第一章 函数近似

1.1 设计思想

1. 网格环境定义

   网络大小：NUM_STATES = GRID_ROW * GRID_COL

   网格结构：使用一维向量 grid 定义网格世界，包含不同的状态和对应的奖励。状态可以是普通状态、禁区、目标或边界。

   奖励设置：

   OTHERSTEP：普通状态的奖励。

   FORBIDDEN：禁区的惩罚。

   TARGET：目标状态的奖励。

   BORDER：越界惩罚。

   设置 gamma。

   设置 EPSILON。

2. 动作定义

   动作数 NUM_ACTION = 5。

   定义五种可能的动作：RIGHT、DOWN、UP、LEFT、STAY。每个动作对应一个整数值。

3. 初始化策略 policy，状态值 V

   policy：采用的策略，定义为大小[NUM_STATES, NUM_ACTION]的矩阵，其中 policy[s][a]表示在状态 s 时，选择动作 a 的概率。

   V：状态值向量，长度与状态个数相同，初始化为$v_i = 0$。

4. 函数近似

   使用 $\hat{v}_\pi(s,w)$近似网格世界$v(s)$，最小化目标函数$J(w) = E[\left(v_\pi(S) - \hat{v}(S,w)\right)^2]$，以找到最优 w，使用随机梯度下降算法进行最小化。迭代式为：

   $$w_{t+1} = w_t + \alpha_t\left(v_\pi(s_t) - \hat{v}(s_t,w_t)\right)\nabla_w \hat{v}(s_t,w_t)$$

   根据时序差分学习的思想，使用$r_{t+1} + \gamma\hat{v}(s_{t+1},w_t)$作为状态值$v_\pi(s_t)$的近似，算法变为：$w_{t+1} = w_t + \alpha_t\left(r_{t+1} + \gamma\hat{v}(s_{t+1},w_t) - \hat{v}(s_t,w_t)\right)\nabla_w \hat{v}(s_t,w_t)$

1.2 伪码描述

函数近似的时序差分学习

初始化：函数$\hat{v}(s,w)$在 w 处可微。初始参数w_0

目标：近似给定策略π的真实估计值

对于按照策略π生成的每个 episode，do

　　对于每步(s_t, r_{t+1}, s_{t+1})，do

　　$w_{t+1} = w_t + \alpha_t\big(r_{t+1} + \gamma\hat{v}(s_{t+1}, w_t) - \hat{v}(s_t, w_t)\big)\nabla_w \hat{v}(s_t, w_t)$

1.3 时间复杂度分析

特征向量 $\phi(s)$使用多项式。$\phi(s) \in \mathbb{R}^n$。

时间步长度 T，$\hat{v}(s,w)$计算$O(n)$，每个 episode 复杂度$O(T*n)$，共 K 个 episode 时间复杂度$O(K*T*n)$

1.4 结果分析

使用 5x5 网格世界，奖励设置为 $r_{boundary} = r_{forbidden} = -1$，$r_{target} = 1$，$r_{otherstep} = 0$。折扣率为$\gamma = 0.9$。

使用以下给定策略生成 500 个长度为 500 的 episode。使用多项式近似。

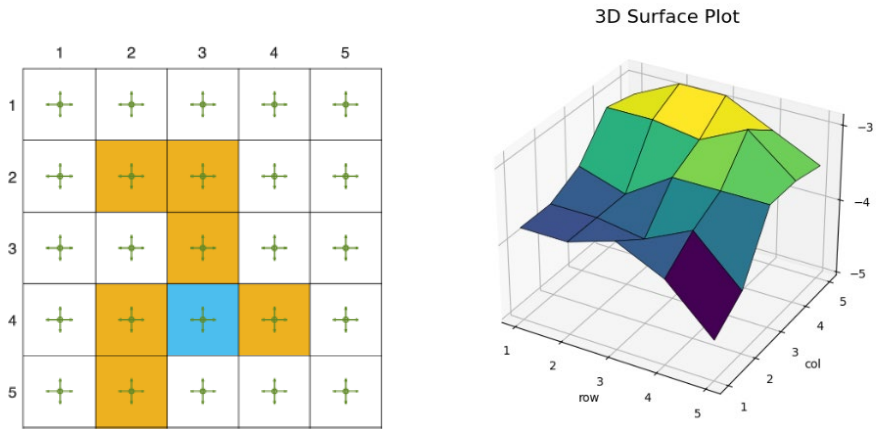
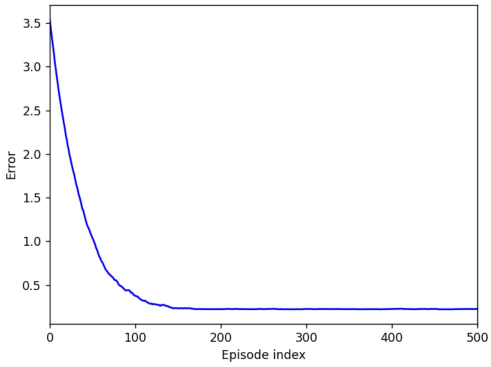
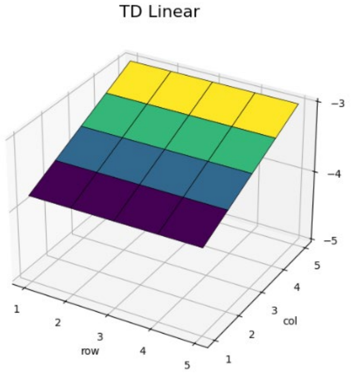


