CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

I. Abstract

This assignment explores Markov's Decision Processes (MDPs). Two problems are analyzed including a small grid-world problem and a large non-grid world problem. Each MDP is solved using a combination of Value Iteration, Policy Iteration and Q-learning (Reinforcement Learning).

II. MDP Problems

A. Frozen Lake

The frozen lake problem represents a very simple problem. An agent can move around the grid world consisting of 4 types of grid cells. Start, Goal, Frozen tile, and Hole. The goal of the agent is to move from Start to Goal without falling into a Hole. The interesting characteristic of the frozen surface tile is that the surface can be slippery or non-slippery. In a slippery surface, an agent can be commanded to move in one direction, but can actually end up moving in another, due to the slippery nature of the surface. The grid-world problem represents real world autonomous vehicle problem space and, more so, adds a interesting "slippery" property to learn with. In order to better characterize

B. Gambler's problem

The Gambler's problem is another classic problem where a gambler bets on the outcome of a coin flip. The Gambler either wins the same amount as his bet, or loses the bet. The game stops at any number desired, but is default at 100 dollars or loss of money (0 dollars). Due to the variable game stop condition, the problem could possibly contain ∞ states. The interesting part of this problem is the unfair coin which can be varied in the probability of landing "heads".

III. Frozen Lake Problem

A. Environment Setup

The Frozen Lake environment used was imported from the gym library for Python. The gym library is powerful in the sense of allowing customized grid sizes and hole/frozen tile ratio in order to customize, analyze and simulate different environments. For this assignment, only one grid problem is required, but four will be analyzed in order to exhaust the exploration of algorithms and the effects of these parameters on the outcomes. Namely, a 4x4 Grid with 0.5 ratio and 0.8 ratio as well as a 8x8 grid with 0.5 ratio and 0.8 ratio will be studied.

B. Value Iteration

Value Iteration uses the Bellman equation to evaluate all states of a problem until a convergence of a solution is reached. By starting at the end, the utility of each adjoining step can be calculated on discount as the iterated. The utility of each calculated state is the discounted future reward added to the immediate reward.

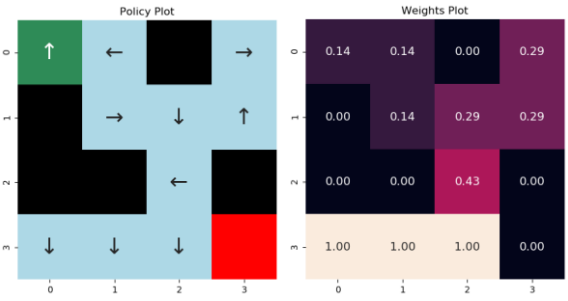


Figure 1: 4x4-50% Frozen

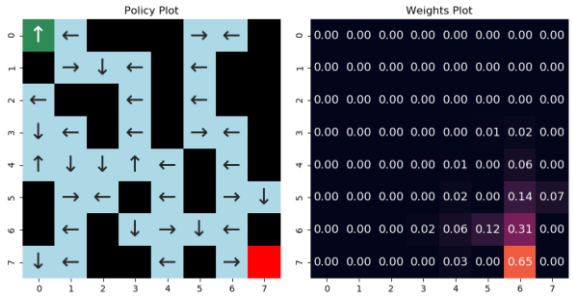


Figure 2: 8x8-50% Frozen

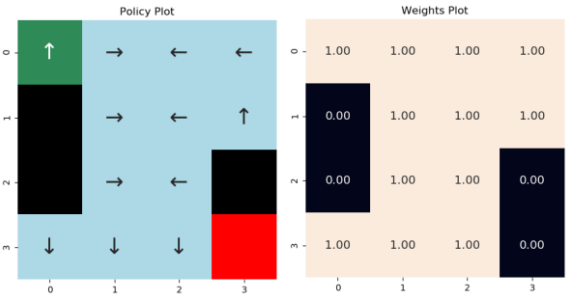


Figure 3:4x4-80% Frozen

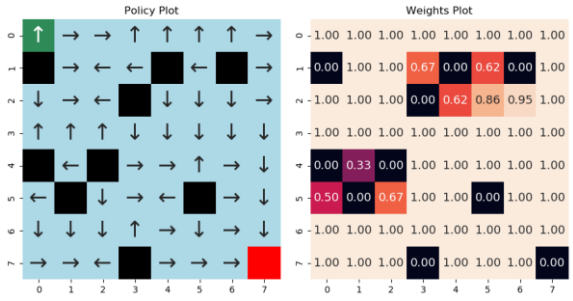


Figure 4: 8x8-80% Frozen

First, it is important to notice the directional arrows of the policy in Figures 1-4. Initial analysis may lead to a conclusion that the graphing or evaluation algorithms are broken, since the arrows do not always make sense. However, a recall of the "frozen" aspect of this problem leads to the understanding that the agent may not always be able to follow the guided direction.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

Due to the frozen nature of the Frozen-Lake grid-world, a command to move in a specific direction may not necessarily be executed in the direction requested. Furthermore, the value plots for the 80% grids show much more confidence in

To fully analyze Value Iteration, the gamma (γ) value was varied by values of γ =1.0, γ=0.7, and γ =0.4. Additionally, a new variable theta (θ) is varied in values θ=1e-7, θ=1e-5, and θ=1e-3. The gamma value is the discount factor, or a way of viewing future reward. A value of 0 values only immediate reward whereas a value of 1 values only the future rewards. The theta value, serves as the tolerance at which point convergence is considered achieved.

