CS885 - Assignment 1

Connor Raymond Stewart - 20673233

September 2021

1 Introduction

The report herein is a submission for assignment one in CS885 at the University of Waterloo, as it was taught over the Fall term of 2021 by Pascal Poupart.

2 Part I

The results of the testMDP.py code is as follows:

—valueIteration
V = [31.49636306 38.51527513 43.935435 54.1128575]
—extractPolicy
Policy = [0 1 1 1]
—evaluatePolicy
V = [-5.74175948e-16 -0.00000000e+00 1.81818182e+01 1.00000000e+01]
—policyIteration
V = [31.58510431 38.60401638 44.02417625 54.20159875]
Policy = [0 1 1 1]
—evaluatePolicyPartially
V = [0. 0.08727964 18.18181818 10.08727964]
—modifiedPolicyIteration
Policy = [0 1 1 1]
V = [31.50523718 38.52414925 43.94430913 54.12173163]

The results of the value iteration function are as follows:

V = [60.62388836 66.03486523 71.80422632 77.09196339 59.81429704 65.18237783
77.83066489 84.14118981 58.09361039 7.98780239 84.86704922 91.78159355
69.49584217
76.80962081 91.78159355 100. 0.]
nIterations = 20
epsilon = 0.008079508521518619
Using the extractPolicy function, we obtain: [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]

The results of the policy iteration function are as follows:

policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.63256172 66.03897428 71.8062328 77.09295576 59.81945165
65.18457679
77.83151901 84.14149059 58.0955782 7.98862928 84.86730581 91.78165089
69.4968138
76.80991653 91.78165089 100. 0.]
nIterations = 5

The results of the modified policy iteration are as follows. Note that each successive set of results below is for a partial policy evaluation with the number of iterations varying from 1 to 10. The tolerance value is fixed at 0.01, and we start with the policy that chooses action 0 in all states and start with the value function that assigns 0 to all states.

policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62876104 66.03718127 71.80535493 77.09252307 59.81720318 65.18361291
77.83114699 84.14135891 58.0947136 7.98826957 84.86719334 91.78162593
69.4963892
76.80978687 91.78162593 100. 0.]
nIterations = 6
epsilon = 0.003528613005464365

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62388836 66.03486523 71.80422632 77.09196339 59.81429704 65.18237783
77.83066489 84.14118981 58.09361039 7.98780239 84.86704922 91.78159355
69.49584217
76.80962081 91.78159355 100. 0.]
nIterations = 19
epsilon = 0.008079508521539935

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62958804 66.03755603 71.8055451 4 77.09261306 59.81766514 65.18382585
77.83122335 84.14138772 58.09490586 7.98834194 84.86721832 91.78163102
69.49647811
76.8098156 91.78163102 100. 0.]
nIterations = 11
epsilon = 0.002815779877039404

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.6290706 66.03732148 71.8054255 77.09255667 59.81737843
65.18369179
77.83117557 84.14136957 58.09478741 7.98829682 84.86720256 91.78162783
69.49642341
76.80979751 91.78162783 100. 0.]
nIterations = 8
epsilon = 0.003252262983522769

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62934805 66.03745592 71.80548979 77.09258923 59.81754974
65.1837614
77.8312038 84.14137915 58.09485314 7.98832445 84.86721065 91.78162974
69.49645648
76.80980686 91.78162974 100. 0.]
nIterations = 7
epsilon = 0.0029823627902345606

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62876104 66.03718127 71.80535493 77.09252307 59.81720318 65.18361291
77.83114699 84.14135891 58.0947136 7.98826957 84.86719334 91.78162593
69.4963892 76.80978687 91.78162593 100. 0.]
nIterations = 6
epsilon = 0.003528613005464365

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62684779 66.0363074 71.80491664 77.09231201 59.81607965
65.18312711
77.83096688 84.14129264 58.09415333 7.98809797 84.86713622 91.78161387
69.49613862
76.8097214 91.78161387 100. 0.]
nIterations = 5
epsilon = 0.005285946952227505

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62899149 66.03727177 71.8054104 77.09254469 59.81729516
65.18367944
77.83116404 84.14136775 58.09476487 7.9882834 84.86720147 91.78162704
69.49640699
76.80979588 91.78162704 100. 0.]
nIterations = 6
epsilon = 0.0034316310956086227

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62687175 66.03625199 71.80492934 77.09229895 59.81597385 65.18315886
77.83095043 84.14129651 58.09428558 7.9880721 84.8671415 91.78161268
69.49616014
76.80972578 91.78161268 100. 0.]
nIterations = 5
epsilon = 0.005632742934892576

