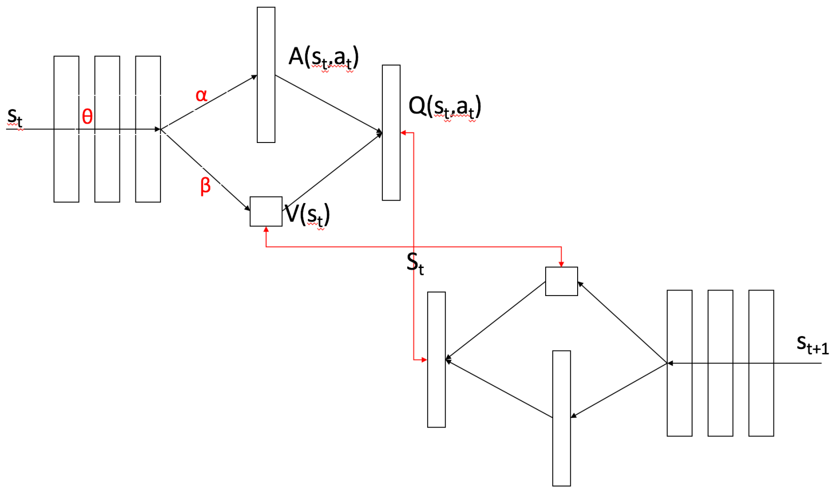
本模型基于icml 2016 best paper：dueling DQN，在它的基础上对模型进行了改进。

1. 模型简介



模型如上图，其中红色的双向剪头表示回归。

模型基于 dueling DQN，该方法的目的是增强学到的 Q function 对某 state 下不同 action 的泛化性，给出了两种约束 advantage function A 的方法，其中一种是对 A 加 normalization，此时 V 就是标准的 state value function。但是这种方法存在的问题是，当利用 DQN 的 loss 对 Q 进行梯度下降（或上升）时，并不确定有多少梯度传递给了 V，有多少给了 A。举个例子目前的 V(s)=3，A(s,a)=0，则 Q(s,a)=3，通过 target 网络得到 Q 的 target 为 4，此时需要进行梯度上升使 Q 增大，可能更新为 V(s)=3.5，A(s,a)=0.5，但其实并不是因为低估了 V 导致 Q 过低，则令 V 增大是没有意义的，在以后的 step 中需要再减小 V，这会导致震荡，不能很快的收敛。

我的思路是先对 V function 进行回归训练，训练参数 θ、β，然后固定这两个参数，对整体的 Q function 进行回归训练，训练参数 α。

2. 理论推导

如果没有 advantage function 那一部分，本模型就是标准的 value iteration，（见增强学习 memo P9，可参考 policy iteration）。也就是说 state value function V(s|θ,β)是一定可以学到的：

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi [G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \forall s \in \mathcal{S},
\end{aligned}
$$

由于是 model-free 模型，所以要基于 policy π 与环境互动得到的(s,a,r,s’)数据，增量的进行回归训练：V(s|θ,β) = r + V(s’|θ^,β^)，其中右侧 target 网络的参数是固定的，训练一段时间后，将参数 θ,β 赋给 target 网络。

Advantage function 能不能像 V 一样，A(s,a)由 A(s’,a’)加上 reward 作为 target 训练呢？相当于将 Q 拆成两个部分，分别定义 loss 去训练。这是不可以的。

已知 Q 的 bellman optimal equation 为（见 memo 2.15）：

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,\Big|\, S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma \max_{a'} q_*(s',a')\Big]. \end{aligned}$$

为了方便，我们将数据记做(s,a_1,r,s')，并对所有 value function 省略 π。可以对 $A(s,a_1)$进行如下推导：

$$\begin{aligned} A(s,a_1) &= q_*(s,a_1) - v(s) \\ &= \sum_a \pi(a|s)[\sum_{s',r} P(s',r|s,a_1)(r + max_{a'}q_*(s',a') - \sum_{s',r} P(s',r|s,a)(r+v(s'))] \\ &= \sum_a \pi(a|s)[\sum_{s',r} P(s',r|s,a)\frac{P(s',r|s,a_1)}{P(s',r|s,a)}(r + max_{a'}q_*(s',a') - \sum_{s',r} P(s',r|s,a)(r+v(s'))] \\ &= \sum_a \pi(a|s)\sum_{s',r} P(s',r|s,a)[\alpha(r + max_{a'}q_*(s',a')) - (r+v(s'))] \\ &= \sum_a \pi(a|s)\sum_{s',r} P(s',r|s,a)[\alpha(max_{a'}q_*(s',a') - v(s')) + (\alpha-1)(r+v(s'))] \\ &= \sum_a \pi(a|s)\sum_{s',r} P(s',r|s,a)[\alpha max_{a'}A(s',a') + (\alpha-1)(r+v(s'))] \end{aligned}$$

其中：

$$\alpha = \frac{P(s',r|s,a_1)}{P(s',r|s,a)}$$

我们发现 A(s,a1)取决于大括号中的部分，α 是一个大于 0 的值，但由于 reward 可以为负值，A(s,a1)与 α 不确定是正相关或者负相关，所以直接训练这一部分是困难的。我们可以固定已经训练完的 V 的参数，对 Q 进行回归训练，从而得到 A。

3. loss function

在第 i 个 iteration，关于 V 的 loss 为：

$$L_i(\theta_i,\beta_i) = E_{s,a,r,s'\sim U(d)}[(r + \gamma V(s'|\theta_i^-,\beta_i^-) - V(s|\theta_i,\beta_i))^2]$$

关于 A 的 loss 为：

$$L_i(\alpha_i) = E_{s,a,r,s'\sim U(d)}[(r + \gamma max_{a'}Q(s',a'|\alpha_i^-) - Q(s,a|\alpha_i))^2]$$

4. 应用场景

由于是很小的改动，可能撑不起一篇纯做增强学习模型的文章，DQN 和 dueling DQN 都是在好几十个游戏上跑实验的，这样结果看起来比较靠谱，但是很费时间。但理论上是可以做的，因为这个方法可拓展性和 dueling DQN 一样，DQN 能有的改进，都可以应用上去。

或者用到有序列数据的 POI 预测上，将 user 用社交网做 embedding，将地点用物理距离构建的网络做 embedding。State 为地点的 embedding，action 为在某一地点预测的下一地点，预测错误则 reward 为 0，继续预测，直到预测正确，

reward 为 1，V 为用户对每个地点的偏好，Q 为用户从一地点到另一地点的概率。将某一时刻 user 和地点的 embedding 作为输入（user 的使用参考 linUCB），然后得到 Q(s,a)。本方法是基于 TD(0)，可以改成基于 TD(λ)，也就是不只用一步的 reward。但是这样基本上和 RNN 完全没有区别，只是套了个迁移学习的壳。现在的问题是找不到合适的任务，最好满足 1. Action 和 state 不是同一个东西 2.环境具有更多的随机性，这种 model-free 的方法会更能发挥优势。