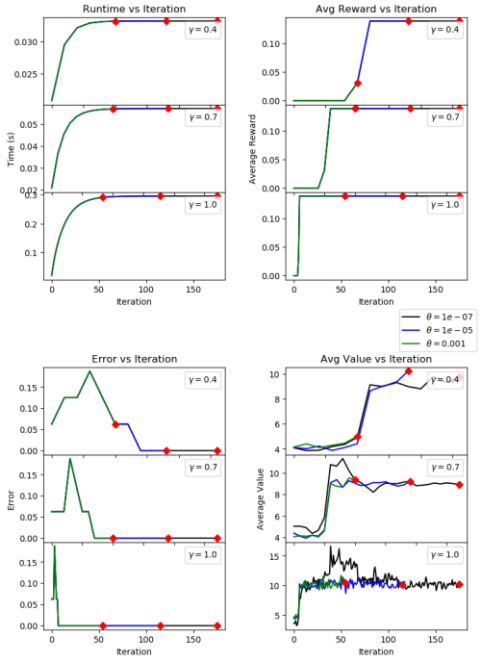


Figure 5: 4x4-50% VI Test

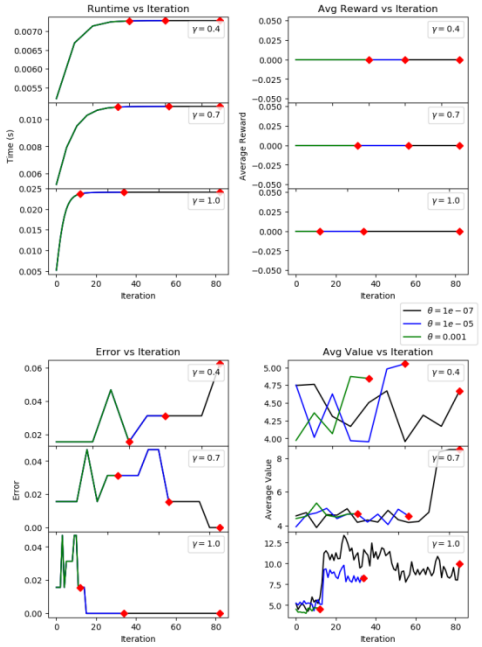


Figure 6: 8x8-50% VI Test

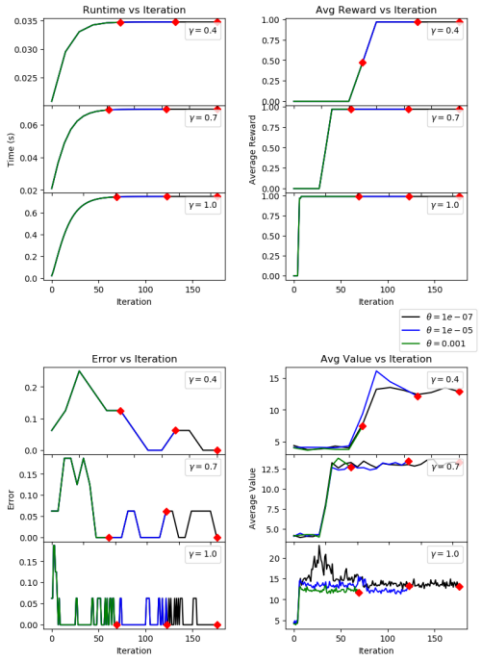


Figure 7: 4x4-80% VI Test

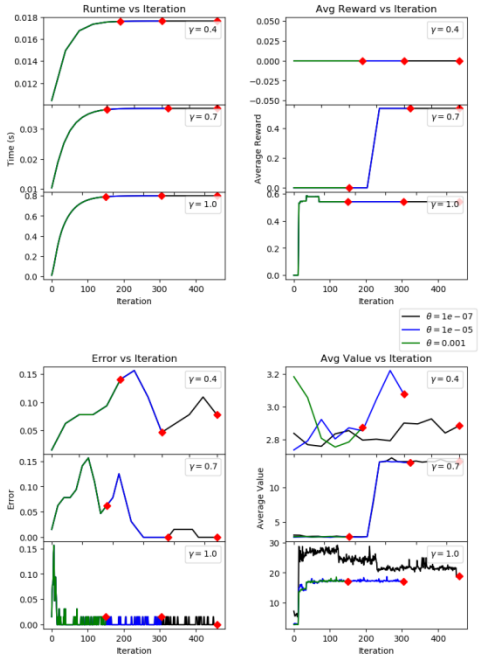


Figure 8: 8x8-80% VI test

Time Analysis - The figures 5-8 above have Iteration on all X-axes, the values are in “x100” units, where the maximum is 20,000 iterations. The y-axes units are all normalized to the average per 1000 calculations. One common trend to notice is that the θ value does not change the results for time, reward and error graphs, where a smaller θ simply takes longer to converge, but

following a similar path. Logically it follows that a larger 8x8 grid-world take longer to converge than a 4x4 grid. From a time analysis, we see that larger γ values lead to longer computation time. This is logical since preferring future rewards implies calculating and taking into account future actions, expending more computes times. If short-term gains are preferred, the algorithms can simply look at its closest neighbors to calculate reward without more complex computations and accounting for the variance in long-term reward evaluations.

