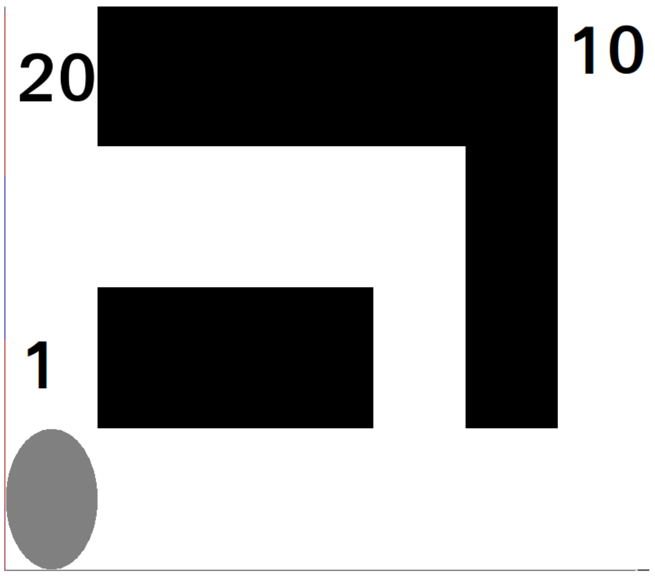
Anish Moorthy (amoorthy8)

CS4641 Project 4: Markov Decision Processes

Section 1: Easy Markov Decision Process



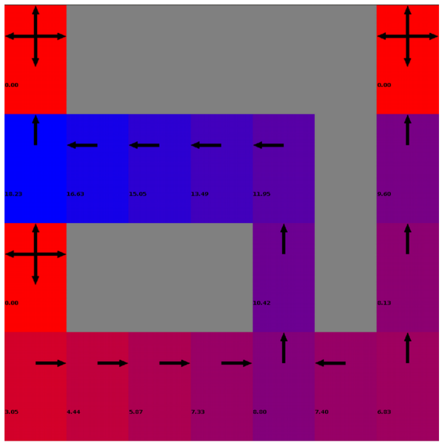
For my first MDP, I created a small gridworld layout. The setup is as follows: at each step, the agent receives an "alive" penalty of 1 (that is, a net change of -1). The numbers represent terminal states which yield the listed reward on entry. Finally, the black spaces represent barriers which the agent cannot cross. The agent can move stochastically in the four cardinal directions: with probability 0.8, movement functions as "intended" and with probability 0.2 the agent moves perpendicular to the intended direction of movement. This MDP technically has a total of 24 states: however only 14 states can ever be accessed by the agent, 3 of which are terminal.

Although the MDP is exceedingly small, it is not trivial. For instance, take the reward state immediately adjacent to the starting position. This state poses an exploration-vs-exploitation challenge: should an agent continually choose to exploit the reward of 1, or should it endure highly negative rewards in order to venture out and seek rewards? Value/Policy iteration will not experience this issue, since they are able to obtain optimal solutions as a consequence of knowing exactly the dynamics of the MDP, but any reinforcement learning algorithm must confront this question.

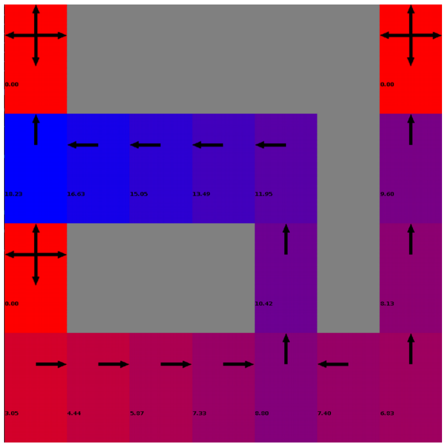
Another interesting feature of the above MDP is that, depending on your current state, the optimal choice of goal changes. For instance, the path from the 10-reward state to the 20-reward state is 12 states long. Combined with the fact that nearly 20% of the agent's moves are "wasted," this means that in states close to the 10-reward state, it is best for the agent to cut its losses and move towards the lower-reward state..

