编程作业1

实现值迭代算法、高斯-赛德尔值迭代算法、策略迭代算法，复现其在10x10栅格世界问题的结果

- **代码架构**：

  我创建了一个class:DP_on_GridWorld，初始化了转移概率T、最大迭代次数max_iter、行动方向dire、四个奖赏reward和四个奖赏位置reward_pos等。主要函数有：

  - `act(i,j,direction,U)`：表示在状态(i,j)处采取行动direction，根据公式

    $$R(s,a)+\gamma\sum_{s'}T(s'|s,a)U_k(s')$$

    返回在效用矩阵U上计算该状态的新效用值
  - `value_iteration`：值迭代算法
  - `gauss_seidel_value_iteration`：高斯-赛德尔值迭代算法
  - `policy_evaluation(policy,n)`：计算策略policy的n步期望回报
  - `policy_iteration`：策略迭代算法
  - `compute_best_policy(U,policy_matrix=None)`：根据效用矩阵U和策略矩阵policy_matrix计算最优策略
- **算法实现**：(具体实现详见`value&policy_iteration.py`文件)
  - **值迭代**：

```
U_old=np.zeros((self.N,self.N))  #效用矩阵U_k
U_new=np.zeros((self.N,self.N))  #效用矩阵U_{k+1}
while iter<self.max_iter:
    for i in range(self.N):
        for j in range(self.N):
            U_s_list=[]   #表示四个动作计算得出的新效用
            for k in range(4):
                U_s_list.append(self.act(i,j,self.dire[k],U_old))
            U_new[i,j]=max(U_s_list)  #对应伪代码中的求max操作
            iter+=1
            delta=U_new-U_old
            if(np.linalg.norm(delta,ord=np.inf)<self.epsilon):
                print(COLOR+"iter:%d,值迭代收敛,"%iter,BACK,end='')
                break
            U_old=U_new.copy()
```

  - **高斯-赛德尔值迭代**：

```
U_inplace=np.zeros((self.N,self.N))
while iter<self.max_iter:
    U_last_iter=U_inplace.copy() #保存上一轮迭代的结果，从而进行收敛性的判别
    #按状态矩阵的左下到右上顺序更新状态，因为有非0奖赏的状态集中在左下部分，这一更新顺序要比值迭代更快一些收敛。
    for i in range(self.N-1,-1,-1):
        for j in range(self.N-1,-1,-1):
            U_s_list=[]
            for k in range(4):
```

```
                #inplace更新状态
                U_s_list.append(self.act(i,j,self.dire[k],U_inplace))
            U_inplace[i,j]=max(U_s_list)
    iter+=1
    delta=U_inplace-U_last_iter
    if(np.linalg.norm(delta,ord=np.inf)<self.epsilon):
        print(COLOR+"iter:%d,高斯赛德尔值迭代收敛,"%iter,BACK,end='')
        break
```

- **策略评价**:

  与值迭代几乎相同，所不同的是没有打印信息且最后返回U_new

- **策略迭代**:

```
#10x10矩阵的每个元素是一个1x4小矩阵，对应于策略pi采取上下左右行动的概率,初始化为1/4
policy_old=np.ones((self.N,self.N,4))/4  #pi_k
policy_new=np.ones((self.N,self.N,4))/4  #pi_{k+1}
U_pi=np.zeros((self.N,self.N))
while(iter<self.max_iter):
    U_pi=self.policy_evaluation(policy_old,self.max_iter)
    for i in range(self.N):
        for j in range(self.N):
            U_s_list=[]
            for k in range(4):
                U_s_list.append(self.act(i,j,self.dire[k],U_pi))
            #表示新效用值中最大值的下标，对应伪代码的求argmax操作
            max_action_index=[]
            for k in range(4):
                if(U_s_list[k]==max(U_s_list)):
                    max_action_index.append(k)
            prob=1/len(max_action_index)
            #在pi_{k+1}中拥有最大效用值的动作平分概率，而其他动作概率为0
            for k in range(4):
                if k in max_action_index:
                    policy_new[i,j,k]=prob
                else:
                    policy_new[i,j,k]=0
    iter+=1
    if(np.all(policy_new==policy_old)):
        print(COLOR+"iter:%d,策略迭代收敛,"%iter,BACK,end='')
        break
    policy_old=policy_new.copy()
```