Reward Analysis – Both 4x4 grid experiments with 50% froze and 80% frozen display an interesting behavior. The higher tha gamma value, the quicker the plateau of reward gains. This is an interesting observation considering that a high gamma value evaluates future rewards. It may be prudent to conclude that the policy's favoring immediate gains too often fell into Holes in the grid. Favoring a future outlook served well for the purpose of evaluating how to avoid falling in a Hole, and getting to the Goal more efficiently. Furthermore, an analysis of the 8x8 gird-world for both 50% and 80% frozen, notice that the reward at γ=0.4 remains flat at 0 for both. This appears to imply that with the favoring of immediate rewards, even with an 80% frozen world, there was not a instance of the agent reaching the goal. As γ value is increased, the 8x8 grid-world fares better only when the grid is 80% frozen. The 50% frozen grid remains with a 0 reward state, regardless of γ value, whereas the 80% frozen grid reaches the goal in a few states, with the γ=1.0 value converging in fewer iterations. For further testing, an increase to 1,000,000 iterations would be an interesting test to determine whether the agent can ever reach the goal in those scenarios where reward remained 0.

Error Analysis – In almost all instances the error plot displays a sharp initial peak with a descent of error value until reaching almost 0. This is in line with the logic that as more iteration accumulate, the agent should be better at predicting and the resulting values should look more and more alike. One outlier to this is the 8x8-50% with γ=0.4 where the error is observed trending upwards. This is most likely due to iterations count. It is evident that with lower γ value, the peak error occurs at later and later iterations. Thus what si observed in this particular graph is the trend to the peak. The actual peak and fall is not present since more iterations are required to evaluate it. Again, a suggestion of running for 1,000,000 iterations would show this to be true.

Value Analysis – The graphs of cumulative average values vs iterations display well how the θ value effects how early a "convergence" is determined. The higher the θ value, the earlier the run terminates. However observe that most instances, the ending Value at which point convergence is determined is very similar across all values of θ.

C. Policy Iteration

Policy Iteration produces arbitrary policies and calculating the effectiveness of that policy. The iteration occurs until a convergence is reached whereby new policies do not improve the score.

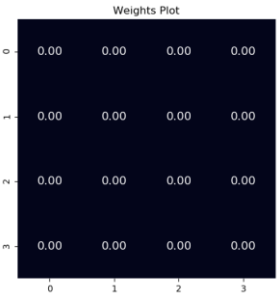


Figure 9:4x4-50%

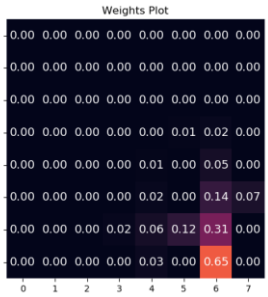


Figure 10: 8x8-50%

Figure 11: 4x4-80%

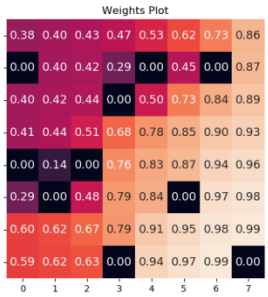


Figure 12: 8x8-80%

Observing the weights plot of the best policies shows a very interesting outlier. The 4x4-50% plot has no values anywhere on the plot, the entire grid is black with 0.00 values. Furthermore it is interesting to observe the weights at the 80% plots, with the 8x8 world displaying a very gradual gradation of weights leading towards the Goal.

To fully analyze Policy Iteration, the same test as Value Iteration was performed. Namely, the gamma (γ) value was varied by values of γ =1.0, γ=0.7, and γ =0.4. Additionally, a new variable theta (θ) is varied in values θ=1e-7, θ=1e-5, and θ=1e-3. The gamma value is the discount factor, or a way of viewing future reward whereas the theta value, serves as the tolerance at which point convergence is considered achieved.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

