A Comparison of Exploration/Exploitation Techniques for a Q-Learning Agent in the Wumpus World

A. Friesen
Department of Computer Science
afriesen@cs.washington.edu

Abstract

The Q-Learning algorithm, suggested by Watkins [1], has become one of the most popular reinforcement learning algorithms due to its relatively simple implementation and the complexity reduction gained by the use of a model-free method. However, Q-Learning does not specify how to trade off exploration of the world for exploitation of the developed policy. Multiple such tradeoffs are possible and preference of one over the other should depend mainly on whether a fast, but less accurate convergence to a policy is desired or whether a slower convergence to a more accurate policy is better. This paper will present the results of several exploration vs. exploitation (EE) methods within the contrived environment of the Wumpus World [2].

1. Wumpus World

The Wumpus World is a simple grid-world environment that can contain multiple hazards (in the form of pits and the wumpus itself) and a goal. The hazard and goal reward values are fairly arbitrary, but the values chosen were -100 and 100, respectively. Some experimentation was done with unbalanced hazard and goal rewards (i.e. -10000 for a hazard and +100 for a goal) but this did not significantly affect the results. In the actual Wumpus World, the agent can move in any of the four directions (North, South, East, and West) and can shoot a single arrow in one of these directions to attempt to eliminate the wumpus. The arrow action was removed from the version used in these tests as it added too much complexity. Thus the wumpus functions like a pit, in that if the agent moves onto the wumpus square then the agent is killed.

However, to add some complexity to the world, the agent's actions were altered to be non-deterministic. Thus, if the agent attempts to move East, there is a certain probability that it will actually move South instead. Some experimentation was done with the different possible combinations of transition probabilities. More random transitions lowered the agent's average score, while more deterministic transitions had the opposite effect. Furthermore, more random transitions caused the agent to play more cautiously by moving as far away from hazards as possible so as to lessen the probability that it would accidentally be killed.

2. Q-Learning Algorithm

The Q-Learning algorithm used was taken from Russell and Norvig ([2]) and just uses the standard Q-update function

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s))$$

Where *Q(a, s)* is the Q-value for the current state and action, *R(s)* is the reward of the current state, α is the learning rate (determines to what extent newly acquired information overrides old information), and γ is the discount factor (determines the importance of future rewards).

One benefit of Q-Learning is that the transition function $T(s,a,s')$ does not have to be learned; instead, all that the agent is required to compute is a single value for every state, action pair. Thus, the required memory is only $O(|S| \times |A|)$ instead of $O(|S|^2 \times |A|)$. A second benefit is that learning is performed online. Thus, the agent is constantly updating its Q-values as it explores the environment and can immediately begin using these estimates to determine which actions to take (the agent can explore by performing actions that have been executed fewer times in order to obtain more accurate Q-values, or it can exploit by choosing the highest Q-value actions to safely traverse the environment as soon as possible). The four exploration/exploitation strategies that were investigated each use different methods of separating exploration and exploitation.

In addition to the Q-learning equation and the exploration/exploitation strategies, the remainder of the code is merely bookkeeping and world and state manipulation. A wumpus world is created that contains one wumpus, multiple pits, and the goal. The starting position is always at *(0, 0)* in the grid and the goal is always *(num_rows – 1, num_cols – 1)*. The constant start and goal states were chosen because this is the maximum distance that is required to be traversed by the agent to get to the goal, and thus the most learning is required. A random start state would not add insight into the learner, but would instead only add difficulty

in comparing and evaluating the different learning methods. Additionally, many problems have a static start state and thus a random start state does not make sense in that context. However, the implementation could be very easily altered to use a random start state. For similar reasons, a fixed world was used for all of the tests performed (see results section below).

2.1. Random Exploration

The first exploration/exploitation (EE) technique that was investigated was also the simplest; purely random exploration. For random exploration, whenever the agent needs to move, an action is chosen completely arbitrarily, with no consideration of the current Q-values. The benefits of this approach are that the action-choosing algorithm is extremely fast and simple and the state-space gets explored more fairly than by the other EE methods. Unfortunately, the agent dies during most trials as the probability of running into a hazard is much higher than that of reaching the goal. This could be very costly for a real-world robot. Additionally, because the state space is investigated evenly, areas that the agent would never enter (i.e. the hazards) are explored equally as much as high-reward areas. This is highly inefficient as once the agent knows an area is bad it should instead focus its resources on more promising areas. Random exploration essentially completely separates the exploration from the exploitation and no exploitation is ever done while the agent is exploring.

