**Abstract**

Q-learning requires a value for all state-action pairs. This is difficult in partially observable environments since we don’t know the state. Instead, we maintain a belief state, which is constructed from past observations, as well as actions previously taken. There are various strategies for representing the belief state, such as maintaining a finite-length history of observations. Our objective is to analyze how different belief state representations affect whether or not the optimal policy can be learned. We will be using the *TigerEnv* environment, which we created to represent the Tiger Problem (Kaelbling, Littman, & Cassandra, 1998).

**Q-learning overview**

Q-Learning is a model-free reinforcement learning algorithm. The goal of Q-Learning is to develop a policy which tells the agent what actions to take under each set of circumstances. It is referred to as model-free because it does not require a model of the world in order to learn. In other words, the algorithm is completely clueless as to what’s going on in the world around it. The “Q” refers to the function that maps states (circumstances) to actions, and is sometimes said to stand for “quality” of taking an action in a particular state.

There are multiple ways of updating the Q values. We will use the TD0 (“tee-dee-zero”) method. In TD0, the q value update is defined as:

The intuition is to update Q with the “target” or “true” Q value, which is estimated by the *next* Q value, . Since the *next* Q value is also unknown, and is estimated using subsequent value values, this is clearly a dynamic programming problem.

**Tiger Env. Overview**

The Tiger problem is a well-known toy problem often used when analyzing POMDP policies. It is defined as follows

* A tiger is randomly placed behind one of two closed doors (referred to as Left door and Right door). Gold is placed behind the other door.
* You (the agent) need to decide which door to open. Your options are the following: Open Left door, Open Right door, Listen.
* If you open the door with the tiger, the tiger will attack you, and you’ll receive a negative reward of -100. On the other hand, if you open the door with the gold, you’ll receive a reward of +10. Instead of opening a door right away, you also have the option to wait and listen for any noise the tiger makes, in order to be more certain where the tiger is. But… there’s a catch. Listening isn’t free (-1 reward), nor is it 100% accurate. There’s a 15% chance that you may hear the tiger behind the left door, when it’s actually behind the right door (and vice versa).
* When a door is opened, the game restarts and the tiger and gold are randomly placed behind the doors.

It is important to note that the observation function is heavily dependent on the action taken. This will limit our q-learners to certain types of architectures, as we will see later.

**Introduce State representations for evaluation**

There will be three different representations of how we represent the belief state. They are

* ObsSingle – the state is represented by the most recent observation only.
* ObsSeq – the state is represented by the *k* most recent observations. Note that ObsSingle is the same as ObsSeq when *k* is one.
* ObsActSeq – the state is represented by the *k* most recent observation-action pairs.