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.62757676 66.0366112 71.80509113 77.09238611 59.8164457
65.1833289
77.83102736 84.14132014 58.09443814 7.98814971 84.86716094 91.78161789
69.4962458
76.80974848 91.78161789 100. 0.]
nIterations = 5
epsilon = 0.004845384255254714

Policy = [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
V = [60.63178095 66.03865552 71.80607955 77.09287867 59.81897368
65.1844102
77.83145231 84.14146772 58.09542203 7.98856416 84.86728644 91.7816464
69.49673581
76.80989415 91.7816464 100. 0.]
nIterations = 5
epsilon = 0.0012871115868122729

3 Part II

As a test trial, TestRLmaze.py is run on a Q-learning set for epsilon values ranging from 0.05, 0.1, 0.3, and 0.5. see the attached python document for the specific testing cases. The results are as follows:

Epsilon 0.05
Q = [[74.93049922 80.44149302 79.06996692 88.66703058 75.39307038
76.98671122 73.11010132 89.41498541 80.83008322 15.95352353
90.7254751 93.18612834 82.92067734 63.0104086 95.50494899
110.26124142 6.22153536]
[81.01420788 85.5464041 76.40905897 95.8893351 82.8848246
24.27820223 98.53908526 97.77311071 77.51421872 28.06371985
99.90178017 103.17843462 81.64166912 82.16307005 106.7744575
107.14699442 6.30839849]
[74.48396847 80.14642724 77.67765386 75.33180922 78.97999579
78.65875666 88.39568136 102.09567691 82.65057016 16.10071226
23.11044993 95.12532302 82.23318629 83.323241 94.40494694
105.07649488 6.20857708]
[77.06164499 78.21204113 81.2413452 81.71534574 83.58637263
91.91538757 94.30836309 90.60618085 80.49211294 28.25553652
97.38832307 96.72322123 87.84982475 100.1779349 103.54954589
109.717156 6.52530343]]
Policy = [1 1 3 1 3 3 1 2 2 3 1 1 3 3 1 0 3]

Epsilon 0.1
Q = [[73.8708296 75.80694712 78.1471264 78.26684553 73.0343897
80.52565573 89.76331506 76.88904883 72.56509678 6.73894061
87.44013902 81.57200292 73.42623424 19.25147525 92.71908269
108.48900108 9.09116131]
[75.55385977 75.79760546 80.00739491 78.35147437 77.78244582
21.3977792 91.51736501 90.52722234 77.68937617 19.67245708
22.02884204 103.270439 83.71004884 84.67390956 96.90336994
105.68699574 8.68842244]
[72.69946791 69.77883798 73.9593046 77.88375723 73.55630157
72.96135742 78.10193857 83.4814486 76.85294916 4.09233459
45.98783333 93.77428596 80.0381702 79.83597601 91.94421391
107.44867153 8.64302245]
[75.28326438 79.4674759 79.60688821 77.7818774 74.51893651
84.87692329 83.53742524 80.55706594 22.65515293 24.09800527
96.87150299 98.20339946 86.46887209 92.09203046 102.61039142
108.72434354 8.34929021]]

Policy = [1 3 1 1 1 3 1 1 1 3 3 1 3 3 3 3 0]

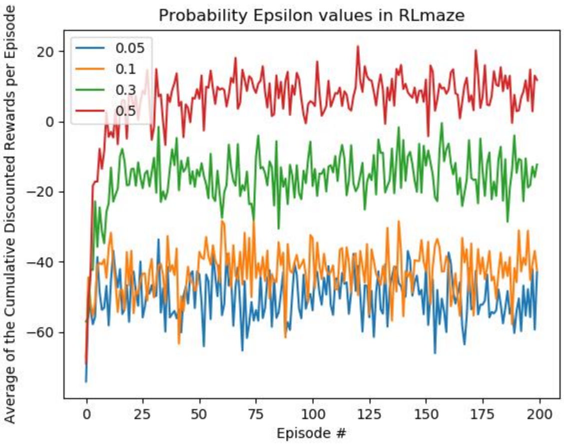
Epsilon 0.3

```
Q = [[ 56.01588423  61.77614738  69.5728815   75.69493484  58.47290407
   61.70198135  66.86478913  71.91169326  60.97857488   6.34277018
   84.95577555  87.43963785  66.29662775  21.61159715  87.33496631
  110.10273909   9.2703664 ]
 [ 60.51442543  80.81151567  72.3392583   69.42019438  63.63392653
   18.27816353  91.11602629  96.23816414  68.31104946  17.49584054
   20.10547684  99.29496862  68.47565589  68.10922996  89.42018514
  107.26808975   9.4923582 ]
 [ 57.33022566  59.52169544  70.01411028  73.9050578   57.81822792
   62.05570334  74.60253891  72.8392298   65.52405556  -4.11583081
   20.63190567  90.91924765  67.72650482  63.76378329  75.32367664
  107.06555443   9.54815065]
 [ 58.53170048  62.27961837  71.68138486  71.77793755  60.30876725
   88.04186225  89.86211542  70.67128743  18.81035521  22.57190091
   95.27757599  96.24382125  67.69033074  94.87987501 102.70707198
  108.38092503   9.52641974]]
Policy = [1 1 1 0 1 3 1 1 1 3 3 1 1 3 3 0 2]
```