Results

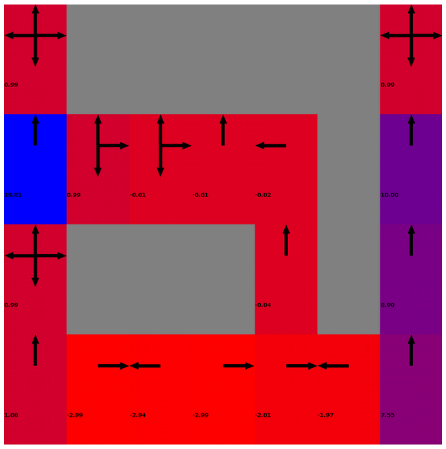
Value Iteration



Policy Iteration



Q-Learning



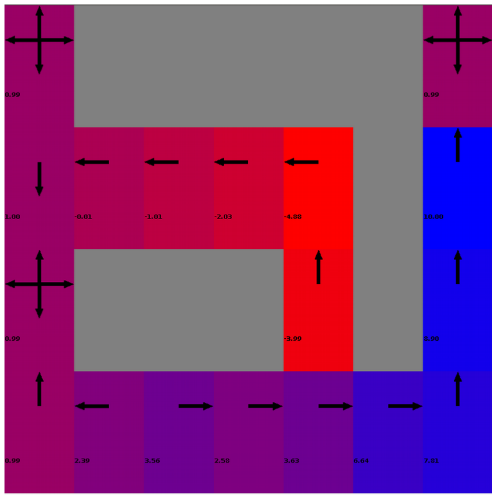
Each learning method was run for a total of 100 episodes, where an episode is defined as

1. A single pass of value / policy iteration of every state of the MDP for Value/Policy iteration
2. A single sequence of actions the agent takes until a terminal state is reached.

Value and Policy Iteration both converge to the optimal solution. However the times for convergence are illuminating. Value Iteration takes **3.02s** to converge, while Policy iteration and Q-Learning converge quickly at **1.6s** and **.19** seconds respectively. Illustrated here is the fact that in practice Value Iteration tends to converge long after the policy it defines has stabilized, due to the fact that it is really only the relative ordering of values which matters. Policy iteration does not have this problem, and therefore converges in a much shorter period of time.

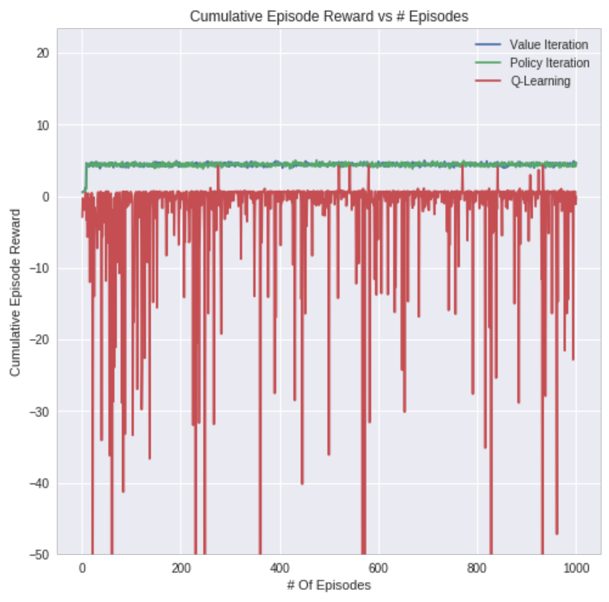
Unfortunately, the Q-Learning reinforcement learning algorithm does not reach the optimal policy, or anything closely resembling it. As I detailed before, the problem is one of exploration vs exploitation (one which Value/Policy Iteration circumvent entirely: we see that the agent becomes “addicted” to the low-reward state, and learns to exploit this fixed reward over and over. Random initialization of starting state helps (and seems to have led to some success on the right side of the map), but in general the policy is incoherent: observe, for instance the numerous locations where following the policy would just cause infinite cycles.