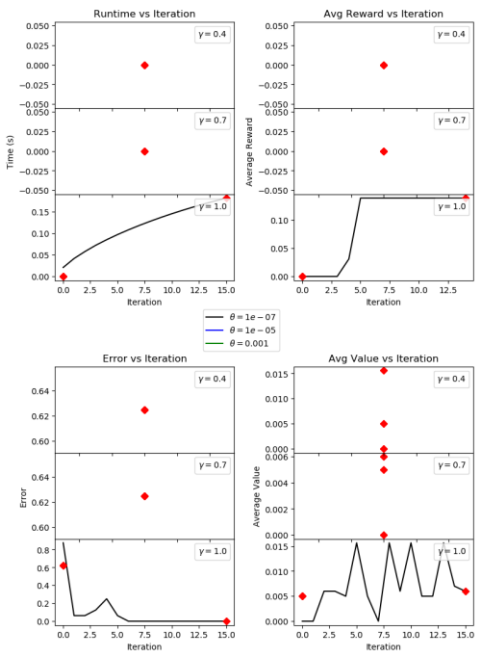


Figure 13:4x4-50% PI Test

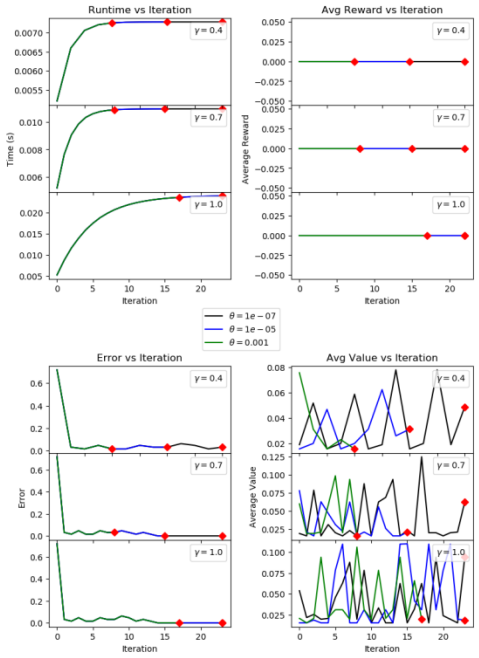


Figure 14: 8x8-50% VI Test

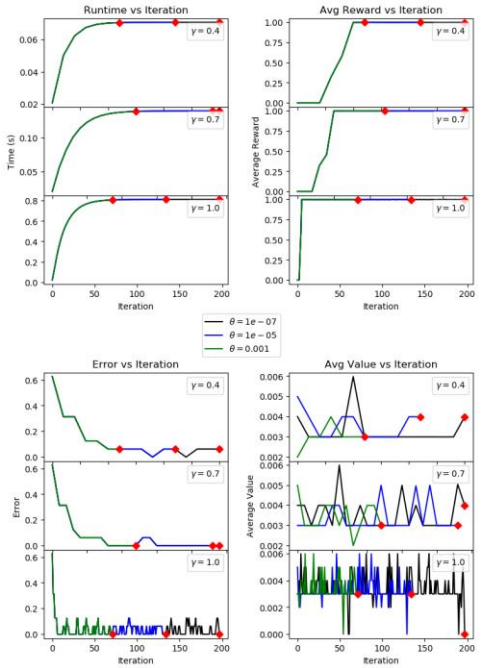


Figure 15:4x4-80% PI Test

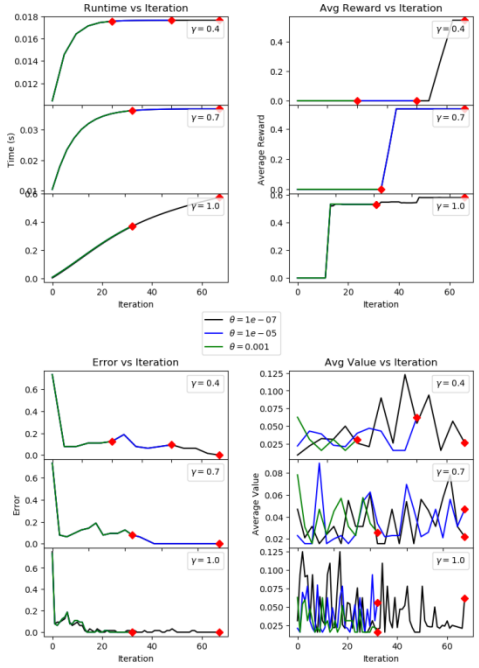


Figure 16: 8x8-80% PI Test

Time Analysis – The most prominent observation from this test is the failure of the 4x4-50% grid. It is not evident where or why the failure happened in the algorithm, but "convergence" was evaluated to be true on the very first iteration for all but the last γ=1.0 and θ=1e-07 test. This observation may imply that the observed cumulative value was close to, or at 0, leading the algorithm to evaluate the tolerance test as passed. Even in the observed case, convergence was reached in 15 iterations. Observing other grid-worlds, it is interesting to notice that both 8x8 world runtimes follow a linear curve for high gamma and a parabolic curve for low gamma. However, the 4x4-80% grid-world Figure 15, all gamma value tests follow a parabolic execution time with increased iterations. Additionally, all theta values converge around a similar iteration for all gamma values, on the 4x4-80% grid. The most logical explanation for this observation is the size of the grid-worlds. A small 4x4 grid with 80% surface frozen leaves only 3-4 holes to fall into. Thus the discount factor appears to have little effect, at least for run-time.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)
Sergiy Palguyev (gtid: spalguyev3)

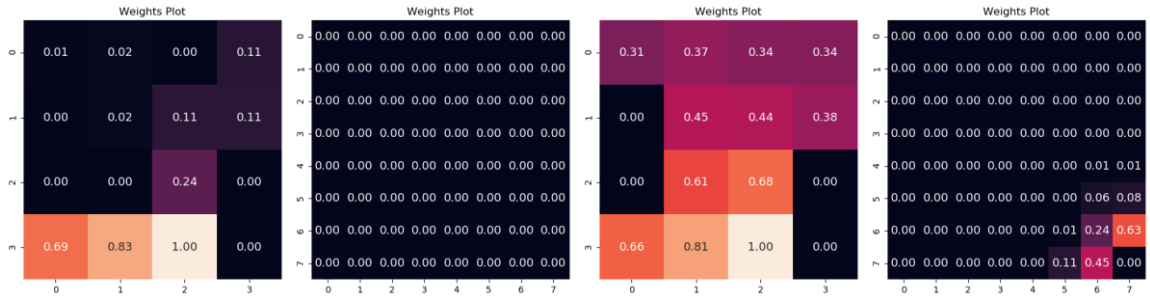
Reward Analysis – Starting with Figure 15, again it is evident that the maximum reward is reached early for γ=1.0. However, the difference between γ=0.7 and γ=0.4 is not as many iterations as one would assume. Again, this is more likely due to the small state-space being observed. Curiously, the Figure 14, 8x8-50% grid-world maintained 0 rewards for all gamma values. This reflects exactly the result shown in Figure 6 VI test of the same grid-world. It is evident aht having too many hole in the grid produces a difficult time for either value or Policy iteration to produce a good result. It is feasible that a higher number of iterations, say 1,000,000 may provide successful runs, which is good experiment to evaluate for future work.

