CS7641 Unsupervised Learning HW4

Aleksandr Plokhikh (aplokhikh3@gatech.edu)

Introduction, dataset selection, data preprocessing

The goal of the assignment is to study solutions for Markov Decision Problems (MDP), via model-based algorithms such as Value Iteration (VI), and Policy Iteration (PI), and model free reinforcement learning algorithm.

As a first problem, I took the Frozen Lake model, since this model works with a multidimensional world (in the general case) and has uncertainty and stochastics in movements. It lays a fundament to robotics, car autopilots operation, etc. All futurologists of the past (until 2022) thought that the next breakthrough in AI will be in robotics, and there were prerequisites for it like work of Boston dynamic, but great leap came from completely other thing of AI field, and robotics is lagging behind probably because to teach it we need a real interaction with the physical world, good exploration-exploitation models and balance, and we cannot achieve it with extensive calculations on GPUs. I used an 8x8 Frozen Lake model with 64 states, which is considered to be a small problem. In my work I will mainly use analogy not with frozen lake but with The Lord of the Rings. Map is the map of middle earth, and we need to deliver the ring of power from Shire to Orodruin. I think it is more live, and an amusing analogy. Also, it is much easier to explain the concept of discounted reward here – the longer we go to Orodruin the less Middle Earth dweller we will save during the war with Sauron, and Stronger the Sauron – the faster we need to destroy the ring and make more riskier movements.

The second problem is Blackjack. Of course, the problem has been solved a million times before and, on average, you cannot beat the house, but we probably all saw the movie “21” where guys counted cards in Las Vegas and turned, results expectancy on their side. Current release of the cards from the deck changes the actual value of the MDP state. We can add released cards as prehistory and make new MDP states based on it. The state space in this case can be as high as trillions instead of the basic 290, which is not true as a basic model takes all cards with replacements, which does not happen in real life. Modern casinos increase the number of decks to increase their own margin, and the sampling without replacement has become close to drawing cards with replacement, but is still not the case, and we need to take it into account and use more complex models. MDP with trillions of states based on the prehistory of released cards can be useful for finding the best strategy, but it will be very hard to find one, so the best judgment should be applied, and simple counting like in the movie can be helpful and increases the number of states to only several thousands, while providing a better strategy than the default one with no knowledge of prehistory. I do not think that the guys in the movie solved MDP for the most popular “hot” state of the deck, so it will be interesting to evaluate them overall. Plus, basic strategies do not push players to the end of the game faster, and treat the world as if we have unlimited time (works with gamma=1) but in real life and as in the case of movie “21”, if you get cards, the casino security may come after you, and you may have only few minutes left to finish several games and escape with prize before got bitten by security team. In these cases, we may want to finish games more quickly and sacrifice the pursuit of an optimal strategy, which may require, reveal of additional cards. For that, we can solve MDP with discounted rewards (gamma<1). So, the blackjack problem is more interesting, undiscovered, and harder than it may seem. In the scope of the assignment, I will explore only the standard case with 290 states, which is already considered big according to the assignment note and leave much harder cases for further investigation.

Frozen lake value iteration

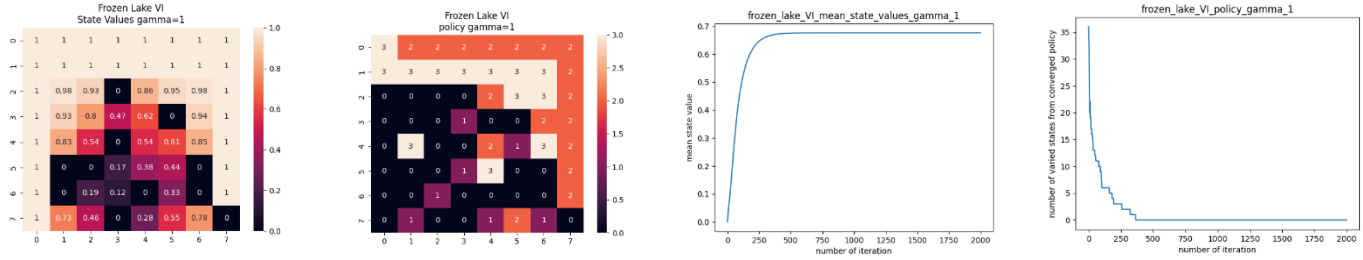

Fig.1 Value iteration on frozen lake problem with no discontinued reward (gamma=1)

At condition of no discontinued reward (gamma=1) the state value of the start point is 1 which is equal to the reward of the end which says that if we do not have discontinued reward and allowed to play as long as we want, we are guaranteed to finish the game at the goal and will not fall in the hole. The holes have a state value of 0 which is obvious since games do not continue if we reach these states. Since on the whole map only goal state has nonzero immediate reward than positive state values gradually spread across the map with every step of VI, until saturation with logarithm like manner which is seen on the plot of median state value vs number of

iterations. The algorithm takes 569ms and, 1425 iteration to converge with the standard epsilon of 1e-10. Mean state values do not change much after 300 iterations. With tightening/losing epsilon requirements the convergence time will exponentially increase/decrease respectively which clearly visible form the mean state value graph which clearly has logarithmic behavior. No epsilon skew is interesting to make here, the behavior is obvious. What interesting to see is when the policy convergence to the optimal one because it is always nice to know the precise expectancy reward at the state, but it may not worth extra calculation cost if further improvement of the state value to the true one does not change policy which may be already optimal, and this extra calculations in this case does not improve our performance in the game since we use the same policy anyway, and will cause just a waste of resources on computation, and the plot with policy difference shows that the policy converges to the optimal one in only 365 iterations, and we need just 156ms for it, which is significantly less than the number for state values convergence.