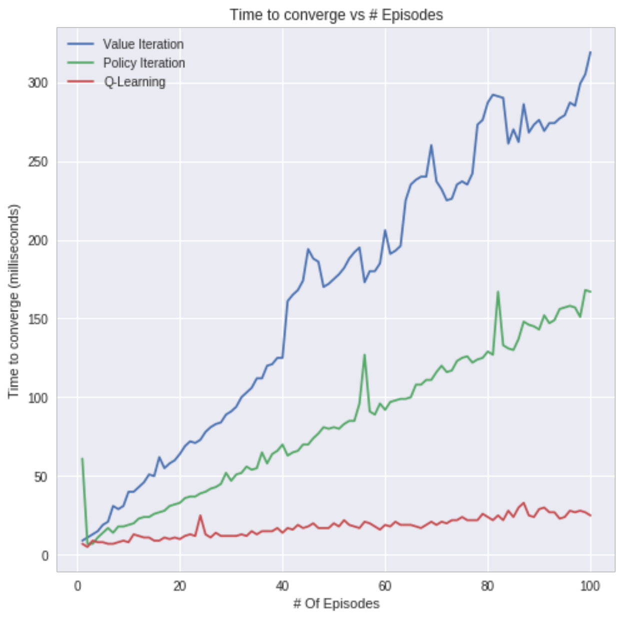
Increasing the number of episodes to 1000 helps Q-Learning a lot, as evidenced by the below graph.

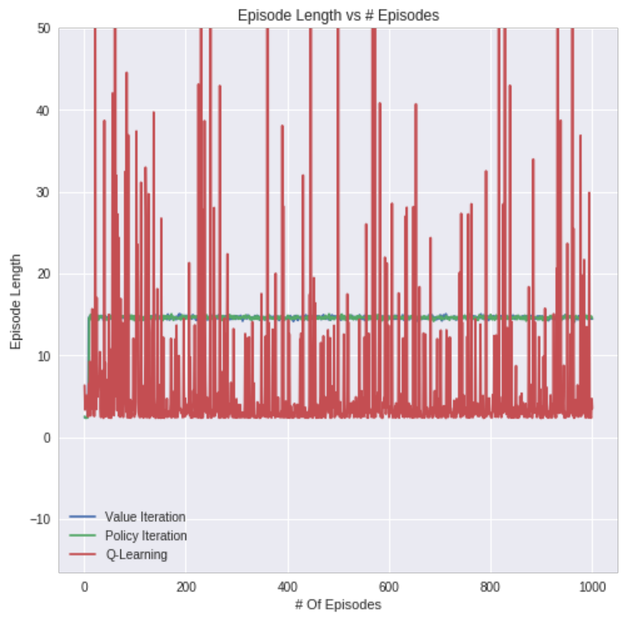


The policy is significantly closer to the optimal one, as obtained by V/P iteration. However, there are still significant issues. For instance, the agent still prefers the low-reward location from the bottom-left area of the map. In addition, in the square immediately between the 1-reward and 20-reward tiles, Q-learning *still produces a policy which chooses the 1-reward tile*! This is of course a completely unjustifiable choice, and is evidence of the fact that even more training episodes are required to ensure that Q-Learning will produce the optimal policy.

On the face of it, this would appear to show that Q-Learning is much more inefficient than the other two algorithms. However, we should remember that we cannot draw a 1-1 correspondence between Q-Learning episodes and V/P episodes, due to their different definitions. Even 2000 episodes of Q-Learning take approximately the same amount of time as 200 episodes of Value Iteration, and in fact the larger the state space the longer Value Iteration will take (since the complexity of an episode of Value Iteration is O(n^2)). After 20,000 episodes of Q-Learning, the policy finally converges to the optimal one.