Error Analysis – All error graphs quickly converge to a low error with higher gamma values producing a steeper slope than lower ones. It is interesting to note there are certain peaks and valleys in the error plots. This is most likely due to the future looking gamma factor which does not account for immediate reward as much as a low gamma would.

Value Analysis – One evident feature between Figure 15 and 16, best viewed at the Value analysis, is the fact that the 8x8 grid-world converges with less iterations than the 4x4 grid-world. A conclusion from the time-analysis above initially state that the 4x4 grid-world should be easier to traverse and, thus, quicker to analyze due to small number of holes in the grid-world. However, observing the number of iterations it took to converge, it appears the opposite is true. This paradox leads back to the definition of convergence. For this experiment, the θ tolerance defined at what point convergence would occur. Considering the tolerance is evaluated at the average values difference, it may be the case that with more grids, the difference between policies, when averaged leads to a quicker convergence than with a 4x4 grid-world.

D. Q-Learning

This algorithm does not depend on a model and is a form of Regression Learning. As an agent visits each state, it assigns new values based on an immediate/delayed rewards ratio. Eventually the model converges to the correct values.



The analysis of Q-Learning algorithm applied to the grid-world problem varied the most number of hyper-parameters in order to analyse the entire problem space. First and foremost all variable that can decay with number of episodes, do. Alpha (α) learning rate decays from 0.3 to 0.01 with decay rate of 0.0001. Gamma (γ) discount factor decays from 1.0 to 0.8 with decay rate of 0.0001. Both of these values are set to decay since as the learner gains experience and moves towards convergence, less weight should be put on future rewards and less weight should be put on newly learned information.

The exploration of Q-Learning is conducted based on the Action choice type and it's varied parameters. ε-greedy action strategy greedily selects the action with the greatest reward and is varied by decaying from eMax = [1.0, 0.9, 0.8] to eMin = [0.5, 0.4, 0.3] with decay rate of eDecay=[0.001, 0.0001, 0.00001] in all combinations. This action strategy presents a exploitative method whereby the current knowledge of the world is valued most and the ε value adjusts the probability of taking a random action versus the best action. Boltzmann action strategy chooses the best action with weighted probability between the best one currently estimated and a random one. The Boltzmann strategy is varied in T = [1.0, 0.1, 0.001]. The T value is used to control confidence of the algorithm in the action being taken. Finally, the Upper Confidence Bound (UCB) algorithm uses the uncertainty in the action estimate to balance exploration and exploitation by varying β=[1.0, 0.7, 0.5]. A small UCB implied high confidence in the action being taken, a large UCB implies more uncertainty in the action.

In all graphs below, the "Episodes" are in x1000, thus an Episode value of 20 implies 20,000 episodes.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

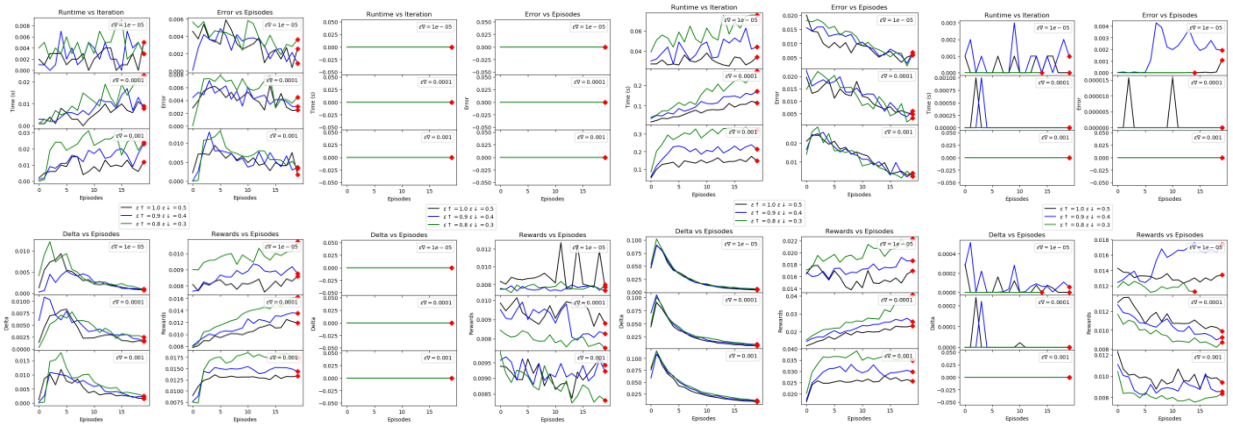


Figure 17: 4x4-50% ε-greedy Figure 18: 8x8-50% ε-greedy Figure 19: 4x4-80% ε-greedy Figure 20: 8x8-80% ε-greedy