2.2. Global temperature

To attempt to alleviate some of the problems that are associated with fully random exploration, a technique was developed that uses a global "temperature" to determine how much randomness the agent should use in its decision making (very similar to simulated annealing in local search algorithms). The temperature starts high, making the exploration very random at the beginning, and slowly drops as the number of iterations increases, allowing the agent to converge on a specific policy. The chosen implementation relates the temperature to the total number of actions that the agent has performed.

This EE technique allows the designer to determine how quickly the agent converges to a policy. The tradeoff here, however, is that the faster an agent converges to a policy, the less likely it is to be the optimal policy (especially in a non-deterministic environment). Thus, a careful selection of the parameters that dictate the temperature, its rate of change, and its effect on the randomness of the actions is required and can seem somewhat arbitrary.

While a global temperature allows the agent to avoid the hazards after fewer iterations than pure random exploration, it can leave rewarding portions of the state space relatively unexplored. As the temperature drops, the randomness in the exploration decreases, and thus any sections of the state space that have not already been explored are less likely to be explored as the temperature drops further. This is especially noticeable for paths that are far away from the start state. The probability of reaching these is lower and thus they will not be explored as fully. The next section presents a technique designed with this in mind.

2.3. Local Temperature

By adding some statistics to the algorithm, it is possible to keep track of the number of times each action has been performed in each state. From these statistics, those actions that have been performed the fewest number of times can be prioritized. The simplest way to achieve this is to choose an "explore count" constant and choose those actions that have not been explored this many times [2]. Once all the actions in a state have been fully explored, the highest Q-value action can be chosen every time, or the Q-values can be used to determine the probability of choosing an action. Only the first of these techniques was implemented and tested but the second might provide a better solution. Once again, this EE technique requires the implementer to experiment with various parameters to achieve the best possible results.

2.4. Boltzmann Exploration

The final EE technique that was investigated was Boltzmann Exploration (BE) [3]. In BE, the probability of picking an action is given by:

$$\frac{e^{Q(a,s)/\tau}}{\sum_a e^{Q(a,s)/\tau}}$$

Where *Q(a,s)* is the Q-value of the action in the state, and τ is the temperature. When τ is large, all actions have approximately the same probability; when τ is small, actions will be chosen proportionally according to their estimated value. Once again, the exploration/exploitation balance can be controlled by altering the temperature. Thus, the initial temperature and the rate of change of the temperature are the two important parameters.

This is the most complicated technique but, depending on how the temperature is regulated, it should provide a good tradeoff between exploration and exploitation: initially starting quite random and quickly biasing the probabilities to further explore only those areas of the state space that seem promising.

3. Results

3.1. World Setup

To test the exploration/exploitation techniques and the Q-Learning algorithm, the following methodology was used. For each parameter variation of each EE technique, 10 complete runs were performed. Each run involved placing the agent in the start state 1000 times and allowing it to traverse the world based on its EE policy until it encountered a hazard or the goal, updating Q-values after each action. Every 25 iterations (per run) a policy was generated from the current Q-values and that policy was run 1000 times within the world state and the average reward per run was returned. These values, averaged over the 10 runs, are what are plotted in the following figures.

The world that was used for the majority of testing had 5 rows, 5 columns, 2 pits, and the single wumpus. The start state was at *(0, 0)*, the goal was at *(4, 4)*, the pits were at *(2, 0)* and *(2, 2)*, and the wumpus was placed at *(3, 2)*.

Start				
P		P		
		W		
				Goal

Figure 1: The main Wumpus World state

The transition probabilities that were used for testing had *P(action works) = 0.70*, *P(action goes left) = 0.15*, and *P(action goes right) = 0.15*. The option of having a probability of reversing the action (i.e. going North when trying to go South) was tested but it only served to lower the average earned reward and did not significantly alter the derived policies unless the probability became unreasonably high. This is most likely because, in this case, no matter what action the agent chooses it can end up going in any direction and thus it must choose the action that will hopefully work.

3.2. Algorithm Parameters

For the Q-Learning algorithm, a value of 0.85 was chosen for gamma (the discount factor). This value was chosen because it gave the highest average reward compared to higher and lower gamma choices. This is understandable because it is a good balance between preferring longer-term reward (gamma close to 1) and short-term reward (gamma close to 0).

The global temperature EE policy was designed to only require a single parameter: the amount of randomness that is removed from future choices after each action. This was called *num_action_scaler* as it scales the number of actions which are set as the direct inverse of the temperature. Three different values were tested (see Figure 2): 100, 1, and 0.01. Higher values slow the temperature decrease rate. Not very surprisingly, the technique that decreased the temperature fastest also converged fastest, but the policies that it converged to were not as accurate as those generated with a slower decreasing temperature.