On the policy map proposed movement to the left denoted as 0, down-1, right-2, up-3. If we look closely in the bottom left corner of the map, we see that the algorithm proposes to avoid holes at all cost and do on the right side trying to navigate agent via road with holes to the goal with short path (does not go in the Mines of Moria in Khazad-dum, due to the presence of Balrog there). Instead, it proposes to “hit” the wall and assume that due to stochastically of the world we will eventually reach the top of their map and will go around all holes to the goal, sticking to the boarder of the map. On the right side of the map, the agent also proposes to “hit” the wall and rely on the stochastically that should eventually lead us to the goal – Orodruin to destroy the ring as in case if we have unlimited time to do so. So, the agent acts like the initial plan of Gandalf – just go around -avoid pitfalls, and we guaranteed to destroy the ring of power and save everyone if Sauron did not start the war yet.

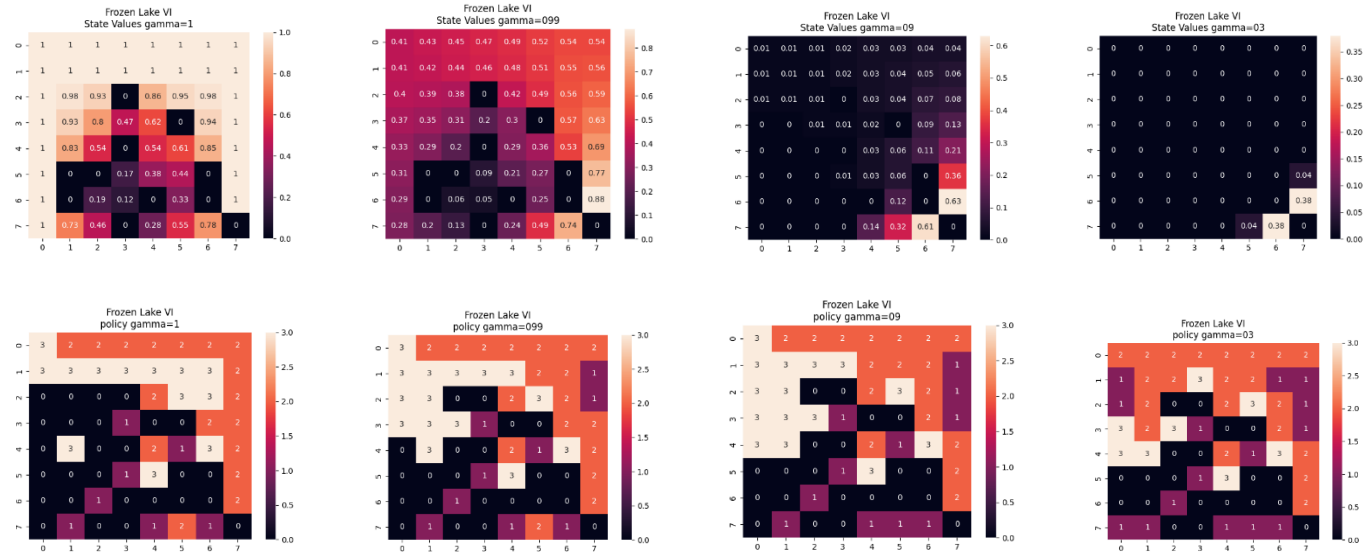

Fig.2 Value iteration on frozen lake problem with discontinued reward (gamma=1, 0.99, 0.9, 0.3)

Completely other things start to happen if Sauron started the war, and we cannot save everyone. Every step, Sauron kills some present of the middle earth dwellers. Not everyone can be saved. If gamma=0.99 we kill 1% f middle earth population every step, and state value of all states dramatically drops. The further state from the finish – the higher the drop. If we start at shire, only 41% of all dwellers can be saved – it is a completely different game, and our strategy changes as well. Now on the left border of the map on the top tiles which are pretty away from holes we do not hit the wall but go up in order to go around faster, here we take the risk to come to the holes closer to save more lives, but this change does not make too much sense since we should not appear at these tiles if we stick with the optimal strategy. The most meaningful changes are happening in the top right corner -with the new gamma=0.99 we stop hitting the wall in the states which are far away from holes (not very risky states) and go along the wall rather than hitting it to save time. If we decrease our gamma even further to 0.9, we can only save about 1% of middle earth population, the changes to the policy map become more prominent, and remain the same character, now more tiles on the top right and left corners change their direction from hitting the wall to go along it. The states which were considered to be risky to change at gamma=0.99 are not risky that much anymore considering higher discount, and now subject to change. On the mean, we reach goal faster but have more chances to die in the hole. Gamma=0.3 is an incredibly special case here because the discount rate of 0.3 is less than the probability of successful movement. Here there should not be any greater risk to the reward than losing time, and indeed with such discount rate the agent proposes to go right away in the “Mines of Moria” dash through the middle ground of the map with all holes and finish the game earlier-try to destroy the ring as soon as possible regardless of all pitfalls there. The agent navigates us between all the holes.