图 1 给定策略与真实 V 值（该策略表示每个动作概率相等）

1. $\Phi(s) = [1, x, y]^T \in \mathbb{R}^3$

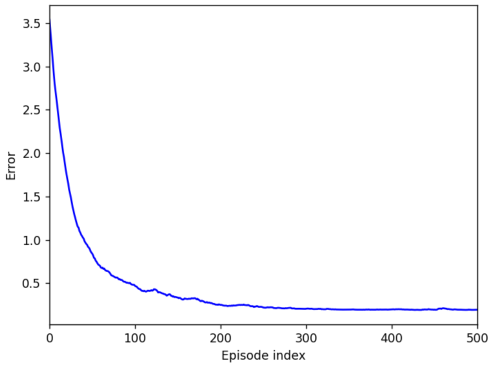


(a) Episode 损失曲线

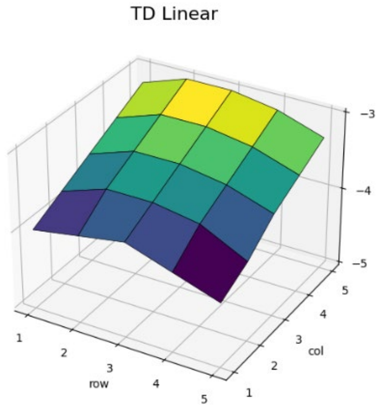


(b) 近似V

2. $\Phi(s) = [1, x, y, x^2, y^2, xy]^T \in \mathbb{R}^6$

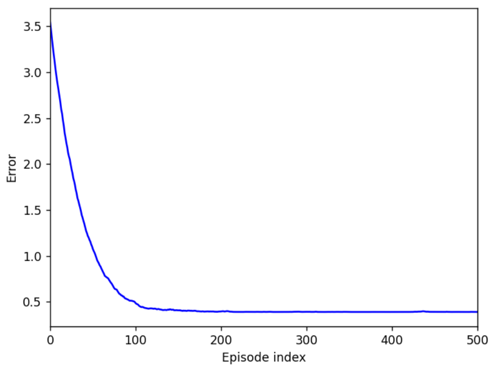


(c) Episode 损失曲线

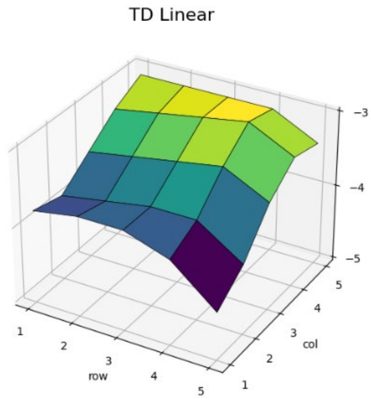


(d) 近似V

3. $\Phi(s) = [1, x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2]^T \in \mathbb{R}^{10}$



(e) Episode 损失曲线



(f) 近似V

从三组结果可以看出，使用$\Phi(s) \in \mathbb{R}^3$近似时，随着 episode 增加，误差逐渐减小，最终稍大于 0，近似得到的 V 和真实 V 有一定相似，但是近似能力有限，无法较好的表示每个状态值。增加近似维度，使用二次曲面$\Phi(s) \in \mathbb{R}^6$近似时，最终

误差值更小，同时近似的 V 与真实 V 更加接近。进一步增加维度，当$\Phi(s) \in \mathbb{R}^{10}$时，误差趋近于 0，近似能力较强，与真实 V 相似度较高，也能较好的表现出 V 对应 s 的变化。

上述例子说明，使用更为复杂的曲面近似，可以使近似得到的 V 更接近真实 V，但是同时也会增加需要的参数量以及计算复杂度，在项目中使用时需要考虑误差与计算复杂度的相互影响。