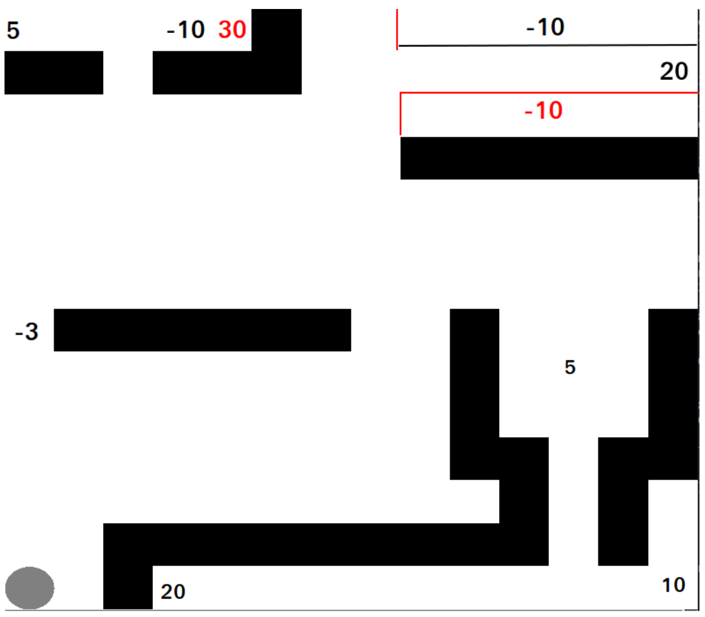
Each datapoint reported is an average of 100 runs of the agent under the calculated policy at each timestep. The most obvious point of note is that the reinforcement learning algorithm is much more unstable than V/P iteration, even after the 100-iteration averaging. The reason for this is that there are several elements of stochasticity in the Q-learning algorithm: which actions you choose from any given step in a trajectory while training, random starting states in training episodes, etc. To allow the noise to "average out" in the end, it is clear that we must train the RL algorithm for many more episodes.

Note that here I only report statistics to episode 1000, and that RL required ~20 times that to achieve the optimal policy: this explains why, in these graphs, RL is outperformed in every metric (except for time) V/I iteration.

Exploration: I tested two different exploration strategies: epsilon-greedy, and softmax. In practice, the policies given by epsilon-greedy were about twice as good (in terms of avg. cumulative episode reward) as those of the softmax (!). The reasoning here is that softmax, by weighting actions according to their expected payoffs, exacerbates the exploitation-vs-exploration issue since the probability of "sub-optimal" actions will go to zero. Especially given the setup here, where denying the proximal reward requires a long sequence of apparently useless actions in order to reach the better rewards, softmax becomes an issue.
I tested epsilon=.01, .1, .2, ..., .8, 9, .99, and found that the algorithm was relatively the same across the first few points, until beginning around .4 (the deterioration associated with randomness dominating your actions is

obvious). I believe that the algorithm was robust to the choice of epsilon in the lower range due to the stochastic nature of the agent's actions, which has the effect of “baking in” some exploration into any policy