Since Gamma reduces further reward and state values, it is no surprise that state values converge to stable values in a low amount of time, and iterations at given epsilon convergence criteria (see Fig 3). What less obvious is why policy also converges faster but if we look deeper with discounted reward, we just need to know the values of states in the shorter vicinity to build a right policy do to the discontinued reward of later states, and since VI gradually spread the rewards through the map step by step from the states with immediate reward and game stop tiles to the rest of states. Here we need less steps for good enough state values which provide a stable policy. Then we decrease gamma from 1 to 0.99, to 0.9 to 0.3 the amount of required iterations for state values convergence reduces in the row 1426, 662, 158, 17 respectively, while the number of iterations needed for policy convergence reduces in the row 365, 127, 48,16. Interesting to see that for gamma 0.9-1 we need 3-5 times more iterations for state value convergence when policy convergence which does make more matter than state convergence while for 0.3 policy and state convergence happening almost at the same time probably because a very low mean state values at this case since high discount in the reward and epsilon/mean state value is high here, and the relative convergence margin is orders magnitude higher than at gamma 1-0.9.

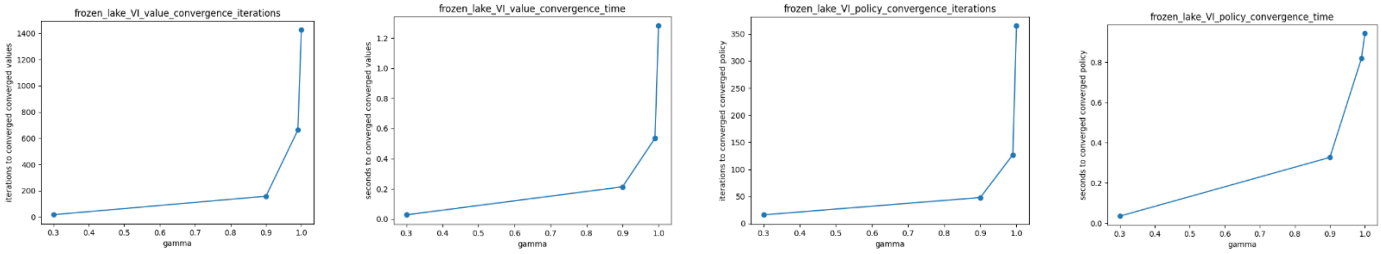

Fig.3 VI state values, and policy convergence on frozen lake problem with discontinued reward (gamma=1, 0.99, 0.9, 0.3)

Frozen lake Policy iteration

PI – is another model-based method to solve MDP. Each iteration of the algorithm is much more costly than in the case of VI since each time we need to solve a system of linear equations, but in this case true state values propagate though the state space much quickly via randomly preselected policy at each step. This leads to faster convergence than VI in terms of iterations, but real time cost may vary. Initial randomness in generation of the first policy map may greatly affect the convergence speed, so I modified the algorithm to parse random seed to it.

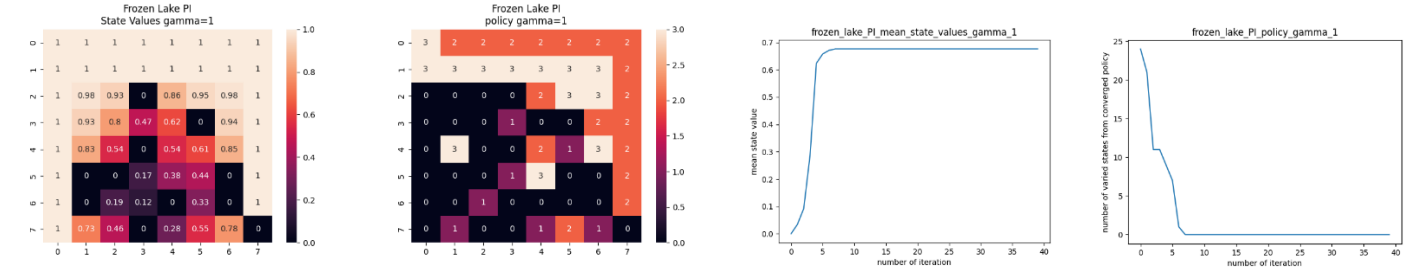

Fig.4 Policy iteration on frozen lake problem with no discontinued reward (gamma=1)

PI converges to the same policy as VI at gamma=1, and state values are the same with accuracy of epsilon, and we cannot expect any other behavior as both algorithms are model based and do converge to the true states and optimal policies, no surprise here. What is interesting to see is how fast it is happening compared to VI, and indeed PI needs only 9 iterations for state values convergence, and 7 iterations for convergence to the stable policy (with VI we needed 1425, and 365 iterations respectively). Unfortunately, it did not transfer to faster run time but on contrary it took more time to run the algorithm -2.36s vs 1.28 for VI. The shape of the "learning curves" are remarkably similar to the one in case of VI. The shape of the mean value curve is fast rising and reminds logarithmic curve but not as close as in case of VI where we have a specific state value update algorithm. PI is different here, and many things depend on the initial policy which drives state values updates on first iteration of the algorithm.

State values at different discount rates exactly match the one from VI ran. But the policy map at gamma=0.3 is slightly different. The only title which is different is ironically the start state, and due to it the difference is very crucial. VI proposes to go right which is sane movement at such discount rate, but PI proposes to go down, and it will lead to harder time navigating between the holes. Such artifact is occurred because the epsilon of 1e-10 here is not infinitely small but is greater that the value of the start state which is 9.7e-12, and it led to not really converged map, and it is surprising that with such ratio of epsilon to the state value all other polices are the same! Extra error could also rose from the solving of the system of linear equations in PI, since the dtype of the map is only 64bit, and we

already have values of 9.7e-10 which leads to fast accumulation of the error in calculation which lead to mistake in the policy map. So, at aggressive discount rate we need to use much lower epsilon, and higher bit values.

