DQN玩像素鸟

2017级软件工程1班
吴吉洲
3016207523

2020 年 2 月 16 日

目录

1 前言 **3**

2 问题描述 **3**

3 Q-Learning算法 **3**

4 训练设计 **4**

4.1 游戏模拟器 4
4.2 输入图像预处理 5
4.2.1 预处理图像 5
4.3 动作选择模块 5
4.4 训练过程伪代码 6
4.5 神经网络结构 6
4.6 训练的三个阶段 7
4.6.1 观察期 7
4.6.2 探索期 8
4.6.3 训练期 9

5 结果 **9**

1 前言

强化学习的主要目的是研究并解决机器人智能体贯序决策的问题。即智能体处在一个环境中，随着时间的流逝，不断地自我学习，并最终在这个环境中学到一套合理的行为策略。而深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，以发现数据的分布式特征表示，使得机器学习模型可以直接学习概念。而深层卷积神经网络应用在处理计算机视觉问题上已经获得了很大成功。

现实中的很多问题不允许我们通过预先的编程来模拟所以可能的情景，而且状态和输入过多，导致一些非神经网络算法难以实现。在这种情况下，就可采用强化学习结合深度神经网络的方式来解决问题。

2 问题描述

FlappyBird游戏中玩家控制的对象是一只像素构成的小鸟，在游戏运行的每一个时刻，玩家可以做的只有一个动作：按下跳跃键使得小鸟向上跳跃。由于游戏中是有重力存在的，所以未按下任何按键时小鸟会加速下降。玩家就这样通过操作小鸟的一跳一落，让小鸟在向前飞行的过程中通过尽量多的障碍以获得更高的分数。

我们的目标是让我们的机器像人类一样去玩这个游戏，所以在游戏过程中，我们不给机器提供游戏内部数据的接口。一方面，机器所能做的所有动作就是让小鸟向上跳越，另一方面，机器只能获得我们提供的图片像素信息以及作出动作后得到的反馈。在这个条件下，我们真正让计算机模拟了人类的学习过程。

3 Q-Learning算法

强化学习的目标是使总回报（reward）最大化。Q-Learning是off-policy的，迭代更新使用的是贝尔曼方程

$$New \quad Q(s,a) = Q(s,a) + \alpha[\Delta Q(s,a)]$$

$$\Delta Q(s,a) = R(s,a) + \gamma Q(s',a') - Q(s,a)$$

Q-Learning维护了一个Q表和一个R表。R表记录了每个状态(state)下执行每个动作(action)的回报(reward)，是预先根据环境设定好的，举个游戏的例子来说，小鸟飞过一个障碍可以认为得到了+1分的回报，小鸟死亡则是得到了一个-100的回报。 Q表则是用于记录给定状态，执行动作的回报期望。 Q-Learning就是通过不断尝试，根据回报修改Q表，最终Q表会收敛。这个最终的Q表也就是我们给定策略下的最佳决策。

DQN是为了解决当状态空间很大时，计算机内存无法存储Q表的问题。我们用卷积神经网络来构造一个函数代替Q表，以下列出损失函数及其梯度下降的函数。

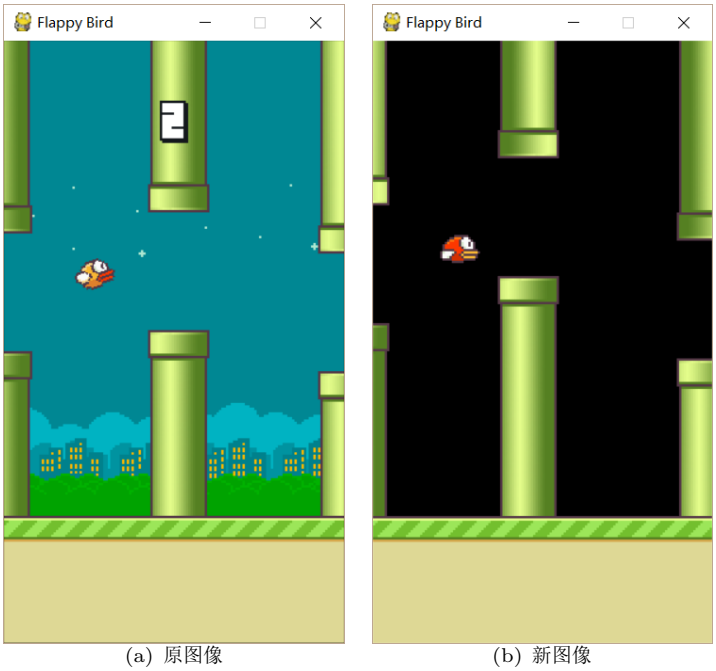
$$L = \sum_{s,a,r,s'} (Q(s,a;\theta) - (r + \gamma \max_{a'} Q(s',a';\theta^-)))^2$$