ε-greedy Analysis – In observing the results for this experiment, both graphs for the 8x8 world seem to have failed. The results for 8x8-50%, Figure 18, graph tracks very miniscule rewards, but maintains 0 values for all other metrics. The 8x8-80%, Figure 20, graph demonstrates similar behavior with slight "noise providing values slightly above 0. The 4x4-50%, Figure 17, graph displays a good trend in the Delta, converging to a low delta over 20,000 episodes. The rewards graphs show a more interesting conclusion: in all cases, eMax=0.8 to eMin=0.3 is a better range for ε-decay. Furthermore the ε-decay value of 0.001 provide the largest rewards values. The 4x4-80%, Figure 19, graph is produced over a grid with more frozen tiles, as such it's behavior mimics that from Figure 17, but with much better fit between variables. What these results imply is the a faster decay and a range of epsilon that slightly favored immediate rewards fared better than a slow decay rate and favoring future rewards above all else. This is an interesting observation since, but not necessarily surprising since the ε-greedy algorithm only takes into account whether the action to be taken is most rewarding, or not. Probability, Confidence, and other metrics are not taking into account for an ε-greedy action.

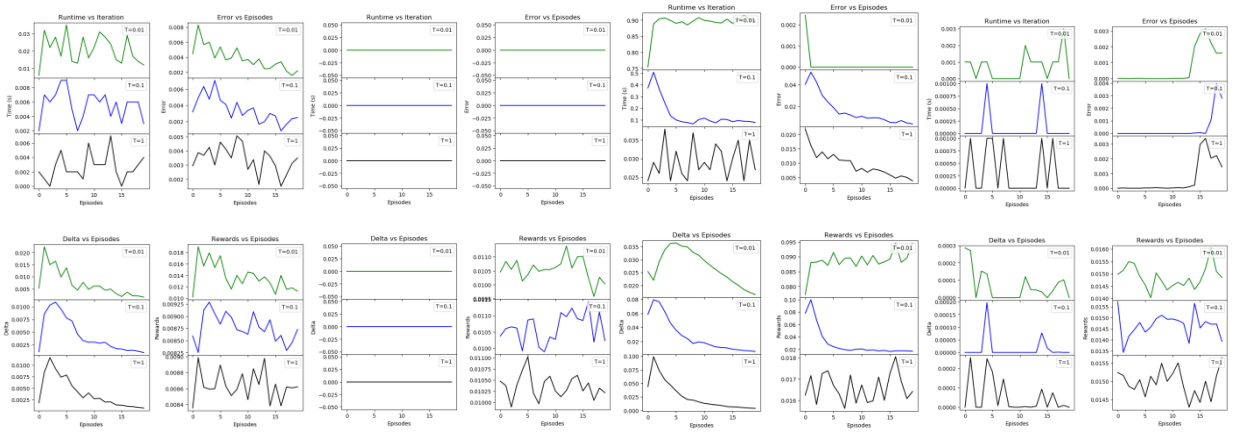


Figure 21: 4x4-50% Boltzmann Figure 22:8x8-50% Boltzmann Figure 23:4x4-80% Boltzmann Figure 24:8x8-80% Boltzmann

Boltzmann Analysis – The Boltzmann experiment provided some very interesting results. The most interesting part is the fact that it is difficult to garner any meaningful information from the observed results. Other than the failed of the 8x8-50% experiment, all others Figure 21, 23 and 24 observed delta graphs trending downwards. However, the Reward graphs are displaying quite random behavior. The rewards in Figure 23, For T=0.1 peaked and evaporated downwards as more Episodes occurred. The rewards for the others are sporadic. Furthermore, the error graphs in Figure 21 and 23 show an expected downward slope to 0. However, in Figure 24, the error graphs for all T values peak upwards at higher Episode values. What the Boltzmann action truly attempts to determine is the measure of how optimal the agent thinks an action is, not how confident it is in a specific action. It is difficult to judge the best next step to achieve good results from this experiment. It is possible that more episodes may yield good results, but Figure 24 shows Error peaks at high Episode values. Different T values may provide better insight, but the current graphs do not provide any clue how this should be adjusted. From observing all the evidence, the simplest conclusion can be drawn that the Boltzmann action selection algorithm may simply be impractical for this particular problem, and is geared more towards problems of other types.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