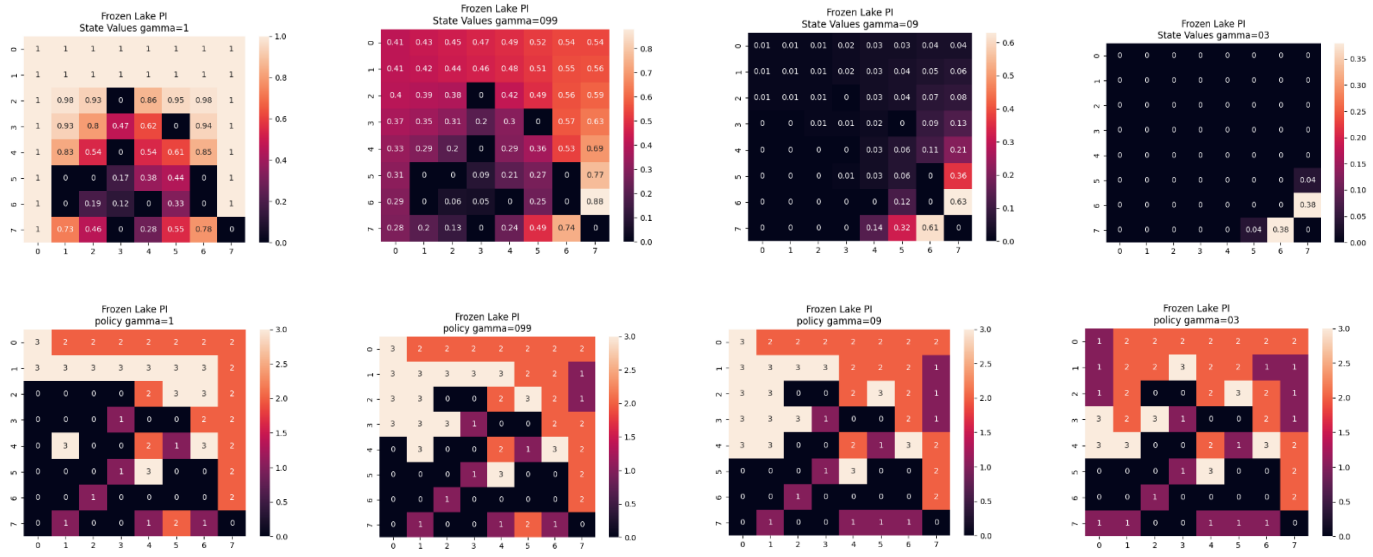

Fig.5 Policy iteration on frozen lake problem with discontinued reward (gamma=1, 0.99, 0.9, 0.3)

It is also interesting to see is how much time and how many iterations of the algorithm do we need for state values and policy convergence. Runtime did not give us any surprises – the behavior was the same as in the case of VI. The run time almost exponentially decays with decrease of gamma, but iterations plot gave us a peculiar artifact the run with gamma=0.99 required the least amount of iterations for state values, and policy convergence this observation almost certainly rose from the random nature of the initial map.

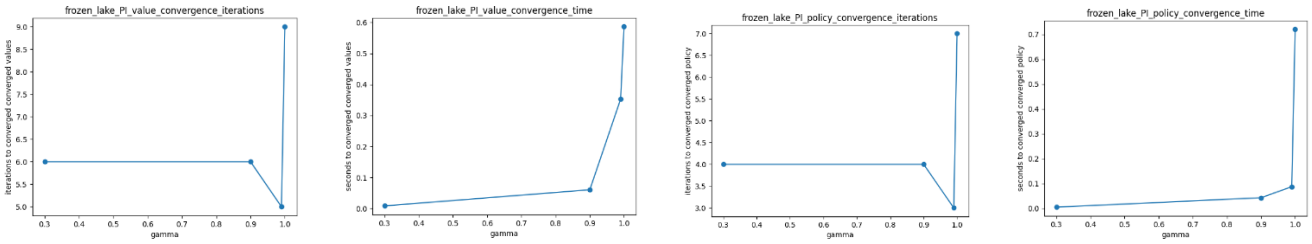

Fig.6 PI state values, and policy convergence on frozen lake problem with discontinued reward (gamma=1, 0.99, 0.9, 0.3)

Frozen lake Reinforcement learning (Q learning)

For the RL part of the assignment I used Q learning algorithm with fixed epsilon, and alpha without decay for better comparison during varying the hyperparameters.