$$\Delta_\theta L = \sum_{s,a,r,s'} -2(Q(s,a;\theta) - (r + \gamma \max_{a'} Q(s',a';\theta^-)))\Delta_\theta Q(s,a;\theta)$$

4 训练设计

4.1 游戏模拟器

本次作业使用了python中的pygame模块完成Flappy Bird游戏程序，使用的是网络开源的代码。为了简化训练过程，做了一些小修改，去除了游戏背景，采用纯黑色背景代替，并且暂时抹去了计分用的数字，以避免干扰。并使得在小鸟死亡后自动重新开始新一轮游戏，而不是弹出结束界面。



(a) 原图像 (b) 新图像

图 1: 对比图

4.2 输入图像预处理

为了节省内存，便于训练，选择将图像缩小为80×80的大小，背景层用纯黑色代替，然后进行二值化，以去除噪声。

同时，基于对游戏的理解，在小鸟的运动过程中我们不仅需要知道小鸟的当前位置，还需要知道小鸟当前的状态。例如，小鸟是刚刚跃起还是正在下落，又或是停留在跳越的最高点。因为不同的状态关系到小鸟的速度，从图像帧的角度来看，即帧之间小鸟的位移大小。所以这里我们需要多帧作为一个状态，在当前帧进入一个状态之前，处理几帧图像叠加组合的多维图像数据，以便了解关于小鸟当前运动轨迹的信息。

4.2.1 预处理图像

1. 转化为80x80大小
2. 转化为灰度图
3. 把图像二值化为黑白两色，即0或者255
4. 将图像转化为80x80x1的矩阵。
5. 选取4帧图像作为一个State

这里用到了opencn中的2个图像处理函数

- cv.cvtColor(src , flag) 其中flag是转换类型,这里使用了cv2.COLOR_BGR2GRAY转化为灰度图
- cv2.threshold(src, thresh, maxval, type[, dst]) 其中thresh是阈值（起始值），maxval是最大值，type是算法类型。这里使用了THRESH_BINARY，即大于阈值则取maxval，否则取0。

4.3 动作选择模块

为ϵ贪心策略的简单应用，以概率ϵ随机从动作空间A中选择动作，以$1-\epsilon$概率依据神经网络的输出选择动作。

4.4 训练过程伪代码

Algorithm 1 DQN algorithm for Flappy-Bird

repeat
  开始新一轮游戏
  初始化初始状态 s_0
  repeat
    提取图像帧 x_t
    用x_t来更新s_t
    按照ϵ的概率选取随机动作a_t
    否则，选取$a_t = arg\max_{a \in actions} Q(s_t, a)$
    执行a_t
    向Replay Memory添加四元组 $e_t = (\phi(s_{t-1}), a_{t-1}, r_{t-1}, \phi(s_t))$
    修改探索期的ϵ值
    从replay memory随机选取一批数据miniBatch
    repeat
      if 游戏还未结束 **then**
        更新Target-DQN $Q'(s, a) \leftarrow Q(s, a)$
      end if
    until miniBatch全部使用过了
    用梯度下降法更新DQN的weights
    更新游戏模拟器的环境
  until 小鸟死了
  重新开始游戏
until 收敛或达到指定次数

