Assignment 1 Report

Anshul Pattoo

April 24, 2023

1 Markov Decision Processes

This section briefly describes both solution approaches, i.e., a linear solver to solve the system $Ax = b$ and a dynamic programming approach of policy iteration. State-value table results for the 5x5 case, with discount rates of 0.85 and 0.75, and the 7x7 case, with the same discount rates, are also shown.

1.1 Description of Linear Solver Implementation

The linear solver approach is written in `A1_Q1.py` and is implemented by the class `Linear_Solver()`, consisting of functions `prob_s_to_s_prime()`, `compute_reward_vec()`, and `linear_solver()`. The implementation is based on an algebraically manipulated version of the following fundamental matrix equation, $v = R^{\pi} + \gamma P^{\pi}_{s,s'} v$, where v is a vector containing values of the state-value table, R is the reward vector, and P is the transition matrix,

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{pmatrix} + \gamma \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

For convenience, the subscripts and superscripts of this matrix equation's symbols are omitted in our discussion. This matrix equation can be algebraically manipulated for our implementation as follows,

$$\begin{aligned} v &= R + \gamma P v \\ v - \gamma P v &= R \\ (I - \gamma P) v &= R \end{aligned}$$

The final line above, $(I - \gamma \cdot P)v = R$, meets the set-up of the linear system $Ax = b$, as indicated in the assignment problem. Where x is our unknown, the system $Ax = b$ is arranged and, using `linalg.solve()` from `numpy`, solved in `linear_solver()`, given A and b as arguments. Each of the components of $Ax = b$ refer to the following from the algebraic manipulation above,

- $A = I - \gamma \cdot P$
- $x = v$
- $b = R$

Providing further details,

- I is an identity matrix of N by N dimension, where N is the number of states in the grid. In particular, for the 5 by 5 case, I is 25 by 25, and for the 7 by 7 case, I is 49 by 49.
- γ is the discount factor or rate.
- P is a transition matrix of N by N dimension, where N is the number of states in the grid. In particular, for the 5 by 5 case, P is 25 by 25, and for the 7 by 7 case, P is 49 by 49. Each entry of P is the probability of transition from state s to s' occurring. For example, the probability of a transition from the top-left cell to itself is 0.5, because there is a 0.25 probability of moving north, plus a 0.25 probability of moving west, both of which result in the agent staying in its cell.
- v is a "flattened" version of the state-value table. In other words, a state-value table of the following set-up, where each cell is marked by its row and column number as "row_column",

1_1	1_2	1_3
2_1	2_2	2_3
3_1	3_2	3_3

  would be flattened to the following as vector v:

$$\begin{pmatrix} 1\_1 \\ 1\_2 \\ 1\_3 \\ 2\_1 \\ 2\_2 \\ 2\_3 \\ 3\_1 \\ 3\_2 \\ 3\_3 \end{pmatrix}$$

  The first element of this vector corresponds to the cell found in the first row and column of the grid, and the second element corresponds to the first row and second column of the grid, and so on.
- R is a reward vector. Each entry of the reward vector is a sum of immediate rewards from the full set of actions that can occur at a state. Each of these immediate rewards is weighted by the probability of the associated action occurring. For example, R_1 (the top-left cell) is as follows,

$$\begin{aligned} R_1 = & -1 \cdot (0.25)(\text{for action “west”}) \\ & + (-1 \cdot 0.25)(\text{for action “north”}) \\ & + (0 \cdot 0.25)(\text{for action “south”}) \\ & + (0 \cdot 0.25)(\text{for action “east”}) \end{aligned}$$

  The full reward vector R is as follows,

$$\begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_N \end{pmatrix}$$

where R_1 is the reward of cell (1, 1) in the grid and R_2 is the reward of cell (1, 2) in the grid, etc.