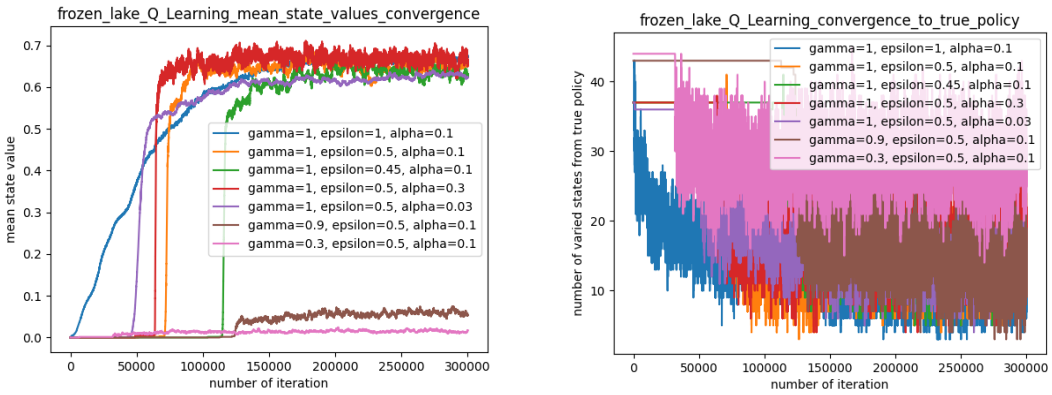

Fig.7 Q learning state value, and policy convergence

One of the main aspects of reinforcement learning and Q learning in particular is exploration-exploitation dilemma -for quickly learning the best policy we must make a lot of random actions, but choosing many random actions prevents us from taking benefit from learned strategy. Q learning perfecting the value state and policy not only when we chose random action but also when we follow the existing policy which helps polish and fine-tune it, and it is interesting to see wheat is the fastest way to learn the best policy-do only exploration and make random choices every time or find some sweet spot between random choices and fine polishing of the existing data. In Q learning, epsilon is part of random choices in the decision-making process. At gamma=1 epsilon=1 the mean state value starts to rise right away, the positive state values start to spread out from the finish state. The rise pace is moderate and tapers out at the values around the ones we saw with model based VI, and PI algorithms. The rise is logarithmic-like, but not as sharp as in the case of VI. The policy is also starts to perfect it-self during learning process almost immediately, and probably the better than all other runs on the late stages of learning, but never matches the ideal policy obtained by VI, and PI algorithms, while at the late stages the number of altered states is low (2-12) it never reaches the ideal policy and the policy is very unstable as in all other cases.

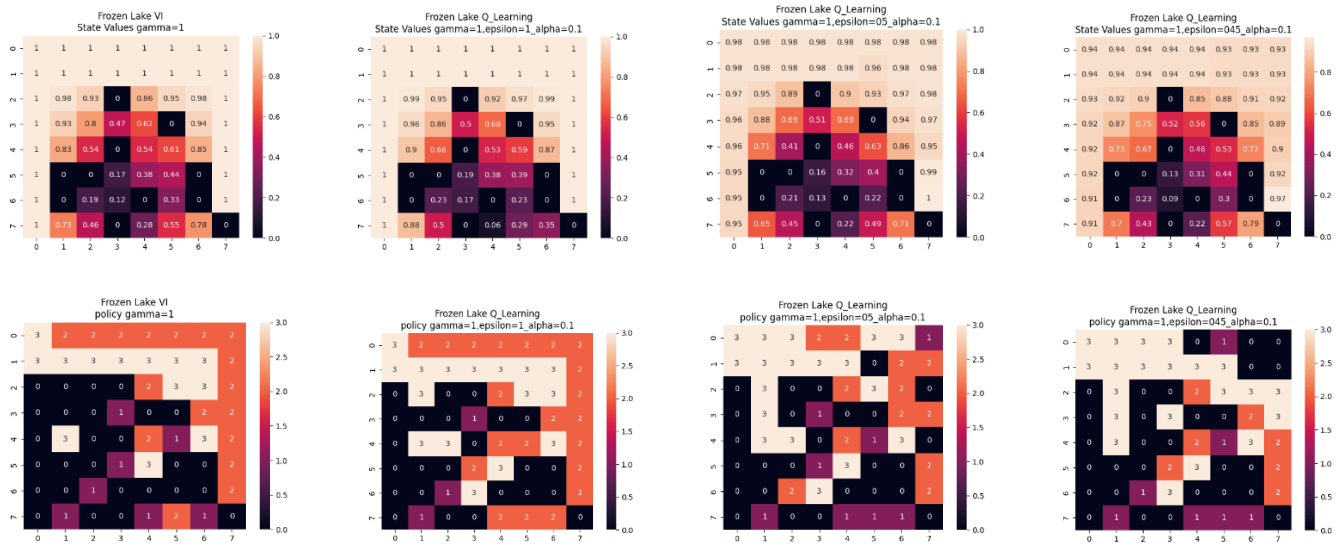

Fig. 8 Value states and policy map comparison between VI and Q learning with different epsilon

When we reduce epsilon to 0.5 learning behavior changes dramatically – which means state values do not change until the 70000 iterations mark. I think it happens due the fact that on the map there is only 1 state with non-zero positive immediate reward – the goal state, and it is the last state in the NumPy array, but in the q learning algorithm then we do a sane move we chose the one with the highest Q value, but when all q values are 0 we do a tie-break and here we chose the state which is closer to the beginning of the NumPy array, and it significantly delays the initial learning (steep learning curve), but once the map picked up some nonzero-state values, the “sane” past of Q-learning quickly starts to spread positive states value across the map and for some time from 75000 to 150000 iterations this agent outperform the completely random Q learner with epsilon=1 in terms of mean state value, after 150000 their performance match, the same story happens with policy perfection – there is a delay with policy update up to 70000 iterations, and after this threshold policy updates to more reasonable one with altered states similar to the learner with epsilon=1. When we sightly decrease epsilon to 0.45 it moves the “learning threshold” significantly further to 120000 iterations mark, at epsilon=0.4 the threshold is already close to 50000 and goes beyond the scope of reasonable observation.