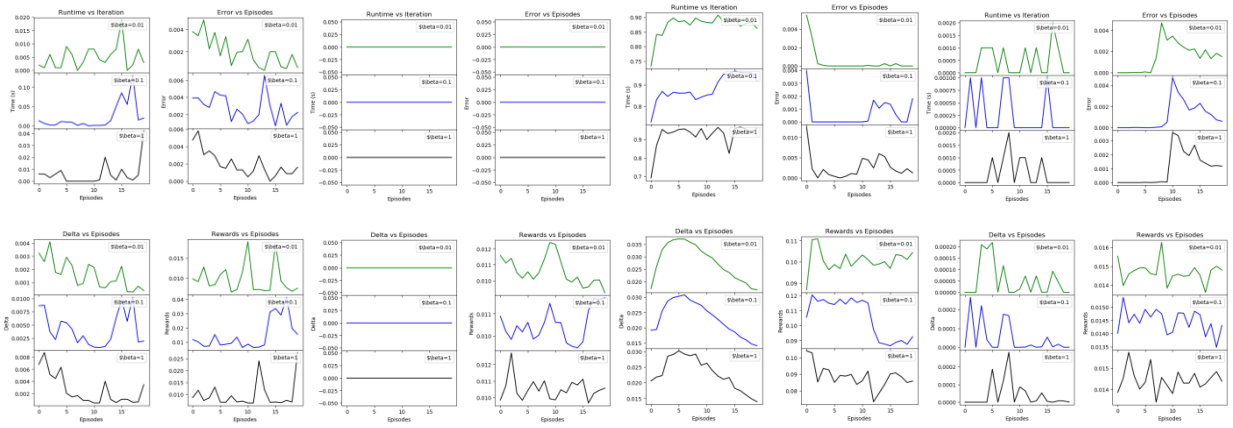


Figure 25: 4x4-50% UCB Figure 26:8x8-50% UCB Figure 27:4x4-80% UCB Figure 28:8x8-80% UCB

UCB Analysis – immediately noticeable is the failure of the 8x8-50% experiment with all graphs steady at 0 values except for the Rewards graph attaining slight rewards with more episodes. The Error graphs across Figure 25, 27 and 28 interestingly contain a spike around 10 Episodes. Considering all grid-worlds are created randomly and are different, it is strange to have a peak at the same exact episode range. Observing the other graphs, it is evident that the Delta, Rewards and Time graphs also have irregularities at the same Episode value regions. UCB picks the action with the highest uncertainty (upper bound). In doing so, the agent either picks the action with the highest reward, or learns the most about the state-space, either way proving a valuable action to take. Stemming from this, it can be concluded that at around the 10,000 episode region, the learners actions selected switch from most knowledge gained, to most rewards gained, or vice versa. Observing the rewards graph can provide a clue, whereby in Figure 25, the rewards graphs see spikes when rewarding actions are chosen. Conversely, Figure 27 shows a more interesting behavior of beginning with highly-rewarding actions which then dip to low-reward actions, implying the algorithm now chooses actions with more knowledge gain, rather than reward gain with higher episodes.

IV. Gambler's Problem

A. Environment Setup

The Gambler's Problem environment setup maintains two parameters: the probability of getting heads on a coin toss, and the maximum capital gains as the goal of the game. The state of the game is the Gambler's capital. To learn most form exploring this environment, the probability will be varied as pHeads=[0.4, 0.5, 0.6], the capital will be varied as defaulted to 100 and as maxCap=[512, 777, 1023, 1024, 1025].

B. Heads Probability Variance

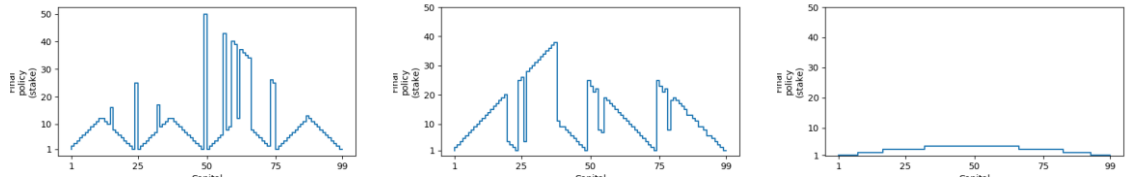


Figure 29: pHeads=0.4 Figure 30: pHeads=0.5 Figure 31: pHeads=0.6

The Policy Solution looks very interesting with different probability of getting Heads. Most notable, when the probability of getting heads is higher than chance, the policy is very conservative, betting a very small amount along the entire game until the goal is reached. This is a surprising outcome as a right-angle strategy, where there is an immediate large bet, seems more logical considering the coin is loaded to favor Heads. The strategy for pHeads=0.5 appears somewhat strange as an expected strategy for the probability of chance, it would seem, should look more triangular, rather than jagged. The pHeads=0.4 is a very interesting graph. This graph coincides well with that of Sutton and Barto Ch.4 graph of this problem. The random spikes, according to Li and Peyat, 2006, may be due to CPU configuration and floating point calculations. This may be the case, but it is interesting to observe the formation of 4 distinctive triangles with increase betting, and then a conservative withdraw of bets.

CS7641 Machine Learning (Fall 2020)

Assignment 4 – Markov Decision Process (Reinforcement Learning)

Sergiy Palguyev (gtid: spalguyev3)

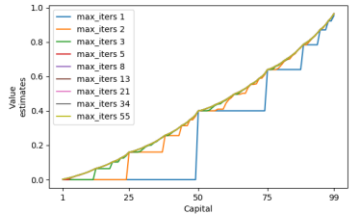


Figure 32: pHeads=0.4

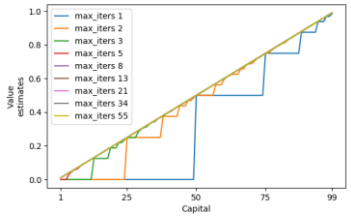


Figure 33: pHeads=0.5

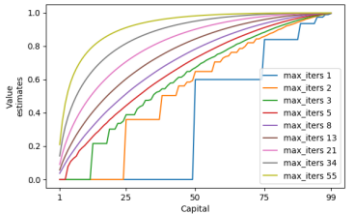


Figure 34: pHeads=0.6

The Value Estimates versus iteration show a very interesting trend for the probabilities of winning from each step in the game. For pHeads of 0.6, iterations above 5 show an increasingly parabolic value gain from earning just the first bit of capital gain. The fact that large number of iterations for pHead of 0.5 is linear makes very good sense as the coin is not loaded. The graph for pHeads-0.4 properly matches that from Figure 29, with slight peaks at capital values of 25, 50 and 75.

