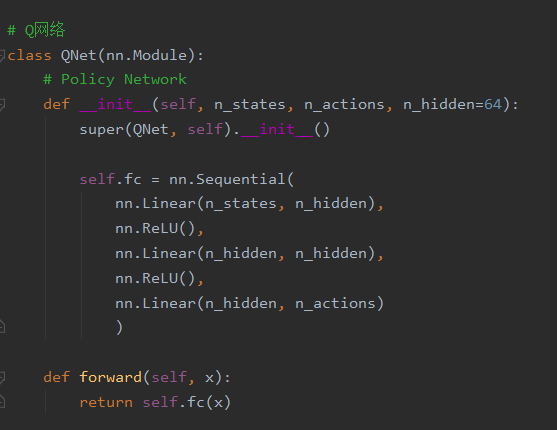
强化学习是一个反复迭代的过程，每一次迭代要解决两个问题：给定一个策略求值函数，和根据值函数来更新策略。DQN使用神经网络来近似值函数，即神经网络的输入是state s,输出是Q(s,a),∀a∈AQ(s,a),∀a∈A(action space)。Q网络只有一个隐藏层的原因：1989年Robert Hecht-Nielsen证明了对于任何闭区间内的一个连续函数都可以用一个隐含层的BP网络来逼近，这就是万能逼近定理。

Q网络代码：

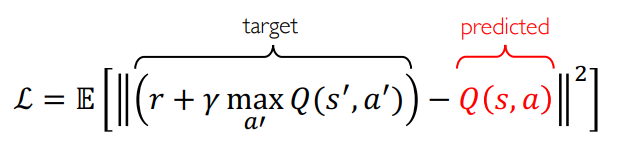


通过神经网络计算出值函数后，DQN使用贪心算法来从备选项中选择action。首先环境会给出一个obs，智能体根据值函数网络得到关于这个obs的所有Q(s,a)，然后选择action并做出决策，环境接收到此action后会给出一个奖励Rew及下一个obs。这是一个step。此时我们根据Rew去更新值函数网络的参数。接着进入下一个step。如此循环下去，直到我们训练出了一个好的值函数网络。

那么每次迭代如何更新神经网络的参数呢？

与机器学习类似，首先会定义一个loss function，然后使用梯度下降来更新参数。接下来首先介绍DQN的loss function，它与Q-Learning的非常类似，只是添加了一个target Q function。

Loss Function



这个公式表面上看起来很复杂，实际上很好理解，它就是一个残差模型，和我们平常见的最小二乘法很类似，真实值与预测值之间的差的平方。预测值就是Q(s,a;ω)，它是神经网络的输出。

不知道环境模型的情况下的强化学习算法成为Model Free RL：无法确定环境模型。那么在不知道环境模型的情况下如何求解值函数呢？答案就是采样。DQN属于Model Free的强化学习算法，它需要采样。它对值函数的估计只依赖下一个step的值函数。下面介绍从DQN的更新值函数的方式。

在值函数近似中，我们都是使用target来替代真实的值函数，即Loss中的真实值。

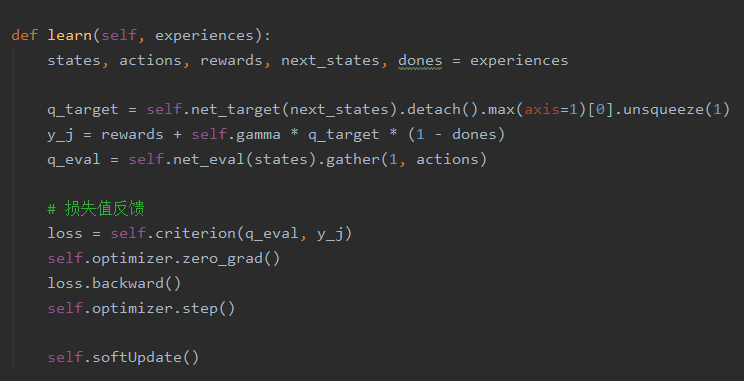
在当前状态下选择行为at+1，随即又用at+1处对应的值函数来计算target，更新上一步的值函数。这种学习方式又称为On-policy。DQN属于Off-policy Learning。

而与Q-Learning不同的是，Q-Learning中用来计算target和预测值的Q是同一个Q，也就是说使用了相同的神经网络。这样带来的一个问题就是，每次更新神经网络的时候，target也都会更新，这样会容易导致参数不收敛。回忆在有监督学习中，标签label都是固定的，不会随着参数的更新而改变。

因此DQN在原来的Q网络的基础上又引入了一个target Q网络，即用来计算target的网络。它和Q网络结构一样，初始的权重也一样，只是Q网络每次迭代都会更新，而target Q网络是每隔一段时间才会更新。DQN的target是Rt+1+γmaxa′Q(St+1,a，ω−)。用ω−表示它比Q网络的权重ω更新得要慢一些。

理解了DQN的target之后也就可以理解DQN的Loss Function了。

神经网络学习的代码：



DQN在Q-Learning上所做的改进

相比于Q-Learning，DQN做的改进：使用了卷积神经网络来逼近行为值函数，使用了target Q network来更新target，还有一个是使用了经验回放Experience replay。

伪代码：

首先初始化Memory D，它的容量为N;

初始化Q网络，随机生成权重ωω;

初始化target Q网络，权重为ω−=ωω−=ω;

循环遍历episode =1, 2, …, M:

初始化initial state S1S1;

循环遍历step =1,2,…, T:

用ϵ−greedyϵ−greedy策略生成action atat：以ϵϵ概率选择一个随机的action，或选择at=maxaQ(St,a;ω)at=maxaQ(St,a;ω);

执行action atat，接收reward rtrt及新的state St+1St+1;

将transition样本 (St,at,rt,St+1)(St,at,rt,St+1)存入D中；

从D中随机抽取一个minibatch的transitions (Sj,aj,rj,Sj+1)(Sj,aj,rj,Sj+1)；

令yj=rjyj=rj，如果 j+1j+1步是terminal的话，否则，令 yj=rj+γmaxa′Q(St+1,a′;ω−)yj=rj+γmaxa′Q(St+1,a′;ω−)；

对(yj−Q(St,aj;ω))2(yj−Q(St,aj;ω))2关于ωω使用梯度下降法进行更新；

每隔C steps更新target Q网络，ω−=ωω−=ω。

End For;

End For.