The Ideal VI or PI policy at gamma=1 leads us along the top and right border to of the map to the goal, and in all states on the path state value is 1 so we guaranteed to finish the game in the goal he same patterns is present in Q learner with epsilon =1, when we decrease epsilon to 0.5 the state values on the best path are 0.98, at epsilon=0.45 the values are dropping down even further to 0.94 so with decrease of epsilon q learner is less curtain that it can guide us to the goal. The actual strategy varies a lot but anyway lest compare them with the model based ideal ones. If we look on the policy map we will see that at epsilon=1 the top two rows and right columns have exactly the same policy as in case of the VI/PI learner, and it is the only states which we may visit if we will stick to the optimal policy So for us there will be absolutely no difference between VI/PI and Q learned strategy with epsilon=1–we will make exactly the same movement regardless the fact that in the middle of the map there is some variation in the policy, but it does not matter since we will never visit those places. At epsilon=0.5 visible changes appear on the top two row and the right column which distracts our path two small fluctuation of the policy map on the policy map actually makes it impossible to finish the game – this is the issue of model free instance base q learning – there can be fluctuation in the policy which prevents us from finishing the task, and all high state values become irrelevant since we are going in the loop. At epsilon=0.45 even bigger disturbance occur – in the top right corner we have an “island” with 4 state which repels us back in the direction of the start, and it also creates the same never-ending

loop of movement, which we cannot break, but here it is even more prominent more savior and needs larger policy change to avoid than in the case of learner with epsilon=0.5, and Q learner cannot know about such flaw since it does not have the model. So, it is quite easy to get useless strategy in Q learning if we do not make enough exploration -random movements, but it is best what we have if we do not have the model.

Another hyperparameter in q learner is the learning rate alpha If we decrease the learning rate in the raw 0.3, 0.1, 0.03 it makes state value and policy convergence graph less noisy, and for alpha 0.03 state values takes much longer time to catch up the other curves and rise to the highest value 0f ~0.65 which reach all other learners including the model based VI, and PI. Such behavior is completely expectable and have analogy in many other learning techniques. High learning rate helps us to learn new thing, low learning rates helps us to remember things we learn and do not overwrite knowledge with probable noise -common learning dilemma.

When we decrease gamma in row 1, 0.9, 0.3 the mean state value drops like in case of model based VI/PI. The behavior is also expectable.

Blackjack, Value iteration

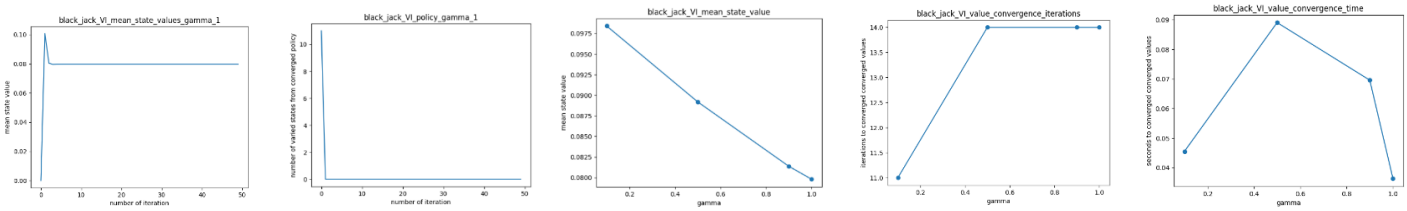

Fig 9 VI learning on blackjack problem with gamma=1, and mean state value, SV, policy convergence vs gamma

Despite the larger number of states 290 vs 64 the VI algorithm learns on blackjack problem much faster than on frozen lake because of much lower mean number of steps needed to finish the game, and much larger number of finish goal states which helps to propagate state values faster via state space. State values converge in just a few iterations in milliseconds. The mean state value for blackjack is positive, and we may falsely think that it is profitable to play in black jack against the house due to this fact, and we may wonder why casino offer blackjack as a game and earn money on it. The issue comes from the fact that when we calculate mean we give every state equal weight as they appear with the same probability during the game, but it simply not true for example in order to have hand worth 4 we need to have two 2 cards and probability of that is much lower than in case of hand worth 15 for example since there are much more valid combinations to get 15 than to get 4, and for example the probability of taking 21, or very strong 20 with the first dial of cards is much smaller than for example 13 or 15 which are less profitable but occur more often. This fact artificially drives mean state value on a positive scale, but it has nothing to do with the real outcome of the game, since for that we need to consider probabilities of initial dials at least. During the learning process we have a positive spike which may occur due to the fact that “strong hands” they on average require less card draws if any if for example we have 21,20,19 in the initial dial a need only few VI iterations to reach it’s positive state value while “week hands”, and hands with low value which are also weak require more turns to finish the game and the expected outcome is negative, so all negative outcomes spread its sate value longer in VI than the wins -it creates the observed spike on the mean state value “learning curve”. For the same mean, state value rises when we decrease the gamma since delayed negative reward for the weak hands will be discounted more than much closer positive outcome of “strong hands” it leads to monotonically increase of mean of the state values with decrease of the gamma. Overall, it is unlikely what strategy with discontinued reward on the real blackjack table quite the opposite, but if the security team is already on the floor to take you in torture room for cards counting you may want to finish the current game quickly and make it the best way possible here discounted reward is useful.

With decrease of the gamma number of iteration also decreases as in case of frozen lake since the further away from the finish states which represent weak hands still have negative but closer to zero state values, and now it is easier to converge them to fixed value while the “good hands” are closer to the end of the game, and they update their state values are much faster to the true one. There is no distinct downward trend in the runtime meanwhile because the runtime is very low, lower than delays caused by the operating system during resources allocation to run the program, so all the time changes are invisible in this noise imposed by operating system.

Blackjack policy iteration

As in case of VI value and policy convergence take few iterations, and several milliseconds to converge, it is much faster than in case of frozen lake for similar reasons. Mean state value, and policy change learning curve are logarithmic. The mean state value curve has a very prominent downwards spike which is certainly related to the choice of randomly selected policy during initialization of the learner, which affects the performance of the algorithm on the first iterations. The mean state value of converged learner is positive, and we

have the same mean state value vs gamma dependency, the causes of both unusual observations' observation explained in the previous chapter. There is also no surprise that PI converges to the same means state values since both algorithms are model based on converge to the trues state values and to the optimal policy.