**ObsSingle**

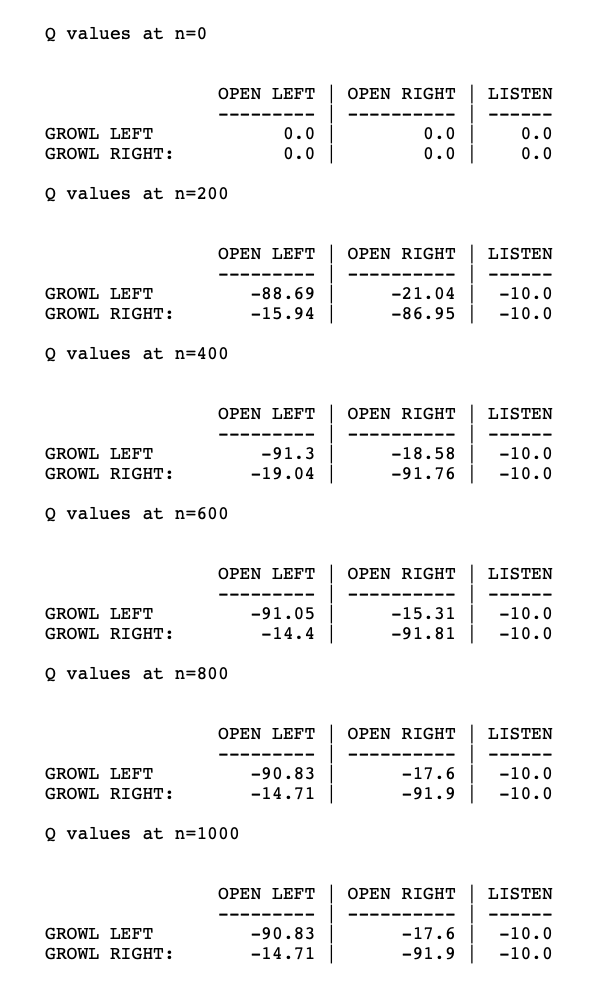
The Q update is defined as

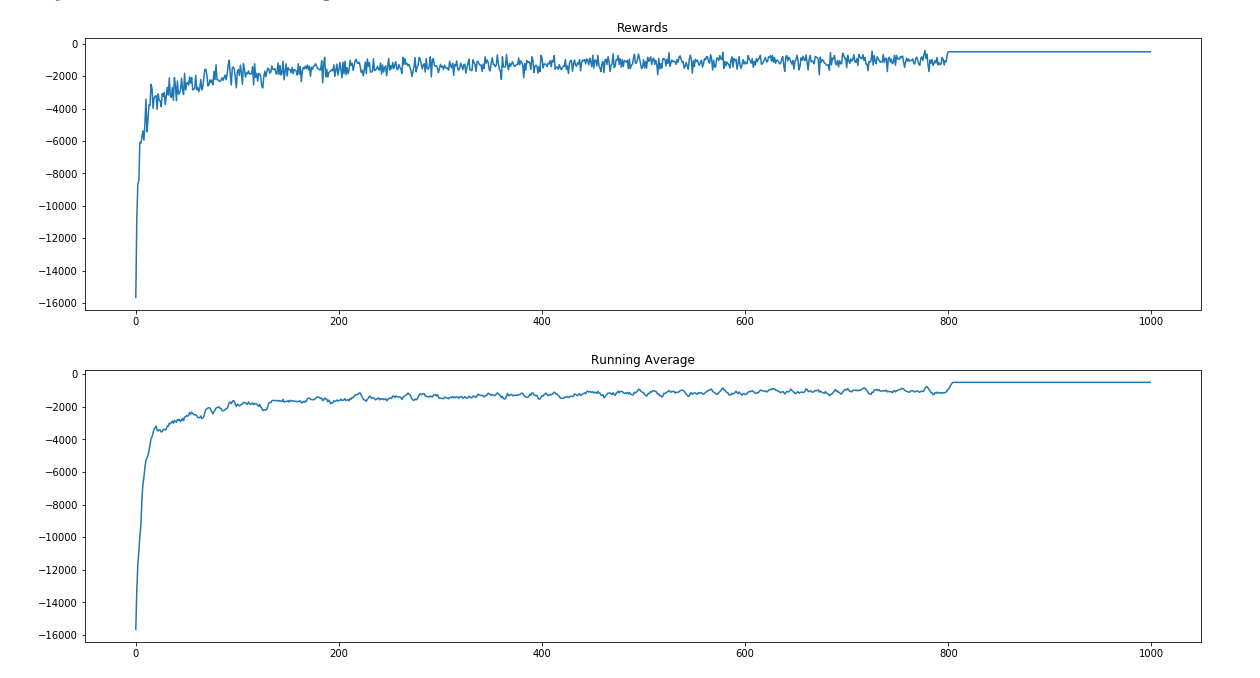


We played 500 episodes with the following environment and model parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter type** | **Parameter description** | **Parameter Value** |
| Environment | Reward for opening door with Tiger | -100 |
| Environment | Reward for opening door with Gold | 10 |
| Environment | Reward for listening | -1 |
| Environment | Episode time horizon | 500 |
| Model | Gamma | .9 |
| Model | Learning rate (alpha decay) | .4 |

As we play more and more episodes, we should expect the Q values to iterate towards their true values. We should also see the policy learn the optimal actions for each state – the action with the highest Q value.





As expected, the Q values drift towards -10 for each of the listen actions. TODO why?

We see that the Q-learner does not learn the optimal policy, which makes sense since a single observation is not enough information to determine if it is best to open a door or not. Instead, the agent is never confident that a tiger is behind either door, and so it takes the “least worst” action, which is to listen.

For example, let’s say that the previous observation was Growl Left. We know that the reward for listening is always -1, so the expected reward of listening is