Each component of the matrix equation $(I-\gamma P)v = R$ is produced using helper functions `prob_s_to_s_prime()` and `compute_reward_vec()` that are called in `linear_solver()`. Matrices I and P and vectors R and v are produced as follows:

- I is computed using `eye()` from `numpy`, with a specification of dimensions when calling this function.
- P is computed by looping over the number of entries of the transition matrix and, in each iteration, calling `prob_s_to_s_prime()` to set the relevant transition matrix entry with the correct probability.
- R is computed similarly to P, that is, by looping over the number of entries of the reward vector and setting each entry. `compute_reward_vec()` produces this reward vector.

1.2 Description of Policy Iteration Implementation (i.e., iterative policy evaluation component of "policy iteration")

The iterative policy evaluation component of the policy iteration algorithm is implemented in this approach. This approach is written in `A1_Q2.py` and is implemented by the class `Iterative_Policy_Evaluation()`, consisting of functions `reward_state_s_action()`, `get_ind_from_coords()`, `prob_s_to_s_prime()`, `get_grid_coords()`, and `iterative_policy_evaluation()`.

Iterative policy evaluation is a process used to evaluate the value function of a policy. The goal of this process is to estimate the expected return of following a particular policy from a given state. This algorithm is indicated in the pseudocode of page six of lecture 3.1 (`https://onq.queensu.ca/d2l/le/content/718240/viewContent/4281953/View`). The `iterative_policy_evaluation()` method executes this algorithm, with all remaining functions of the class being used as helper functions. The algorithm begins with an initial estimate of the value function, which is set to zero for all states. The algorithm then iteratively updates the estimate of the value function for each state by computing a weighted sum of the expected returns from all possible actions that can be taken from that state. The weights used in the update are determined by the probability of taking each action under the current policy.

More formally, let $V(s)$ be the current estimate of the value of state s, and let π be the current policy. Then, the update rule for $V(s)$ is given by:

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r(s, a) + \gamma \cdot V(s')]$$

In this equation,

- $\pi(a|s)$ is the probability of selecting action a in state s under the policy π. In this problem, the probability of selecting any one of the four actions is uniform, 0.25 for each direction of north, east, west, and south.

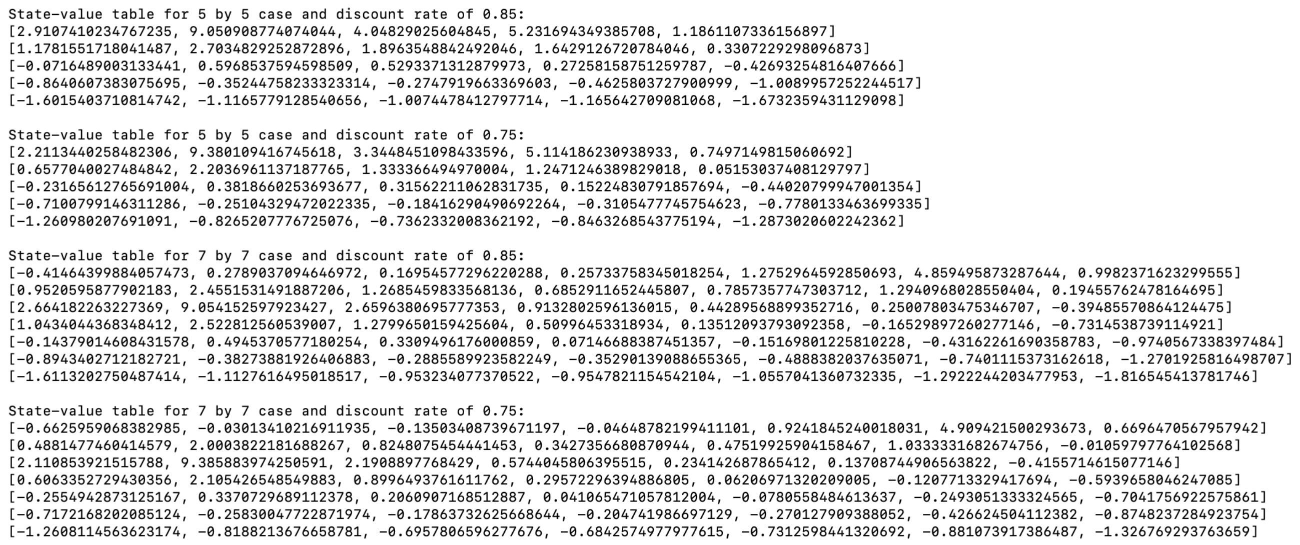
- $p(s', r|s, a)$ is the probability of moving to state s' from state s after taking action a. This value can either be 0 or 1. Please note that this value is not the same as an entry of the transition matrix from the linear solver approach.
- $r(s, a)$ is the reward expected after taking action a from state s.
- γ is the discount factor used to discount future rewards.