Epsilon 0.5

```
Q = [[ 67.97735433  68.44616247  71.50440119  73.27099763  68.96935927
   70.60965733  68.66233889  73.42568788  56.82548683  16.15982854
   79.89234781  87.27553166  68.26694712  17.6935929   84.55783034
  105.98556772   6.58638325]
 [ 60.71513778  64.94023833  73.79400068  75.84053128  58.88888217
   20.47370978  77.95288116  94.04713656   9.56536826  14.88643472
   57.26536251  88.52210487  67.18787851  87.45373171  95.11790017
  105.97421742   6.43195694]
 [ 65.07766084  67.59472577  67.04965935  71.92036578  56.76517512
   67.87584489  77.4056067   76.00741221  60.75669371  -7.30289711
   17.94219408  77.24839941  69.32868501  17.22003041  87.38448997
  105.85484464   6.38093797]
 [ 65.70509129  69.31579828  69.42573294  70.91890239  58.68771381
   17.80878254  75.70342026  77.95048826  21.24270223   7.96600838
   88.05028681  85.87840603  87.41975053  83.98504937 100.75652728
  104.25989246   6.4006658 ]]
Policy = [0 3 1 1 0 0 1 1 2 0 3 1 3 1 3 0 0]
```

For each of the epsilon values above, the code is then run 100 times and the results are averaged and are then plotted onto a matplotlib window. See below:



The average cumulative discounted rewards per episode earned during training tends to approach zero as the epsilon value increases. As the epsilon value increases to the 0.3 to 0.5 range, there is a greater chance that the algorithm will skip over optimal value ranges and fail to learn how to maximize average reward. If the epsilon is too small, there is a chance that the values will fail to converge to a optimal maximum reward state. According to the results above, the optimal epsilon can be either 0.5 or 0.3, but leans more to 0.5 being optimal.

As the epsilon value increases, the resulting Q-values also increase on average. This can be made evident by averaging the mean Q-values over the one hundred test samples. See below:

Epsilon 0.05:
```
[[ 77.14456852  83.05173883  89.07301554  91.27553654  82.98492798
   88.18767116  95.99461986  93.77916058  83.35603481  28.07655306
  102.11600192 101.1100032   99.85711186 100.5101609  106.3996324
  113.54415492  14.85876917]
 [ 80.44136682  87.01831823  92.92468784  96.00041356  85.64526203
   85.53189066  97.58995462 101.9306878   90.94109935  29.0241512
  104.28972297 108.98532029  95.62082193 103.56235822 110.03866071
  114.01309877  15.18409131]
 [ 78.83793704  84.07127608  89.84267416  93.57889184  83.63482927
   80.7085288   92.97875826  99.08028072  86.72502408  26.58377628
  103.28313818 107.39337906  91.67750565  95.02051522  99.39833566
  112.72675424  15.0364902 ]
 [ 81.11227629  86.98524489  91.28418427  95.38350681  86.34938246
   91.94636271  95.30934802 100.98792658  89.22191139  29.91662566
  103.50181054 104.56500565  97.26203982 104.86493318 109.76311061
  115.01025714  14.75855769]]
```

Epsilon 0.1:
```
[[ 80.4409924   83.11957393  86.92245492  89.40591916  82.62582839
   86.37375497  93.14651562  91.02138627  85.50190445  24.5646394
  101.02563385  98.01492111  92.87147512 100.24821254 106.67862119
  112.24945766  14.20548804]
 [ 80.1726084   85.441402    90.21710767  96.90486719  86.11366608
   90.22787588  95.7107819  102.22165609  90.67650392  27.20107876
```

102.10776508 107.58234985 93.26589697 98.92359531 104.95421413
112.38568289 14.1861652]
[78.6720925 84.27450227 88.18732932 93.26837489 83.0747446
83.90160037 94.33660281 98.02727077 88.10856819 26.2155343
99.25240501 107.52731422 89.78082247 92.14542545 97.88569487
113.58072211 14.25953031]
[81.23999901 85.21479023 89.76361722 93.38174639 84.97457916
89.2281453 93.96156535 100.96495317 87.21221165 27.23468243
102.15401585 105.11518636 95.60014499 100.47213511 107.84760694
113.7526852 14.31475847]]