4.5 神经网络结构

用卷积神经网络对游戏画面进行特征提取，也可以理解为对状态空间进行特征提取。神经网络的结构如下图。我们通过获得输入s，神经网络能够：

- 输出Q(s, a1)和Q(s, a2),比较两个值的大小然后选择更优的动作
- 在选择完动作后，根据游戏模拟器状态更新返回回报值，把这个状态转移过程存储进replay_memory中，在其中抽样来更新网络参数。

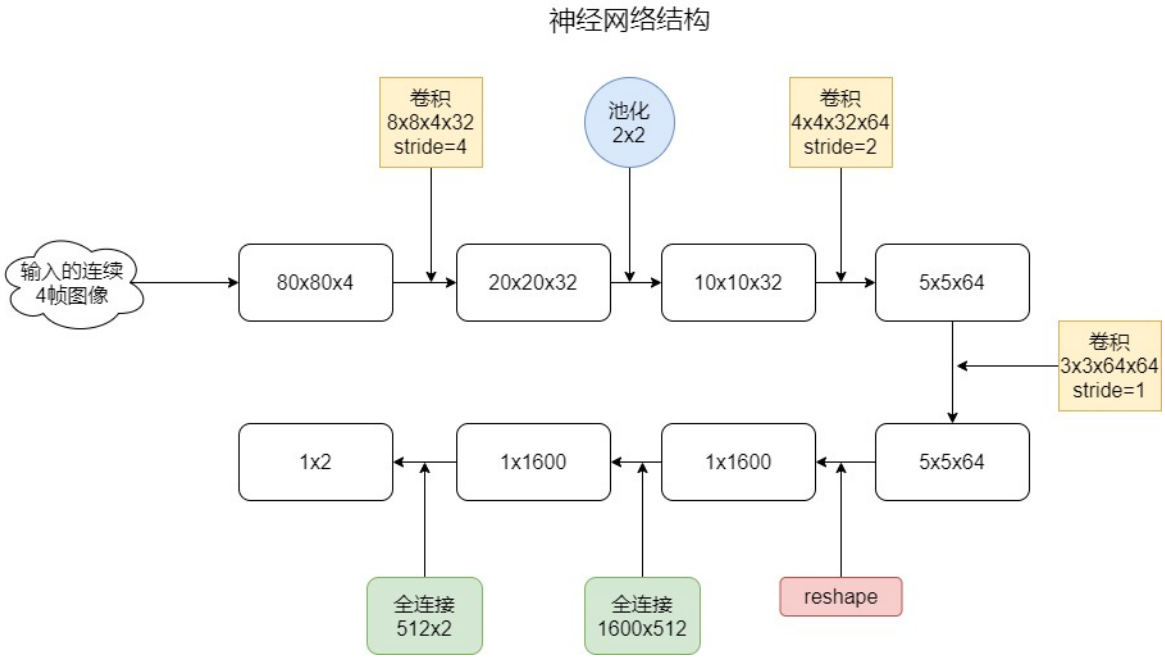


图 2: 神经网络结构

4.6 训练的三个阶段

4.6.1 观察期

程序与模拟器进行交互，随机给出动作，获取模拟器中的状态，将状态转移过程存放在D(Replay Memory)中。

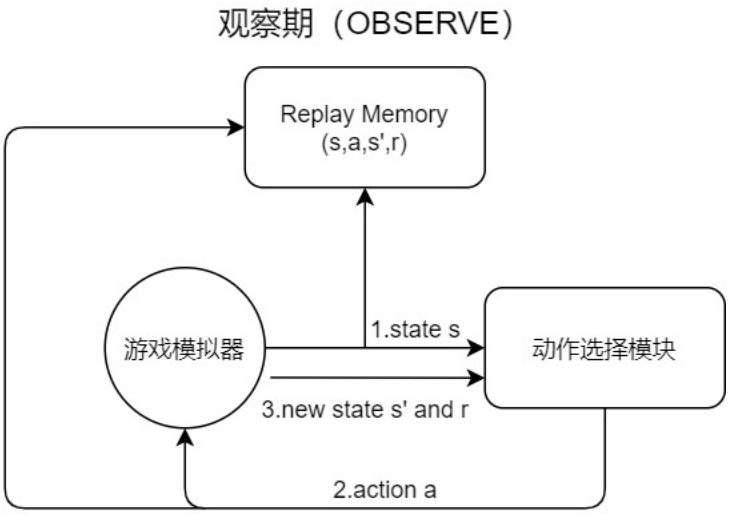


图 3: 观察期

4.6.2 探索期