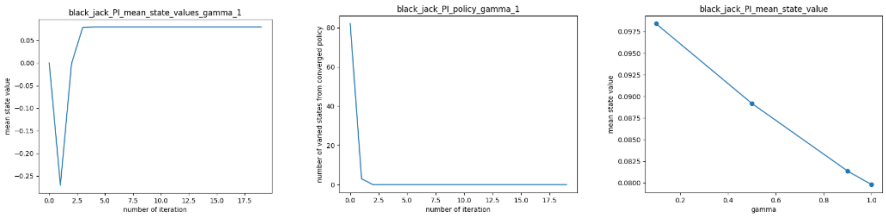

Fig 10 PI learning on blackjack problem with gamma=1, and mean state value dependence on discontinuation of reward

Here on average policy convergence takes 2-3 times fewer iterations than the policy convergence, and we see the downward trend of number of iterations needed for convergence with the decrease of gamma. The runtimes are still too small to measure them reliably, but they have signs of downward trends as well.

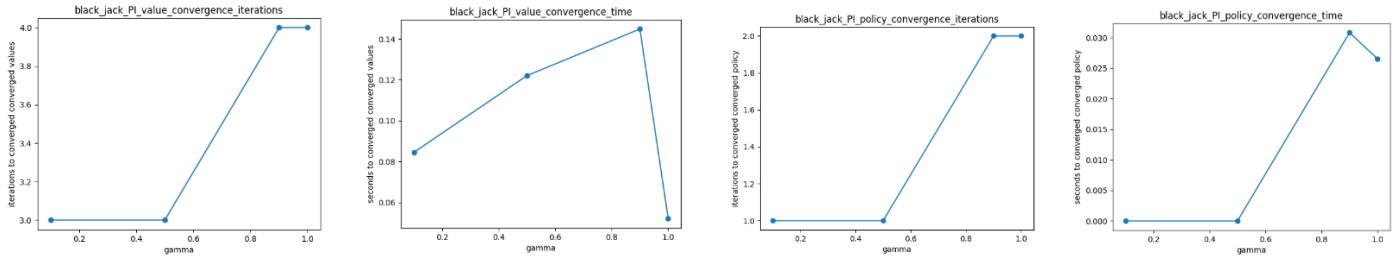

Fig.11 VI state values, and policy convergence on black problem with discontinued reward

Blackjack Q learning

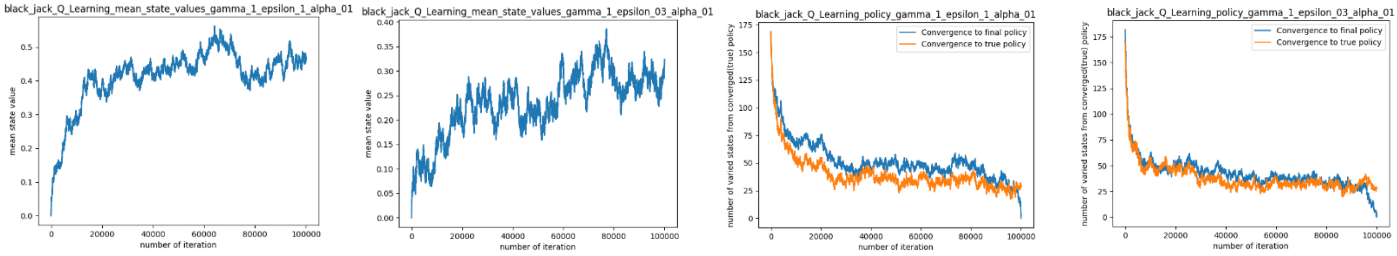

Fig.12 Q learning state value and policy convergence plots at epsilon 1, and 0.3

In case of Q learning we need much more iteration in order to come up to good values and reasonable policy and in this case large number of states starts to make a penalty because in VI and PI we updated all the states each iteration but in Q learning we update only one state at the time on the case by case basis. Even after 100000 with epsilon=1 the policy is far from ideal, and there is no sign of soon convergence. When we decrease the fixed proportion of random movements (epsilon) from 1 to 0.3 it only slightly decreases the rate of mean state value rise, and both values starts to increase from the very first iterations of the learner which is very different with the behavior we saw on the frozen lake problem. This behavior can be explained by the fact that in the blackjack problem we have much more states with non-zero instant reward while in frozen lake we had only one such state, and it very dramatically increase the probability that at the initial iterations the q learner will go to the states with immediate reward and hence will update the Q values of the state and utility value as well. Low mean number of movement till the end of the game (low depth of it) also positively affects on fast update of the Q table and utility values. There is no sign that the exploration only model gets them to the good policy quicker than the one with epsilon=0.3 quite the opposite. Polishing of the existing policy give its own useful fruits and the strategy with “saner” q Lerner gets a bit better strategy on the early stages of learning – the same behavior we saw on frozen lake model.