- **实验结果**:
  - **直接运行`value&policy_iteration.py`文件即可得到运行结果**
  - $\gamma=0.5$:
    - 收敛的效用矩阵为:

```
[[ -0.28  -0.13  -0.12  -0.11  -0.09  -0.04   0.08   0.31   0.07  -0.19]
 [ -0.13  -0.01   0.     0.02   0.07   0.18   0.46   1.11   0.45   0.07]
 [ -0.12  -0.     0.01   0.04   0.15   0.42   1.12   3.     1.11   0.31]
 [ -0.12  -0.01  -0.02  -0.24   0.05   0.19   0.47   1.12   0.48   0.09]
 [ -0.13  -0.02  -0.27  -5.12  -0.23   0.08   0.2    0.46   0.54   0.13]
 [ -0.12  -0.01  -0.04  -0.28   0.02   0.11   0.28   0.65   1.39   0.53]
 [ -0.12  -0.02  -0.06  -0.51   0.05   0.26   0.64   1.55   3.72   1.49]
 [ -0.13  -0.04  -0.53 -10.19  -0.33   0.5    1.39   3.72  10.     3.74]
 [ -0.14  -0.03  -0.07  -0.51   0.04   0.25   0.63   1.55   3.72   1.49]
 [ -0.28  -0.14  -0.15  -0.18  -0.1   -0.01   0.16   0.54   1.32   0.43]]
```

- 最优策略为：

```
[['right' 'down' 'down' 'down' 'down' 'down' 'down' 'down' 'down' 'down']
 ['right' 'right' 'right' 'right' 'right' 'right' 'down' 'down' 'down'
  'left']
 ['right' 'right' 'right' 'right' 'right' 'right' 'right' 'anydire'
  'left' 'left']
 ['right' 'up' 'up' 'right' 'right' 'right' 'right' 'up' 'left' 'left']
 ['right' 'up' 'up' 'right' 'right' 'right' 'up' 'up' 'down' 'left']
 ['right' 'down' 'left' 'right' 'right' 'right' 'right' 'down' 'down'
  'down']
 ['right' 'up' 'left' 'right' 'right' 'right' 'right' 'down' 'down'
  'down']
 ['right' 'up' 'left' 'right' 'right' 'right' 'right' 'right' 'anydire'
  'left']
 ['right' 'up' 'left' 'right' 'right' 'right' 'right' 'up' 'up' 'up']
 ['up' 'up' 'up' 'right' 'up' 'up' 'up' 'up' 'up' 'up']]
```

- $\gamma = 0.9$：
  - 收敛的效用矩阵为：

```
[[ 0.41  0.74  0.96  1.18  1.43  1.71  1.98  2.11  2.39  2.09]
 [ 0.74  1.04  1.27  1.52  1.81  2.15  2.47  2.58  3.02  2.69]
 [ 0.86  1.18  1.45  1.76  2.15  2.55  2.97  3.    3.69  3.32]
 [ 0.84  1.11  1.31  1.55  2.45  3.01  3.56  4.1   4.53  4.04]
 [ 0.91  1.2   1.09 -3.    2.48  3.53  4.21  4.93  5.5   4.88]
 [ 1.1   1.46  1.79  2.24  3.42  4.2   4.97  5.85  6.68  5.84]
 [ 1.06  1.41  1.7   2.14  3.89  4.9   5.85  6.92  8.15  6.94]
 [ 0.92  1.18  0.7  -7.39  3.43  5.39  6.67  8.15 10.    8.19]
 [ 1.09  1.45  1.75  2.18  3.89  4.88  5.84  6.92  8.15  6.94]
 [ 1.07  1.56  2.05  2.65  3.38  4.11  4.92  5.83  6.68  5.82]]
```

  - 最优策略为：

```
[['right' 'down' 'down' 'down' 'down' 'down' 'down' 'down' 'down' 'down']
 ['right' 'right' 'right' 'right' 'down' 'down' 'down' 'right' 'down'
  'down']
 ['right' 'right' 'right' 'right' 'right' 'down' 'down' 'anydire' 'down'
  'down']
 ['right' 'right' 'right' 'right' 'right' 'right' 'down' 'down' 'down'
  'down']
 ['right' 'down' 'down' 'right' 'right' 'right' 'down' 'down' 'down'
  'down']
 ['right' 'right' 'right' 'right' 'right' 'right' 'right' 'down' 'down'
  'down']
 ['right' 'right' 'right' 'right' 'right' 'right' 'right' 'right' 'down'
  'down']
 ['right' 'down' 'down' 'right' 'right' 'right' 'right' 'right' 'anydire'
  'left']
 ['right' 'right' 'right' 'right' 'right' 'right' 'right' 'right' 'up'
  'up']
 ['right' 'right' 'right' 'right' 'right' 'right' 'up' 'up' 'up' 'up']]
```