程序与模拟器进行交互的过程中，依据Replay Memory中存储的历史信息更新网络参数。动作选择不再只是随机，而是依照随机探索率ϵ可能作出随即动作或由神经网络计算的动作。在训练的过程中会逐步降低随机探索率ϵ

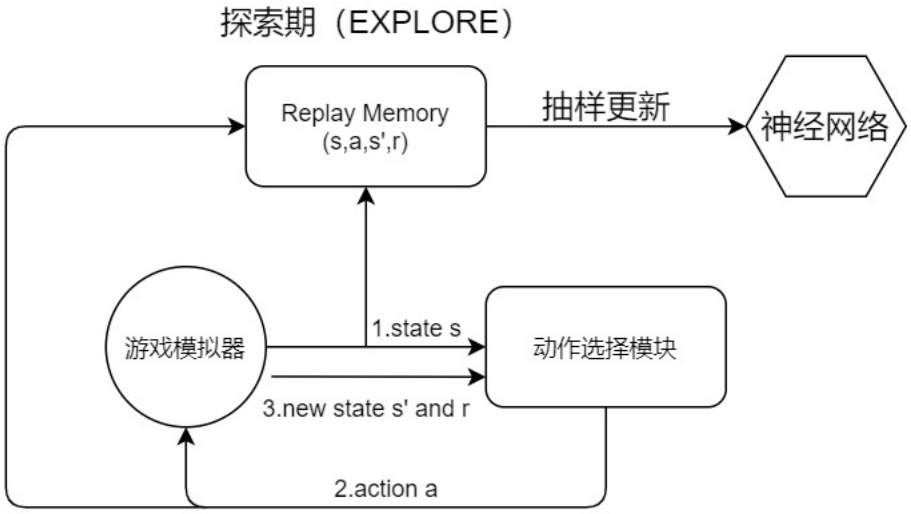


图 4: 探索期

4.6.3 训练期

ϵ已经足够小，不再发生改变，网络参数随着训练过程逐渐收敛。与探索期的区别是，训练期不修改ϵ.

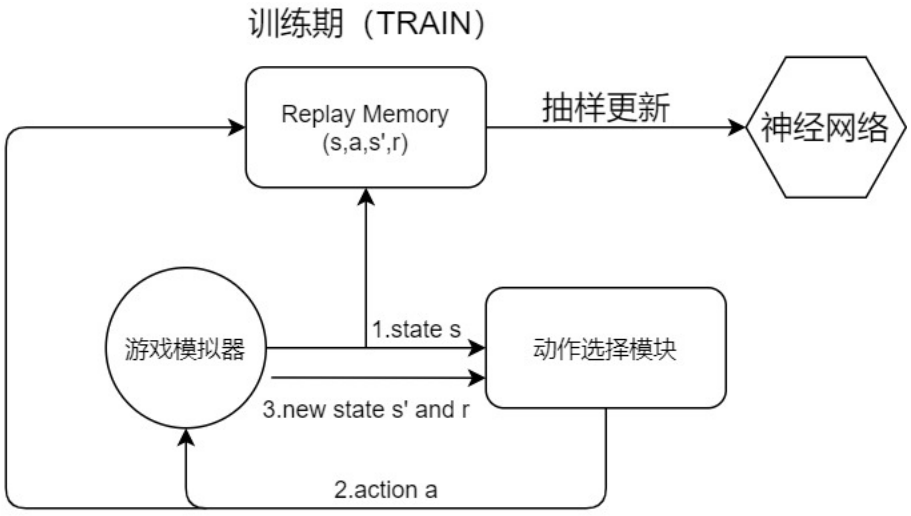


图 5: 训练期

5 结果

训练完成后，修改超参数INITIAL_EPSILON=0，这样可以避免作出随机动作，直接执行神经网络输出的动作，以此来观察训练的结果。

在训练了10万timestep后，小鸟已经可以通过2 3个障碍。在训练达到40万timestep的时候已经能够顺利通过10个左右的障碍，训练达到60万个timestep的时候已经很难死掉了。