第二章 Deep Q-Learning (off-policy) 算法

2.1 设计思想

1. 网格环境定义

网络大小：NUM_STATES = GRID_ROW * GRID_COL

网格结构：使用一维向量 grid 定义网格世界，包含不同的状态和对应的奖励。状态可以是普通状态、禁区、目标或边界。

奖励设置：

OTHERSTEP：普通状态的奖励。

FORBIDDEN：禁区的惩罚。

TARGET：目标状态的奖励。

BORDER：越界惩罚。

设置 gamma。

设置 EPSILON。

2. 动作定义

动作数 NUM_ACTION = 5。

定义五种可能的动作：RIGHT、DOWN、UP、LEFT、STAY。每个动作对应一个整数值。

3. 初始化策略 policy，状态值 V

policy：采用的策略，定义为大小[NUM_STATES, NUM_ACTION]的矩阵，其中 policy[s][a]表示在状态 s 时，选择动作 a 的概率。

V：状态值向量，长度与状态个数相同，初始化为$v_i = 0$。

4. 经验回放缓冲区

使用行为策略π_b生成步长为 T 的一系列状态动作对(s,a)，将其对应的(s, a, s', r)存入经验回放缓冲区，用于后续 Deep Q-learning 采样。

5. Deep Q-learning off-policy

使用神经网络初始化主网络与目标网络。设置单隐层神经网络，隐层节点数为100，输入为三维（x,y,action），输出为 1 维 q_value。使用均方误差，Relu 激活函数。目标网络初始化结构与主网络相同。

从经验回放缓冲区中均匀取出 batch_size 大小样本。对于每个样本，使用目标网络计算目标值y_T。使用小批量更新主网络，最小化目标值与主网络输出误差$J = \left(y_T - \hat{q}(s,a,w)\right)^2$，最小化过程中更新主网络系数。

每经过一段时间采样，更新目标网络=主网络。

迭代结束后利用目标网络生成策略。

2.2 伪码描述

深度 Q-learning (off-policy)

目标：学习最优目标网络，使用行为策略π_b生成的经验样本来近似最优动作值。

将π_b生成的经验样本存储在回放缓冲区中$\mathcal{B} = \{(s,a,r,s')\}$

对于每次迭代，do

从 $\mathcal{B}$中均匀抽取一小批量样本

对于每个样本(s,a,r,s')，计算目标值$y_T = r + \gamma \max_{a \in A(s')} \hat{q}(s',a,w_T)$，其中$w_T$是目标网络的参数。使用小批量$\{(s,a,y_T)\}$更新主网络以最小化$\left(y_T - \hat{q}(s,a,w)\right)^2$

每 C 次迭代设置$w_T = w$

更新目标策略

2.3 时间复杂度分析

经验回放缓冲区大小 M，生成经验样本$O(M)$

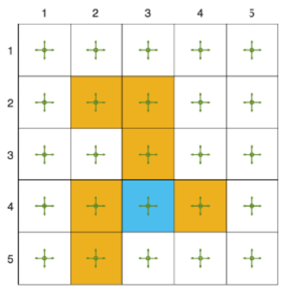
每个状态动作数 A，迭代次数 K，每个小批量大小 B，假设神经网络计算时间复杂度$O(Q)$，对每个样本计算目标值时间复杂度$O(A*Q)$，神经网络更新$O(B*Q)$，总时间复杂度$O(M) + O(K*A*Q + K*B*Q)$，根据 M、K*A*Q、K*B*Q 相对大小确定时间复杂度。

2.4 结果分析

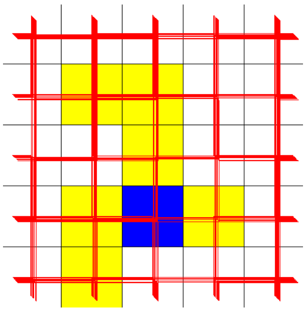
使用 5x5 网格世界，奖励设置为 $r_{boundary} = r_{forbidden} = -1$，$r_{target} = 1$，$r_{otherstep} = 0$。折扣率为$\gamma = 0.9$。 使用一个单独的 episode，根据同图 1 的策略