Section 2: Hard MDP



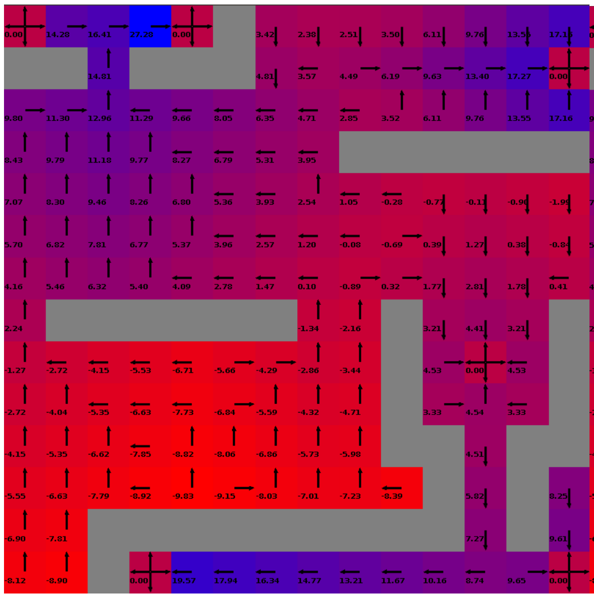
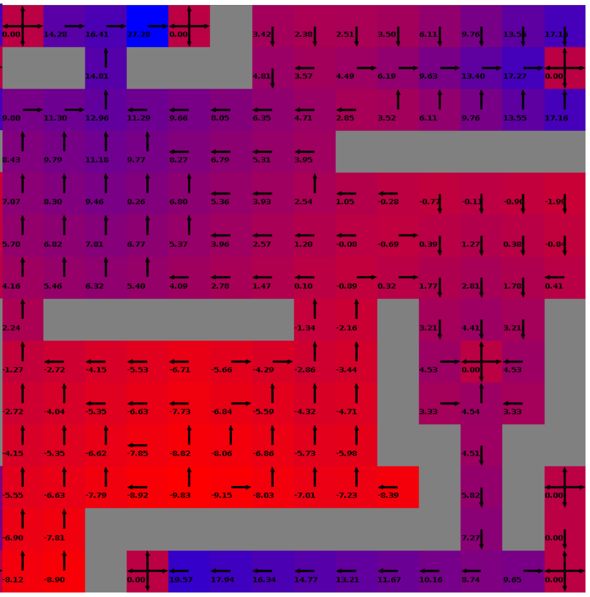
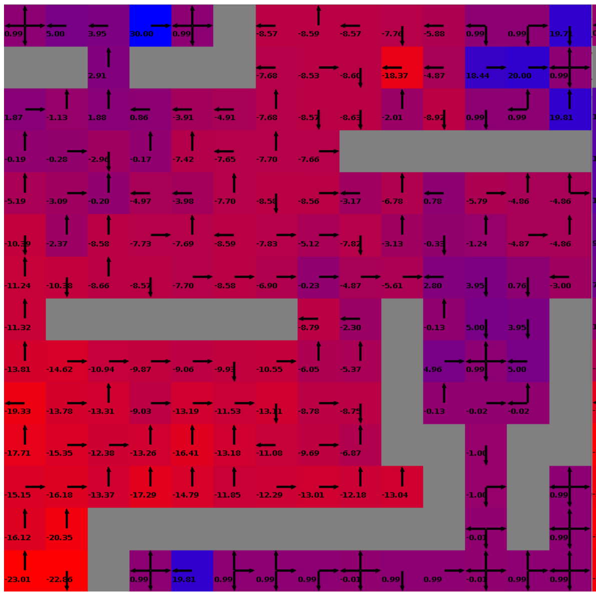
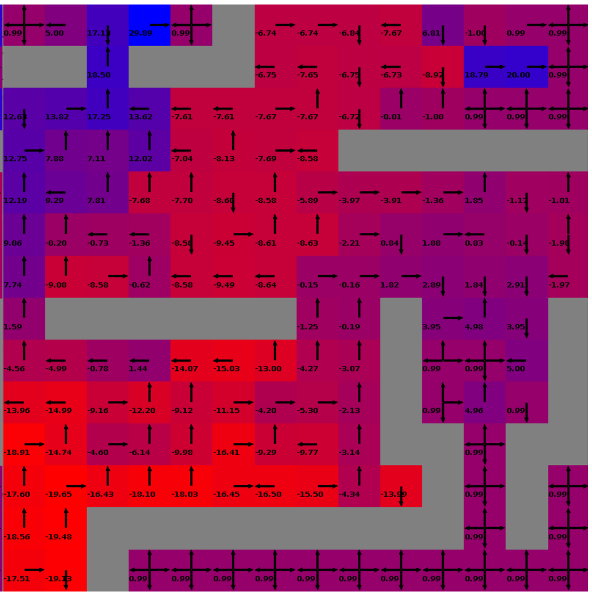
For my “Hard” MDP, I designed a much large gridworld. The “Alive Penalty” and stochastic actions are unchanged from the previous section. Here, the states with positive reward are terminal.

This MDP has a variety of features which make it interesting and a harder challenge than the smaller gridworld. Firstly, the number of states is *much* larger: there are a total of 144 spaces on the grid, approximately 110 of which may be reached by the agent and are non-terminal: an order of magnitude higher than the last.

Furthermore there are quite a few terrain elements of this MDP, most of which revolve around (a) enduring enhanced negative rewards to achieve a better long-term outcome or (b) passing up positive rewards in search of better long-term outcomes. I refer to them as follows

1. “Shortcut”: The wall in the left-middle section of the map, which has a gap with reward -3 (larger than the usual alive penalty of -2). Observe that the wall is long enough that passing through the negative state is actually optimal if you wish to target the top-left corner of the map. If, however, the agent is attempting to reach the right-hand side of the map, then the shortcut is a strictly negative undertaking
2. “Temptation”: Observe the structure in the bottom/bottom-right corner. The agent is funneled into a narrow space where there are two easily-reachable suboptimal rewards which might serve to “addict” a reinforcement learning agent: the optimal policy in that neighborhood is to head for the hills.
3. “Walk of Death”: In the top-right hand corner of the map, the agent may attempt to traverse a narrow bridge to the reward at the end: however the stochasticity of actions means that it will likely fall into the areas of negative reward on the side
4. “The Road Not Taken”: In the top-left corner, the agent must choose whether to pursue an easy small optimal reward or explore past a section of high negative reward in order to discover the high positive reward on the other side