Similarly, we can compute the expected reward of opening the left door as

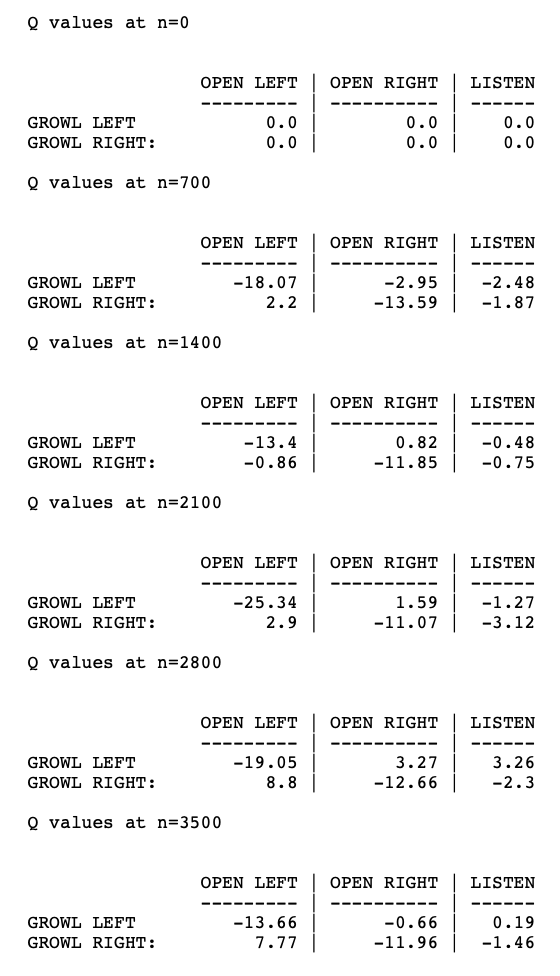
Since the probability of TL is actually dependent on the previous observation *and* the previous action (At-1), we can write P(TL|GL) as

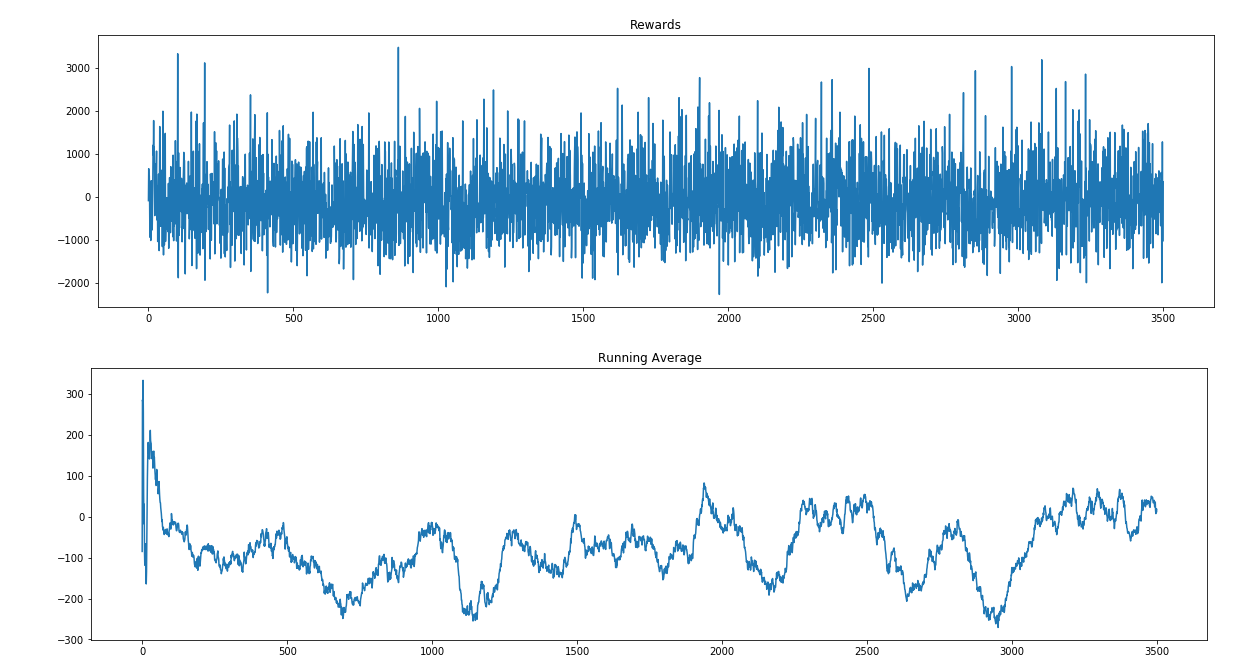
We can write P(TR|GL) similarly.

Then we can rewrite R\_exp(OR|GL) as

So we plugin Acc\_obs=.85, RG=10, RT=-100, P(AL)=1/3, P(AOL=1/3), P(AOR)=1/3, this would be the case where have no idea what the previous action was, then we would get an expected reward of -32.167. If we were confident that we opened one of the two doors, this would actually decrease our expected value, since it ensures that our observation is meaningless. In this case, the expected value would be -45. The best-case scenario is when we know that the previous action was a Listen action, and the expected value is -6.5. If we do this same analysis for the expected reward of (OR|GL), we’d see the same results. Therefore, given the current setup, it is always best to take the Listen action. As we can see, this is what the policy learned to do.