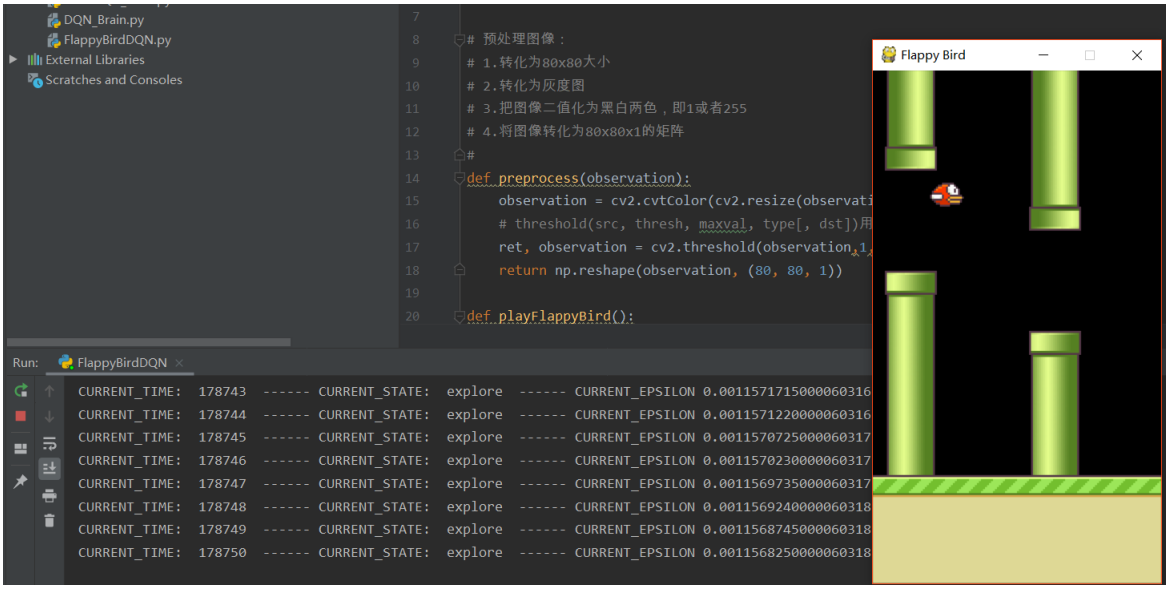

DQN_Brain.py
FlappyBirdDQN.py
External Libraries
Scratches and Consoles
预处理图像：
1.转化为80x80大小
2.转化为灰度图
3.把图像二值化为黑白两色，即1或者255
4.将图像转化为80x80x1的矩阵
#
def preprocess(observation):
observation = cv2.cvtColor(cv2.resize(observati
# threshold(src, thresh, maxval, type[, dst])用
ret, observation = cv2.threshold(observation,1,
return np.reshape(observation, (80, 80, 1))
def playFlappyBird():
Run: FlappyBirdDQN
CURRENT_TIME: 178743 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011571715000060316
CURRENT_TIME: 178744 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011571220000060316
CURRENT_TIME: 178745 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011570725000060317
CURRENT_TIME: 178746 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011570230000060317
CURRENT_TIME: 178747 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011569735000060317
CURRENT_TIME: 178748 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011569240000060318
CURRENT_TIME: 178749 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011568745000060318
CURRENT_TIME: 178750 ------ CURRENT_STATE: explore ------ CURRENT_EPSILON 0.0011568250000060318
Flappy Bird

图 6: 训练结果