生成一个长度 T=1000 的结果，设置经验回放缓冲区大小为 1000。设置每 20 轮采样更新一次目标网络。

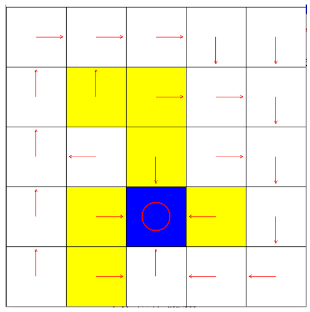
1. 长度 T=1000



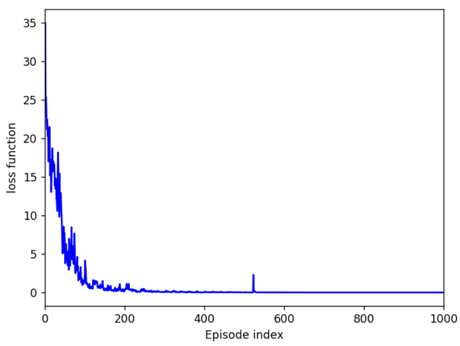
(a) 行为策略



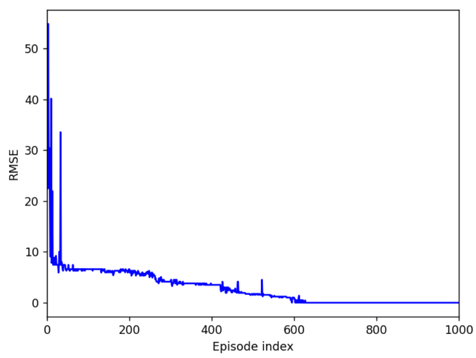
(b) episode 有 1000 步



(c) 最终策略



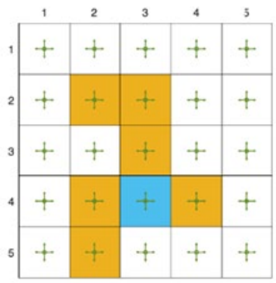
(d) 损失函数



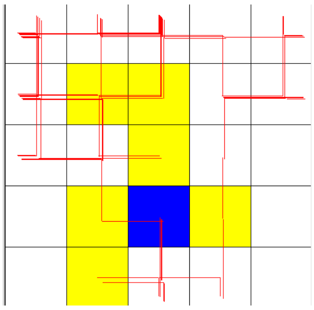
(e) 状态值估计误差

当设置时间步长度为 1000 时，使用探索策略（如图(a)）构造经验回放缓冲区，策略探索结果如图(b)所示，经过 1000 轮训练，损失函数结果、状态值估计误差分别如图(d)、(e)所示。从结果可以看出，训练初期损失函数值与状态值估计误差均较大，随着训练论数的增加，损失值逐渐趋于 0，状态值估计误差不断减小；当训练超过 500 轮，损失函数值趋近于 0；训练 600 轮之后，损失函数与状态值估计误差收敛为 0。最终得到图(c)的最优策略，与贝尔曼方程解出的最优策略一致，使用 1000 步的 DQN 策略可以求得最优策略。

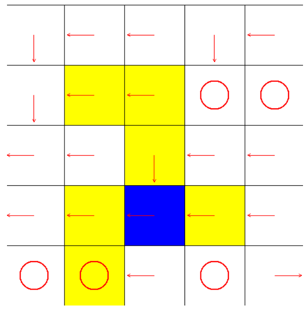
2. 长度 T=100



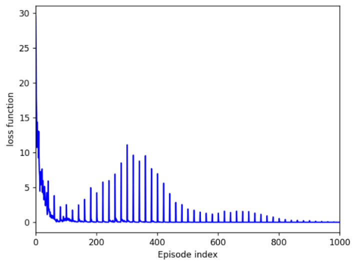
(f) 行为策略



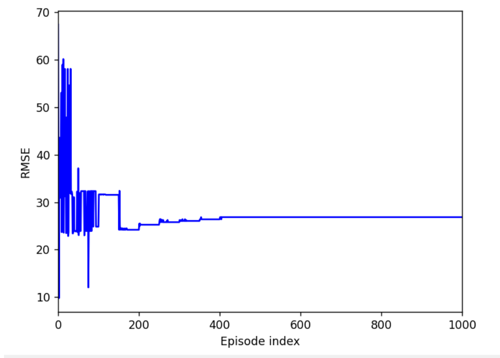
(g) Episode 有 100 步



(h) 最终策略



(i) 损失函数



(j) 状态值估计误差

当时间步长度为 100 时，探索的状态动作对如图(g)所示，仅访问了少数状态动作对，经验回放缓冲区中结果并不能实现均匀抽取每个状态动作对。经过 1000 轮训练，图(i)中损失函数值收敛到 0，而图(j)展示的状态值估计误差仍然维持在一个较大的值，无法趋近 0。得到最终策略如图(h)，与真实最优策略差异较大。

从上述结果可以看出，当使用 DQN 时，如果经验回放缓冲区中样本不是均匀分布，或不是均匀采样时，网络更新无法兼顾到全部情况，导致部分状态动作无法实现最优。由于目标函数采用随机梯度下降，每个状态动作对视为单个随机变量，假设其分布是均匀的，因此当使用经验回放时，需要剔除策略的影响。