该神经网络将检测每个图像中的状态。但是，该卷积神经网络不包含池化层，因为它们会松散图像内对象的位置，当然，AI需要跟踪对象。因此，我们只保留卷积层，然后将它们展平为一维向量，就可以得到之前的 DeepQ-Learning网络的输入。然后，将运行相同的过程。

因此，总而言之，深度卷积Q-Learning与 Deep Q-Learning相同，唯一的区别是输入现在是图像，并且在完全连接的 Deep Q-Learning网络的开头添加了卷积神经网络。检测图像的状态（或仅仅是对象）。



# **异步演员批判代理（A3C）**

## 7.1A3C介绍

到目前为止，每次播放的动作都是一个神经网络的输出，就好像只有一个代理在决定玩游戏的策略一样。 A3C不再是这种情况。这次，我们将有几个代理，每个代理都与自己的环境副本进行交互。假设有n个特工 *A*1, *A*2,..., *An。*

每个代理共享两个网络：演员和评论家。评论家评估当前状态，而演员评估当前状态下的可能值。演员用来做决定。在每次培训on agent的时期，它都会使用共享网络的最新版本，并在n个步骤中使用actor来做出决定。在n个步骤中，它收集所有观察到的新状态，这些新状态的值，奖励等。在n个步骤之后，代理使用收集到的观察值来更新共享模型。纪元的时间，以及因此由代理更新共享网络的时间，是不同步的，因此是名称。

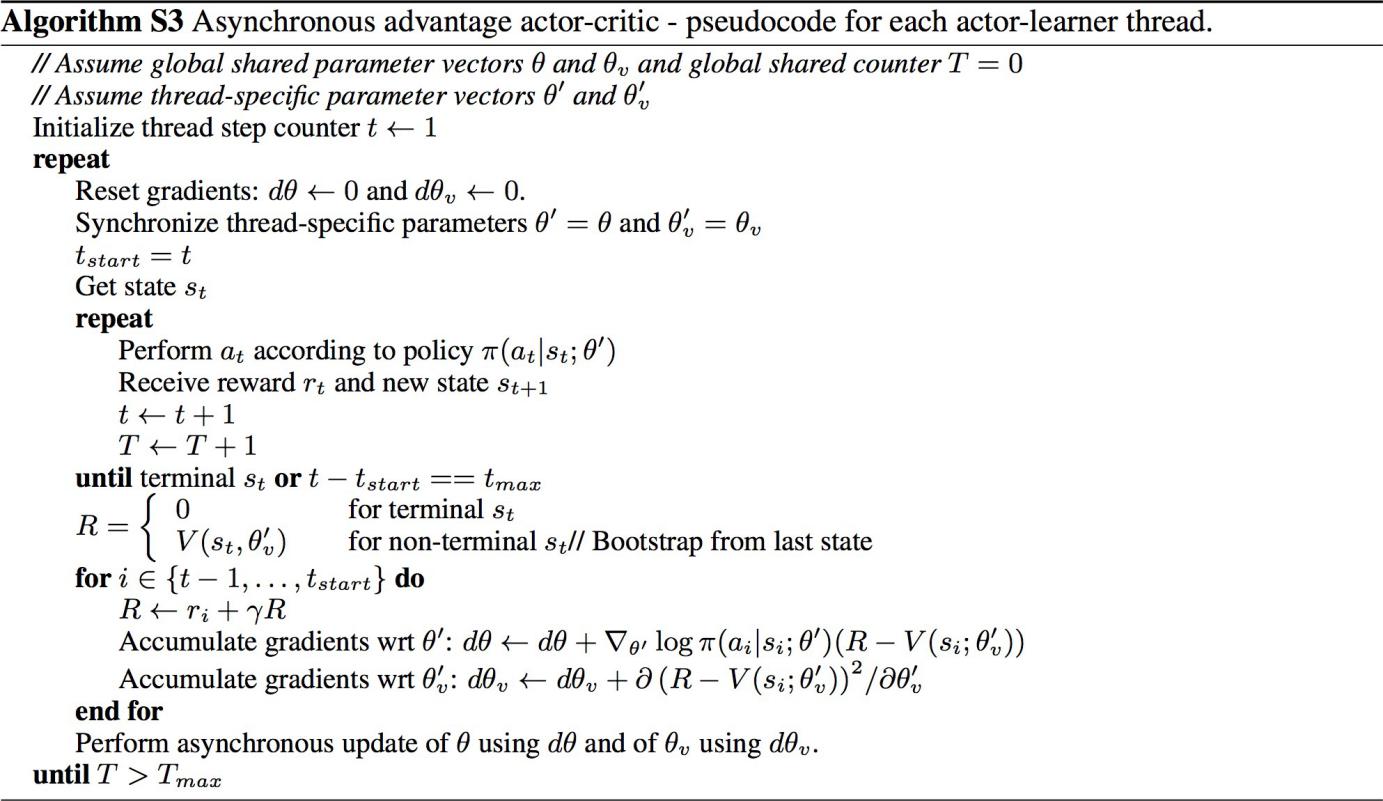
这样，如果不幸的代理开始陷入次优但有吸引力的策略中，它将达到该状态（因为其他代理在代理陷入困境之前也更新了共享策略），并将继续进行有效的探索。