Value Iteration *Policy Iteration*



Q-Learning (100 Episodes) *Q-Learning (1000 Episodes)*


The results obtained on this larger MDP have both important similarities and differences to the ones obtained on smaller one. First of all, the relative time scales of the algorithms remained the same: 1000 episodes of Value Iteration, Policy Iteration, and Q-Learning took 300, 150, and 4 seconds respectively (near-perfect scaling for VI and PI: a factor of 10 increase in state space size leads to a factor of 100 in completion time, exactly in accordance with the fact that both algorithms are O(S^2)).

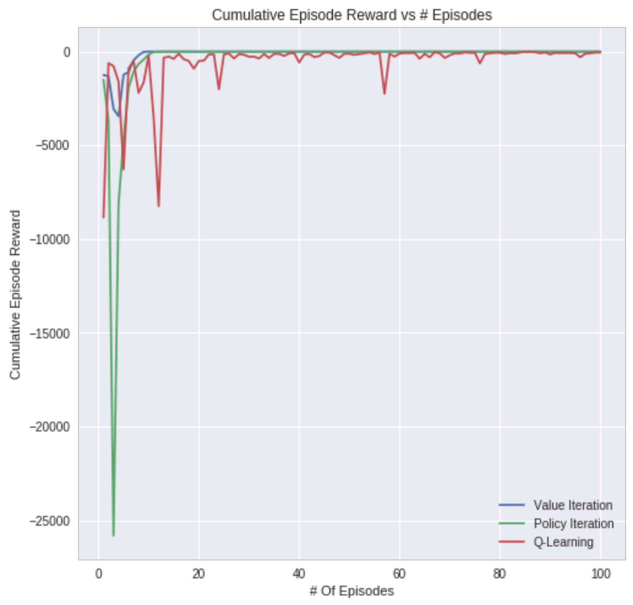
Both VI and PI converge to the optimal policy, as was my hypothesis. Again, we observe that Q-Learning requires many many episodes in order to produce a coherent policy. For instance, if we observe the Q-Learned policy for E=100, then we find that the policy encodes some exceedingly strange (and suboptimal!) strategies.

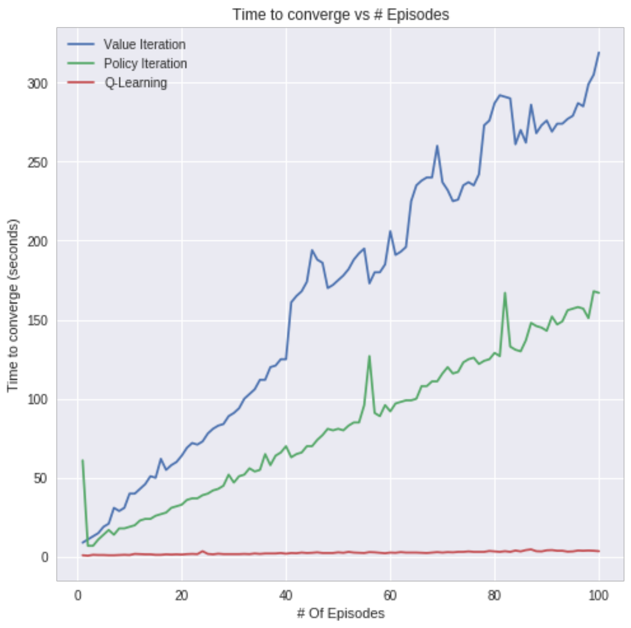
Observe the policy at E=100. At a large scale, it encodes the optimal strategy: from the bottom-left corner, attempt to reach the bigger reward “The Road Not Taken.” And the policy does indeed attempt to take the “Shortcut” as is optimal (the fact that these strategies are optimal is verified by Value/Policy iteration). However the policy does not take the straight-line path from the initial state to the “Shortcut”: and in fact once it has cleared the shortcut, it attempts to move East-West by trying to exploit the stochasticity by repeatedly moving in the North-South direction instead of just making the obvious move.