C. Maximum Capital Variance

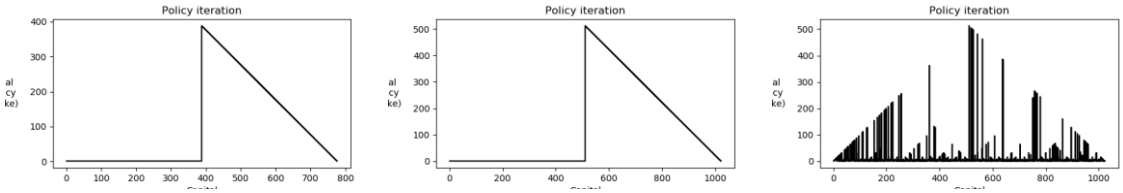


Figure 35: Policy Iteration for maxCapital=777 maxCapital=1023 and maxCapital=1024

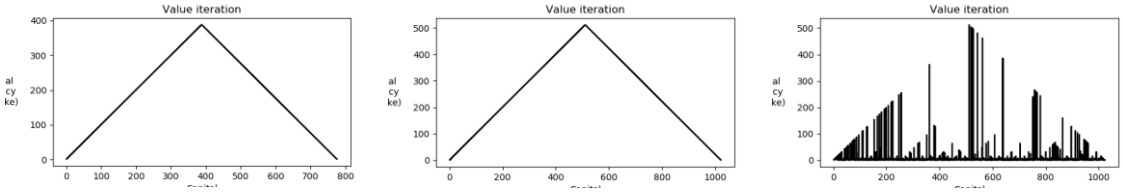


Figure 36: Value Iteration for maxCapital=777, maxCapital=1023 and maxCapital=1024

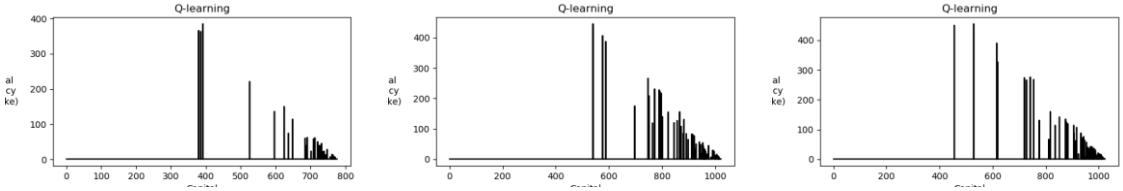


Figure 37: Q-Learning for maxCapital=777, maxCapital=1023 and maxCapital=1024

The maximum capital analysis led to a very interesting observation. Form Li and Peyat, the local maximum is the smallest integer value divisible by a polynomial of two from the number of states. The reason is that the gambler problem is a discrete MDP problem, and every state has an integer value as capital. From this we can see the behavior observed when the maximum capital is set to 1024, the resulting jagged policy resembles that of the one resulting for capital of 1023. It is also interesting to observe that the Value Iteration resulting policy incrementally increases it's bets until a half-way through the capital is gained, then decreasing until target, whereas the Policy Iteration resulting policy stays steady with very small bets until half the capital is gained and then bets it all in a decaying fashion until the maximum is gained. With Q-Learning, short bursts of maximum betting appear to be the best policy regardless of the nature of maximum-capital goal.

D. Time Analysis

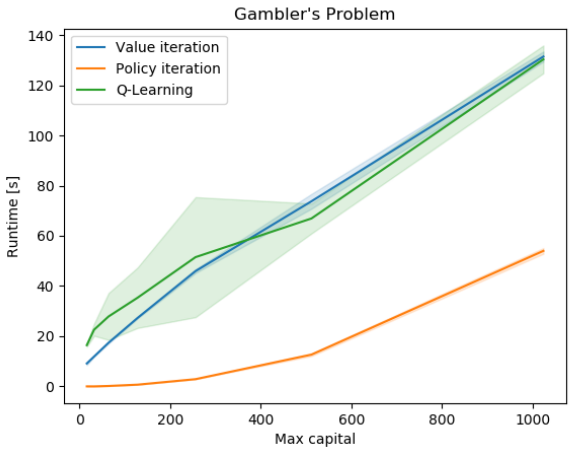


Figure 38: Value Iteration, Policy Iteration and Q-Learning time analysis over 5 iterations

Executing time analysis of the three algorithms with a goal of 1000 capital gains shows a very different behavior as observed for the FrozenLake experiment. The Policy Iteration algorithm appears to run the fastest regardless of the fact that it generates an entire new policy with each execution. Value Iteration and Q-Learning execute in a similar amount of time. This seems a logical result since both evaluate the value at each state of the problem. Finally, the variance observed with Q-Learning is expected due ot the uncertainty and random nature of the algorithm.

V. References

1. http://dl.ifip.org/db/conf/ifip12/iip2004/LiP04.pdf
2. http://web.stanford.edu/class/cme241/lecture_slides/rich_sutton_slides/7-8-DP.pdf
3. http://incompleteideas.net/book/the-book.html