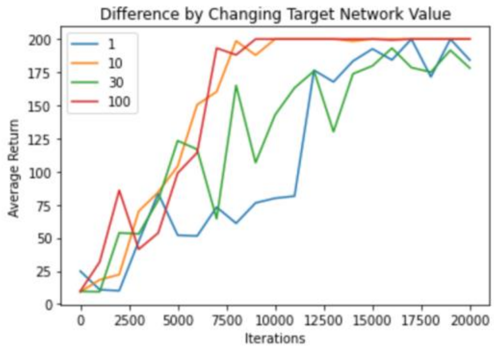
Epsilon 0.3
[[75.0170197 79.52711043 83.87490878 87.72060632 81.23989875
87.08498162 88.95126562 88.75138154 79.9126467 23.22274638
96.50377129 97.90643393 93.25572187 96.77832655 103.72672247
111.21971074 12.54217395]
[78.8863493 82.66617262 89.27009293 96.4333285 83.08092068
88.69493256 96.27309379 102.29194686 86.60687313 24.28624341
100.02026612 106.59449694 91.1721734 96.35691041 104.70934682
111.76394117 12.54769524]
[74.34229081 76.34124407 88.84143924 93.14186311 78.3407818
79.29087685 94.652757 99.7915688 84.90939095 21.0930544
96.09519019 106.65864268 87.53228463 93.44504299 95.65843612
113.69044979 12.75468653]
[77.76198913 83.8250273 87.86486201 92.5201554 84.48639307
89.72482733 96.08304148 96.5385557 85.23910238 26.27713028
101.22034347 103.56596807 93.8114623 99.2494766 105.75492295
110.29223649 12.88317993]]

Epsilon 0.5
[[73.74414235 80.19548661 82.21300016 85.43699313 74.97105787
82.79323166 90.35525226 92.10820009 74.41563284 24.39524786
90.82148921 96.59571203 89.53487715 96.76137722 104.50798968
111.13929312 10.57624101]
[77.51736573 81.33904622 87.38185387 94.41864977 79.75554554
85.29429877 94.38292225 97.83113254 86.30117253 23.25737416
98.29866879 104.9430415 87.95611505 98.32169846 103.87904945
110.06059735 10.34127928]
[74.81378738 81.23982721 79.94573757 89.91405112 76.01049146
77.96684705 95.42769796 96.04657046 85.54367483 20.95544877
95.15756672 101.45503944 84.64689672 87.57034217 95.01256468
110.17442036 10.53270788]
[78.83151225 80.94839959 88.97236261 89.49939527 83.83027101
87.9670727 92.77749168 96.16441282 83.12769891 24.03798194
99.08669953 102.88658552 92.78803161 98.34614744 103.41990269
110.60567305 10.58307353]]

As can be seen, the Q values peak on average for epsilons 0.3 and 0.5. if the epsilon value is very large, the system won't be able to converge on a solution easily, meaning the policy values likely fluctuate between samples. If the epsilon is too small, then the system will not be able to find optimal solutions as the values will be too constant. A small epsilon would result in the policy not changing very much over many successive sample trials of the program.

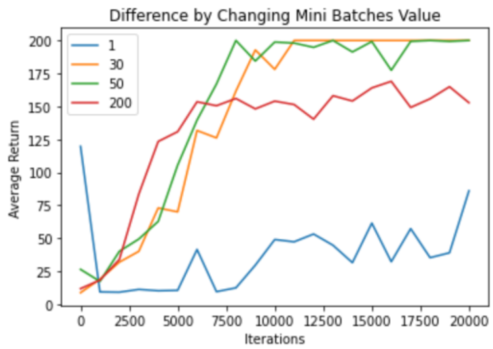
4 Part III

Target Network Value:



As can be seen, the target network value is closely related to convergence speed. Reducing the target network value while updating the weights in the Q-network can allow for a faster convergence at the expense of reduced resolution for the reinforcement learning. The temporal difference model cannot calculate the model accurately when using a target network value which is too low, which causes the network to converge incorrectly or diverge completely. Having an excessive target network value on the other hand can cause the program to slow down too much and consume too much memory.

Mini Batches Value:



The overall variation in rewards increases as we lower the mini-batch size. Small batches are more granular in nature, meaning slight changes in quantities in the batches can cause sudden spikes or depressions in the overall graph. The line is averaged and smoothed out with larger batch sizes. Convergence is slower with smaller batch sizes because the system can only retain so much information from samples. However, larger batch sizes can cause the system to over generalize data and gloss over important details.