We do see that more achievements alleviates this problem somewhat. After 1000 episodes, Q-Learning does in fact learn to move sideways with intention rather than as a side effect of another move. However, it still maintains the convoluted path from the initial state to the Shortcut.

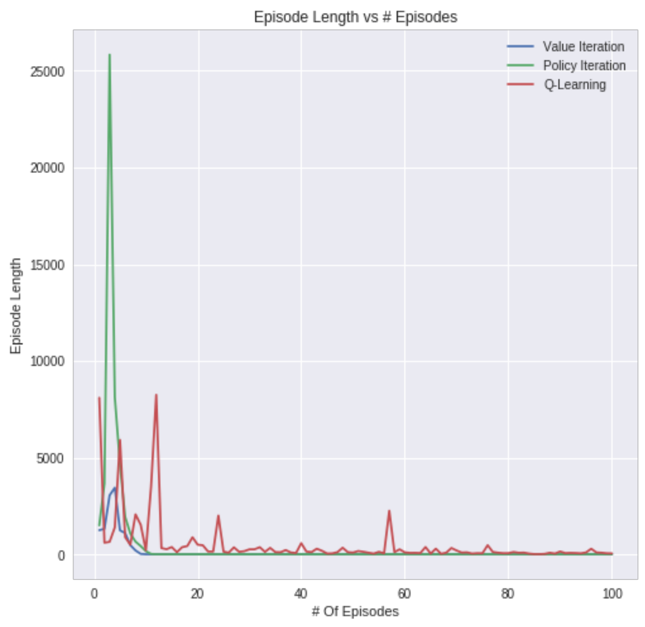
Such a frankly idiotic policy demonstrates the immense struggles that reinforcement-learning algorithms have with exploration vs exploitation: it is likely that the idiosyncrasies of the Q-Learned policy stem from just a few unlucky rolls in the training phase, from which the agent extrapolated incorrect strategies. But yet, even calling the policy “idiotic” is unfair to Q-Learning algorithm, for the algorithm has absolutely no concept of the layout or transition dynamics as V/P iteration or even we can grasp. From its perspective, it shuffling around blind in a possibly infinite world which it has no chance of understanding. Casting the challenge of exploration vs exploitation in this light provides context for how difficult the problem of reinforcement learning truly is.

Therefore, the exploration mechanism in RL algorithms is especially important. Again I tested epsilon-greedy vs. softmax selection, and found epsilon-greedy (with epsilon ~0.4) outperformed softmax selection (which, as explained before, does not seem to adequately balance EE dilemma). The higher epsilon of optimality is due to the fact that the gaps between “interesting” features of the MDP are larger than in the small one: therefore to find them, the algorithm must explore more in any given number of episodes.





With respect to Value and Policy iterations, the graphs here largely tell the same story as in the smaller MDP:



both algorithms converge quickly to the optimal solution. However the data for Q-Learning is markedly different. For one, the lines are nowhere near as unstable as they were in the smaller MDP (though it should be noted that some instabilities do remain, as illustrated via the spikes in the red lines). However, in general, results seem better than in the small MDP.

My explanation here is that in this MDP, Q-Learning actually has less barriers to finding the optimal solution. Recall that the proximal reward in the small MDP *heavily* confused the algorithm, which really zeroed in on exploiting that single tile. Here, there is no analogous issue. All the reward states are far away from the initial state, so at least some exploration is guaranteed. And in fact, the optimal goal state happens to be the closest one to the initial state! Therefore even random exploration will tend to settle on the optimal solution.

In fact, the fact that Q-Learning finds the optimal strategy (if not policy) in this MDP is a bit of an illusion. Given what we know of Reinforcement Learning issues with exploration vs exploitation, it would be more accurate to say that Q-Learning is simply settling on the first policy it finds which gives any reward at all, and that policy happens to be the optimal one. The real advantages of RL approaches are when the state-space is huge, making iteration methods infeasible or even impossible (for instance, as on any continuous domain). On a much larger MDP with these characteristics, the advantages of RL would be much more apparent and we would observe stronger differentiation between the Value/Policy iteration and RL performances.