In essence, this equation states that the value of state s is equal to the expected immediate reward obtained from taking an action a in state s, plus the expected discounted value of the next state s' that is reached after taking that action, and then following the policy π from that state onwards. The iterative policy evaluation process continues to update the value function estimate for each state until the estimate converges to a stable solution. `reward_state_s_action()` is used to compute $R(s, a, s')$.

In the program, for each update of the state-value for a state s, `reward_state_s_action()` computes $R(s, a)$, `prob_s_to_s_prime()` computes $\pi(a|s)$ multiplied by $p(s', r|s, a)$ for the relevant term of the inner sum from the above equation. The other two helper functions are responsible for converting state numbers to grid coordinates, and vice-versa.

1.3 Results from each case, computed by the linear solver

The graphic below is the printed output of `A1_Q1.py`.



```
State-value table for 5 by 5 case and discount rate of 0.85:
[2.9107410234767235, 9.050908774074044, 4.04829025604845, 5.231694349385708, 1.1861107336156897]
[1.1781551718041487, 2.7034829252872896, 1.8963548842492046, 1.6429126720784046, 0.3307229298096873]
[-0.071648900313341, 0.5968537594598509, 0.5293371312879973, 0.27258158751259787, -0.42693254816407666]
[-0.8640607383075695, -0.35244758233323314, -0.2747919663369603, -0.4625803727900999, -1.0089957252244517]
[-1.6015403710814742, -1.1165779128540656, -1.0074478412797714, -1.165642709081068, -1.6732359431129098]

State-value table for 5 by 5 case and discount rate of 0.75:
[2.2113440258482306, 9.380109416745618, 3.3448451098433596, 5.114186230938933, 0.7497149815060692]
[0.6577040027484842, 2.2036961137187765, 1.333366494970004, 1.2471246389829018, 0.05153037408129797]
[-0.23165612765691004, 0.3818660253693677, 0.31562211062831735, 0.15224830791857694, -0.44020799947001354]
[-0.7100799146311286, -0.25104329472022335, -0.18416290490692264, -0.310547774575754623, -0.7780133463699335]
[-1.260980207691091, -0.8265207776725076, -0.7362332008362192, -0.8463268543775194, -1.2873020602242362]

State-value table for 7 by 7 case and discount rate of 0.85:
[-0.41464399884057473, 0.2789037094646972, 0.16954577296220288, 0.25733758345018254, 1.2752964592850693, 4.859495873287644, 0.9982371623299555]
[0.9520595877902183, 2.4551531491887206, 1.2685459833568136, 0.6852911652445807, 0.7857357747303712, 1.2940968028550404, 0.19455762478164695]
[2.664182263227369, 9.054152597923427, 2.6596380695777353, 0.9132802596136015, 0.44289568899352716, 0.25007803475346707, -0.39485570864124475]
[1.0434044368348412, 2.522812560539007, 1.2799650159425604, 0.50996453318934, 0.13512093793092358, -0.16529897260277146, -0.7314538739114921]
[-0.14379014608431578, 0.4945370577180254, 0.3309496176000859, 0.07146688387451357, -0.15169801225810228, -0.43162261690358783, -0.9740567338397484]
[-0.8943402712182721, -0.38273881926406883, -0.2885589923582249, -0.35290139088655365, -0.4888382037635071, -0.7401115373162618, -1.2701925816498707]
[-1.6113202750487414, -1.1127616495018517, -0.953234077370522, -0.9547821154542104, -1.0557041360732335, -1.2922244203477953, -1.816545413781746]

State-value table for 7 by 7 case and discount rate of 0.75:
[-0.6625959068382985, -0.03013410216911935, -0.13503408739671197, -0.04648782199411101, 0.9241845240018031, 4.909421500293673, 0.6696470567957942]
[0.4881477460414579, 2.0003822181688267, 0.8248075454441453, 0.3427356680870944, 0.47519925904158467, 1.0333331682674756, -0.010597977764102568]
[2.110853921515788, 9.385883974250591, 2.1908897768429, 0.5744045806395515, 0.234142687865412, 0.13708744906563822, -0.4155714615077146]
[0.6063352729430356, 2.105426548549883, 0.8996493761611762, 0.2957229639488680 5, 0.06206971320209005, -0.1207713329417694, -0.5939658046247085]
[-0.2554942873125167, 0.3370729689112378, 0.2060907168512887, 0.041065471057812004, -0.0780558484613637, -0.2493051333324565, -0.7041756922575861]
[-0.7172168202085124, -0.25830047722871974, -0.17863732625668644, -0.204741986697129, -0.270127909388052, -0.42662450411 2382, -0.8748237284923754]
[-1.2608114563623174, -0.8188213676658781, -0.6957806596277676, -0.6842574977977615, -0.7312598441320692, -0.881073917386487, -1.326769293763659]
```