Interesting that this time at learning rate of 0.3 the agent learns something almost instantly thanks to the high number of states with instant reward, while decrease of the learning rate alpha not only slowed down the learning process, impeded the rise of the mean state value, and smooth out the mean state value and policy convergence plots but also made the resulting policies much closer to the most efficient policies achieved by model based VI and PI algorithms. This behavior can be explained by the fact that in case of the

blackjack each movement moves us in larger variety of different states as a new card which we draw can have 11 different values and not all the values are equally probably like in case of values 10 while in the frozen lake we had at max 3 outcomes of our movement. This aspect of blackjack world pushes us to remember more -try to infer probability of 11 different drawing cards, so the lower learning rate is beneficial here since it helps us to remember previously obtained knowledge about the world. And it is also interesting to see that slower learning agent which provides policy which is closer to the best one have lower mean state value number so a high mean state utility value can be misleading indicator of well-trained learned with good policy.

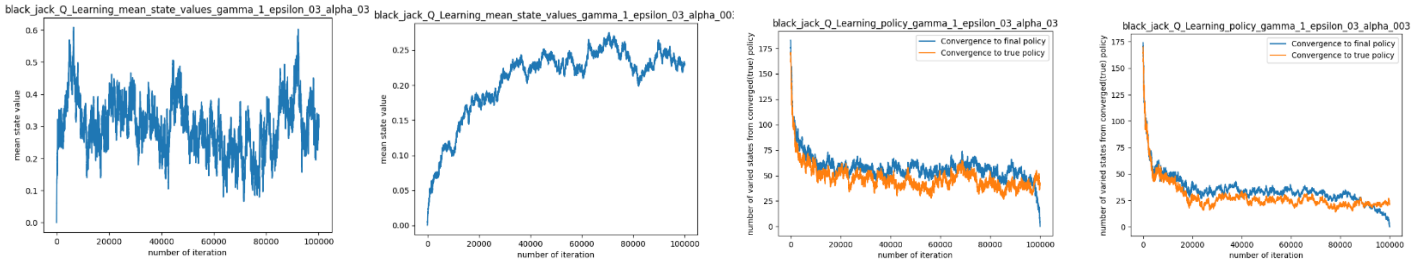

Fig.13 Q learning with different learning rate and its affect on mean state value and policy convergence to the optimal policy

Decreasing gamma increases rate at which mean state values come up to its final values faster which does make sense for the same reason described several times above-states with positive true value ("good hands") tends to have fewer actions to finish the game, and they faster get their true value and the value is not very discounted while "week hands" need more cards to draw – more steps in the game they need more time to come to final negative value and this value is heavily discounted.

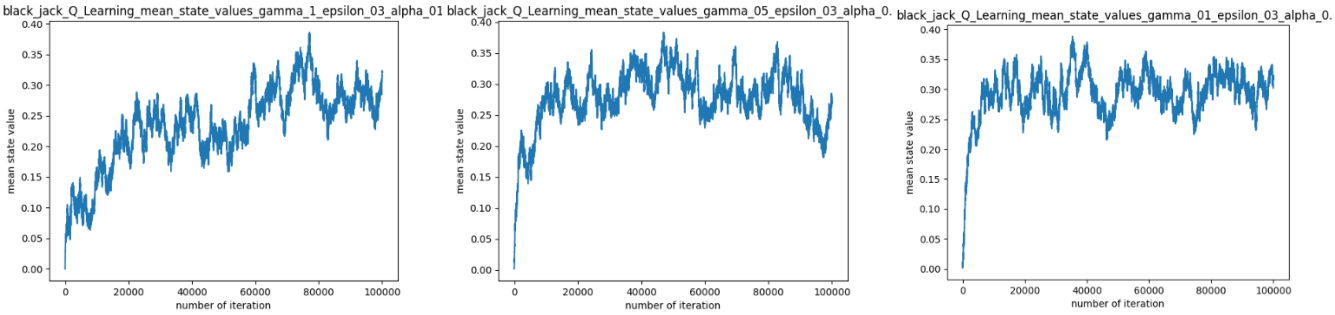

Fig.14 Q learning mean state value convergence at different discount rates (gamma=1, 0.5, 0.1)

Truncated Conclusions

RL is orders of magnitude more expensive than a model-based method to solve such as VI or PI, so if we have a model, we better use it even f it is not ideal. PI always get to the converged states faster in terms of iterations, but each iteration sometimes orders magnitude longer than in case of VI since PI had to solve a system of leaner equations for states update it leads to the fact that VI often runs a bit faster regardless much higher number of iterations to reach converged state. At sufficiently low convergence allowance, both VI, and PI converge to the same policy and the same values. There cannot be variation if epsilon is small enough. High number of states penalties all learners (make run time longer) but the RL learners suffer the most since they update only one state at iteration on case-by-case basis and may not even have enough observations for good policy construction while VI, and PI update all the states every iteration. High stochasticity especially if we have many possible next steps for each movement also penalties run time of all learners and again RL suffers the most for the same reasons since it learns case by case while model-based learners updates all the states with all possibilities every time. Q learner suffers even more if we have an outcome with low probability but unusually high or low state value. What damages the run time of all the learners equally badly is the depth of the game (how many steps we need to take to the finish). The higher the depth – the exponentially longer the run time. If we have small number states with non-zero instant reward it significantly delays convergence and for RL learner, we must shift our priority in exploration side to reduce the delay with learning the policy. If depth of the game is significant, we may get useless strategies which may not lead us to the go regardless of the utility values of the states it happens because RL update only small portions of the state space at the time independently of the rest space it often leads to local fluctuation of policies. If the world is highly stochastic, the low learning rate in RL not only gives more stable mean state values and policies but also makes policy closer to the optimal once since in this case states better remember previous statistics of stochasticity. If the depth of the game is high, the discontinuation of the reward can make initial state values even smaller than epsilon, which leads to false convergence. Convergence criteria in such cases must be reduced accordingly.