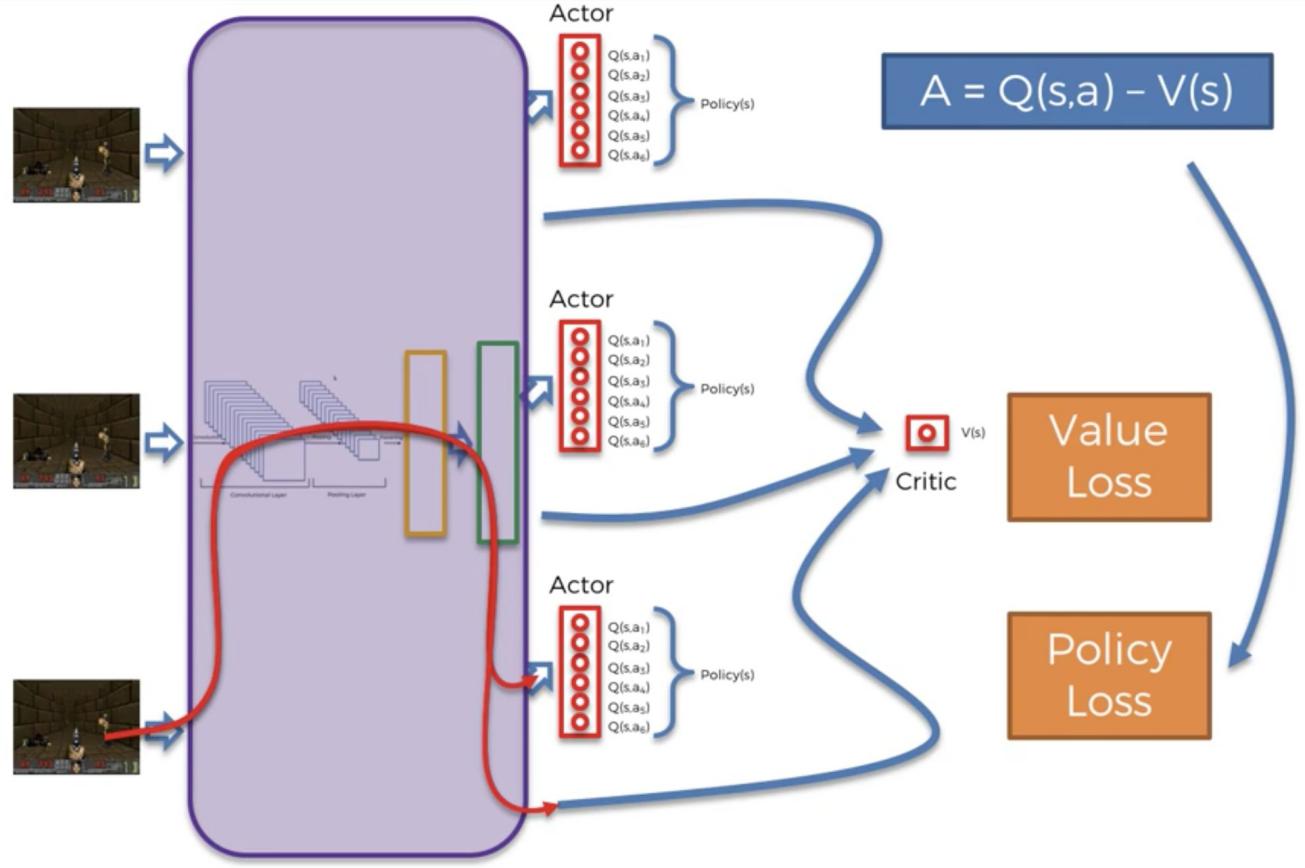
为了解释演员和评论者的更新规则，让我们将网络视为依赖于参数θ

（对于演员）和 θ*v*（对于评论者）的向量的函数。

## 整个A3C流程

官方的A3C算法是Google DeepMind的论文之一，“深度强化学习的异步方法”（https://arxiv.org/pdf/1602.01783.pdf）。在本文中，您可以在以下S3算法中找到它：





在上图中，我们可以清楚地看到三个A3C版本：

•异步：有多个代理，每个代理都有自己的环境副本，并且全部异步（在不同时间玩游戏）。

•优势：优势在于演员的预测 *Q*(*s, a*)与批评者的预测 *V* (*s*)之间的差异：

*A* = *Q*(*s, a*) − *V* (*s*)

**•演员批评：**我们当然可以看到演员和评论家，因此产生了两种不同的损失：政策损失和价值损失。 政策损失是与参与者的预测有关的损失。 价值损失是与批评者的预测有关的损失。 在训练的许多时期中，这两个损耗将反向传播到神经网络中，然后通过优化器通过随机梯度下降来减少。

# **结论**

如今，最强大的AI模型是A3C。 但是研究正在迅速取得长足进展，因此谁知道A3C是否可能在几年后就已经过时了。 到2029年，该课程可能会被称为“弱人工智能A-Z”，因为如果我们相信Ray Kurzweil的预测（到目前为止一直是正确的），那么我们将拥有人类级别的AI（可能会有良心） 2020年代末。 因此，恭喜您接受了学习AI的挑战，因为有一天您可能会扮演重要的角色。