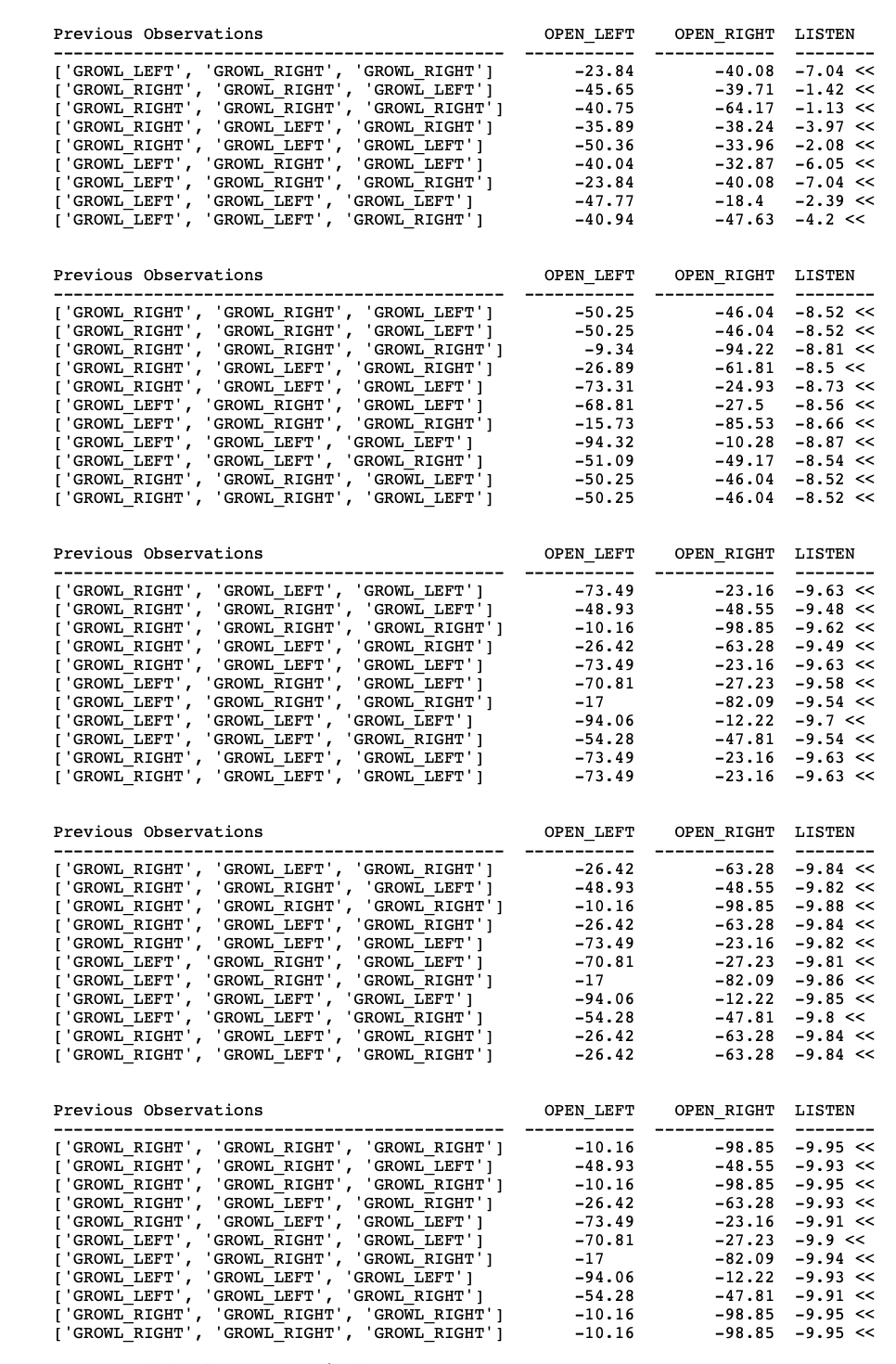
One interesting thing that happens, however, happens when we change the reward function to be -52 for the Tiger, and +48 for the Gold. We actually see the policy *oscillate* around opening left or choosing to listen.