1.4 Results from each case, computed by iterative policy evaluation

The graphic below is the printed output of `A1_Q2.py`.

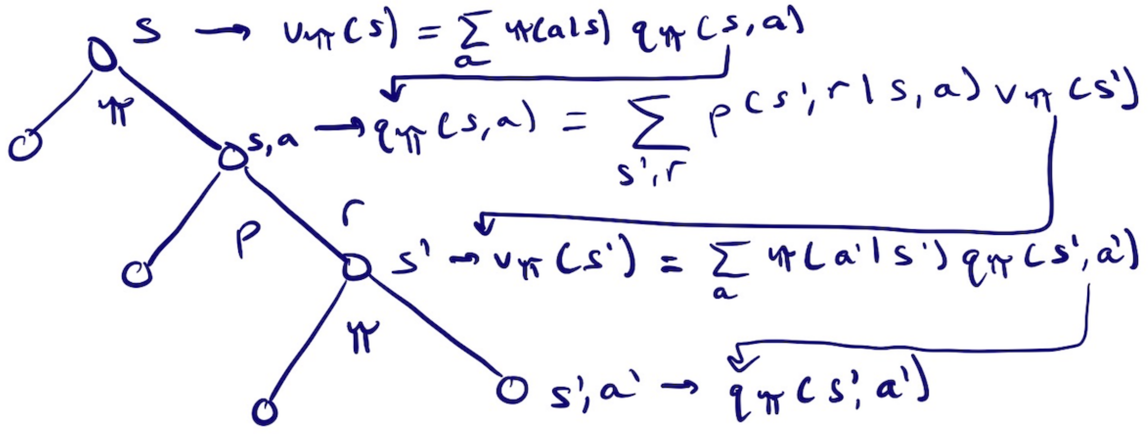
```
State-value table for 5 by 5 case and discount rate of 0.85:
[2.911045619696575, 9.051120792757843, 4.048490748719934, 5.231867789402914, 1.1862984429098016]
[1.1784228532668362, 2.703691076376124, 1.896535357481591, 1.643076736445121, 0.330885055332499]
[-0.07139937399186844, 0.5970524513312283, 0.5295070005530795, 0.2727361473847473, -0.42678334449073896]
[-0.8638205658863338, -0.3522553823238725, -0.27462809590353543, -0.46243166435142147, -1.0088531084397412]
[-1.6013040851550588, -1.116388647361541, -1.00728658254044, -1.165496553003889, -1.673096022365743]

State-value table for 5 by 5 case and discount rate of 0.75:
[2.211458937342739, 9.380173940577647, 3.3449048635886798, 5.114234490449822, 0.749771411306259]
[0.6577958325566791, 2.2037574794429724, 1.3334160375669808, 1.247168269202168, 0.05157398840100649]
[-0.23157377179466854, 0.38192297989008256, 0.31566713845559513, 0.1522880361286394, -0.44016978900554216]
[-0.7100015347414559, -0.2509887938202807, -0.18411999590986844, -0.3105100605886296, -0.7779774258095421]
[-1.2609032237428932, -0.8264672251513676, -0.7361910847573496, -0.8462898982447244, -1.2872669426335142]

State-value table for 7 by 7 case and discount rate of 0.85:
[-0.4143124473086769, 0.2791737572902039, 0.1697743665175548, 0.25753585486079167, 1.2754677321109107, 4.859634711833534, 0.9983940049951003]
[0.9523387744587894, 2.4553781966051083, 1.2687359328042085, 0.6854570785566126, 0.7858840368574695, 1.2942326250299774, 0.19469408438136196]
[2.664432045107455, 9.054349918893696, 2.659806336091665, 0.9134287946655371, 0.4430312973451376 3, 0.2502060962438032, -0.39472973148272905]
[1.0436384763439193, 2.5229990234203727, 1.2801224245523382, 0.5101039046537938, 0.13524938757882482, -0.16517651585516974, -0.7313337253675343]