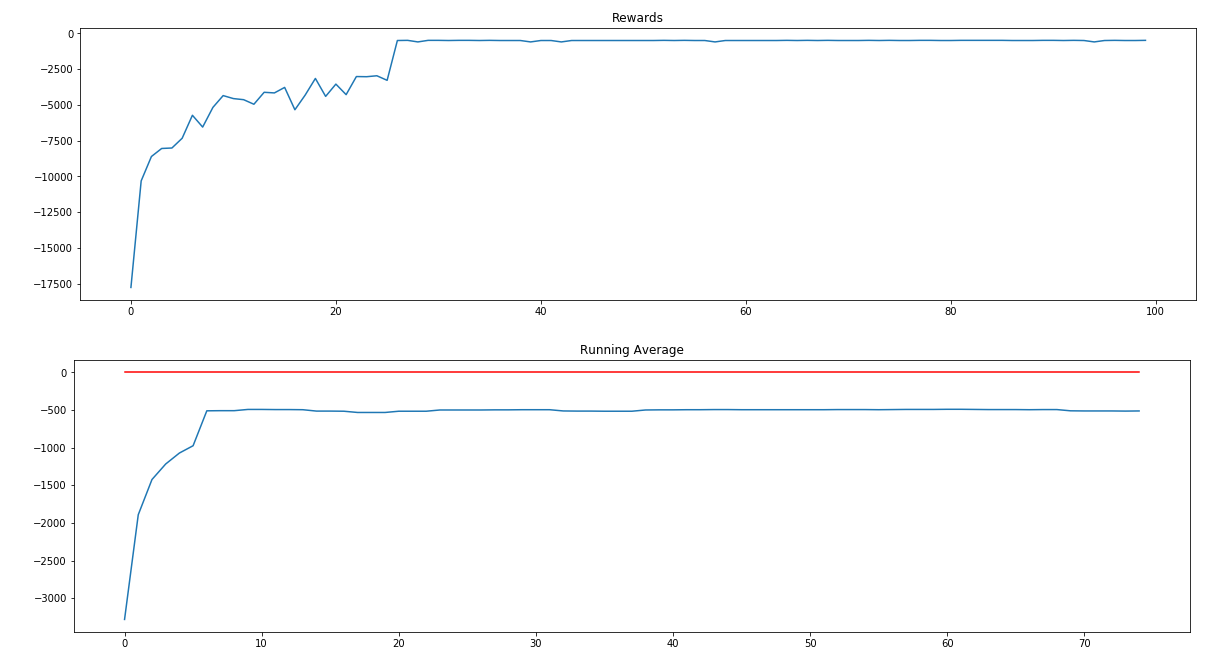




Intuitively, we can imagine why this might happen. The policy learns to always listen, as we mentioned before. Previously, when we computed the expected reward for OL|GL, we saw that even when we *know* that the previous action was Listen, the best we can expect is -6.5 However, let’s say that we change the Tiger reward to be -52, and the Gold reward to be 48. Then, when we know the previous action was Listen, our expected reward becomes 33. Therefore, the *best* action, *given* that the policy *only* listens, is to open left/right on GR/GL. Therefore, assuming epsilon greedy is used, the Q-values of OL/OR will slowly migrate towards being larger than L. Once OL/OR surpasses L, things change. At this point, the best action is now to Listen. This is because we can no longer assume that the previous action was Listen. In fact, we can assume that the previous action was OL/OR, which resets the location of the tiger and produces an uninformative observation. Because of this phenomenon, we see the policy oscillate back and forth between choosing Listen as the best action, and OL/OR.

**ObsSeq**





Similar to the ObsSingle, observations alone are not enough to represent the true state. Therefore, the final policy is suboptimal, and only learns to Listen.

**ObsActSeq**

* Show progression of Q values
* Show converged policy
* Show why this makes sense.
* Show all non-listen best actions
* For 4 seq, do we see any non-listen best actions for <L-GL, L-GR, L-GR, L-GR>?
  + At what sequence length do we start to see this?

**Conclusion**

The Tiger Environment is a tricky problem because the same observation can result from different states, since the observation is dependent on the action taken by the agent. Therefore, if the agent is going to represent its belief state using a history, it is not merely enough to keep track of each of the observations. The agent must also keep track of its actions as well. This hypothesis was confirmed in our results, namely, that the QLearnerObsActSeq was the only algorithm capable of learning the optimal policy. The issue with this algorithm, however, is that the size of the Q-function increases (exponentially?). It is possible to mitigate the size issue, by approximating the state representation. For example, we could approximate the Q function using a deep neural network. In fact, this is exactly what Deep Q Learners (DQNs) do.

**Possible TODOs**

* Compute how quickly each algorithm converges on optimal policy
* Experiment with other P.O. environments besides Tiger Problem.
* Lookup if there are other similar papers on this topic.

**Future work**

A good next step would be to perform the same analysis using a DQN which approximates the state space of past action-observation pairs (insert citation),