[-0.14356446847819582, 0.4947170570051136, 0.33110136718859084, 0.07160148904061013, -0.151573448703605, -0.43150343627556986, -0.9739397570681269]
[-0.8941187278078524, -0.3825621481008361, -0.28841004195058484, -0.352769127335234, -0.4887155894073276, -0.7399940449993383, -1.2700772054026697]
[-1.6111003746096006, -1.112586321937712, -0.9530862442892543, -0.9546507842022177, -1.0555823057309206, -1.2921076149211943, -1.8164306851314462]

State-value table for 7 by 7 case and discount rate of 0.75:
[-0.6624965988660468, -0.030045825989267805, -0.13495222256916742, -0.046409318016767176, 0.9242585113413203, 4.909484783147824, 0.6697221339082885]
[0.4882514063355039, 2.000466515673434, 0.8248771389620302, 0.34279849166622206, 0.4752585423836482, 1.0333900066256672, -0.010539353344814276]
[2.1109673285585164, 9.385977753951439, 2.190956862210851, 0.5744622785107284, 0.2341968090352775, 0.1371401871167226, -0.415518710832407]
[0.6064461599571676, 2.1055075312989238, 0.8997130526312226, 0.2957788120705709, 0.06212235273874303, -0.12071989382761168, -0.5939146681691901]
[-0.25538452072137674, 0.3371511012402308, 0.20615329221050654, 0.04112096854012264, -0.07800331218384747, -0.24925372739426815, -0.7041246370396185]
[-0.717107008013459, -0.2582227892131712, -0.1785748850381725, -0.2046863813698787, -0.27007515848726205, -0.426572849766713, -0.8747724274271831]
[-1.2607014748215744, -0.8187436863618079, -0.6957181840760411, -0.684201796532778, -0.7312069630124471, -0.8810221199429892, -1.3267178476658739]
```

2 State-value and Action-value Functions



$s \rightarrow v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$
π
$s,a \rightarrow q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\, v_\pi(s')$
p
r
$s' \rightarrow v_\pi(s') = \sum_a \pi(a'|s')\, q_\pi(s',a')$
π
$s',a' \rightarrow q_\pi(s',a')$

2.1 Write v_π in terms of q_π

$v_\pi(s) = E_{a\sim\pi}[q_\pi(s,a)|S_t = s, A_t = a] = \sum_a \pi(a|s) q_\pi(s,a)$

2.2 Write in q_π terms of v_π

$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) v_\pi(s')$