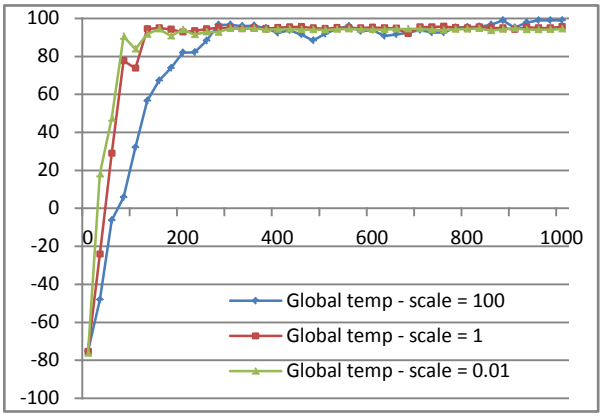

Figure 2: Average reward for the Global Temperature EE Policy over 1000 iterations (averaged over 10 runs)

The local temperature EE policy has two parameters, only one of which gets altered. The parameter that is kept constant is the optimistically estimated reward that any state can achieve at the current time. This is set to 200 (the maximum possible reward is 100). This ensures that all state, action pairs that have not been explored enough times will be prioritized above everything else. The second parameter is the number of times that a state, action pair must be explored. Three different values were tested: 5, 25, 70. Surprisingly, the value that caused the most random exploration (*N_e = 70*) resulted in the least accurate policies, and yet converged almost as quickly as the other two.

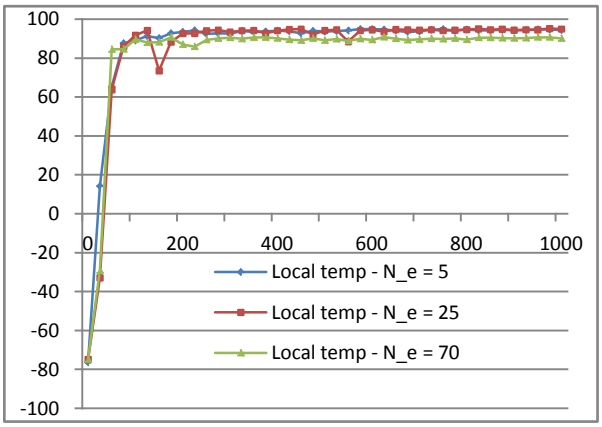

Figure 3: Average reward for the Local Temperature EE Policy over 1000 iterations (averaged over 10 runs)

The Boltzmann Exploration policy also has two parameters, only one of which gets altered. The two parameters in BE are the initial temperature and the temperature decrease rate. Together, these also determine how long the agent explores before starting to exploit its knowledge of the environment. Adjusting each parameter has similar effects, so only the temperature decrease rate was adjusted. Again, three different values were used: 0.01, 0.25, and 1.0. This value is subtracted from the current temperature (to a minimum temperature of 1.0) after each of the 1000

iterations used to learn a policy. The Boltzmann EE plots are as expected. For a slower temperature decrease rate, the policy converges slower. This can be attributed to the increased randomness and the lack of early exploitation. An interesting note is that all three variations converge to the same accurate policy and there does not seem to be any benefit for lowering the temperature slower. It would be interesting to test further with a faster temperature decrease rate.

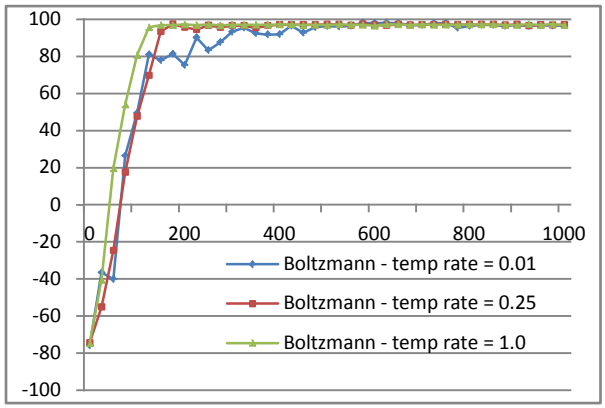

Figure 4: Average reward for the Boltzmann EE Policy over 1000 iterations (averaged over 10 runs)

Finally, the four EE policies are compared in Figure 5. As expected, random EE takes the longest to converge but eventually becomes quite accurate. BE converges to the most accurate policy but does not converge as quickly as the local and global temperature policies. This most likely occurs because BE causes the agent to equally likely pick from similar Q-value actions and thus explore them more thoroughly to determine which is more rewarding, whereas global temperature EE just picks randomly if the temperature is high enough and local temperature EE just explores each action a fixed number of times and then takes the ideal policy.

Not very surprisingly, local temperature outperforms the global temperature policy. Somewhat surprisingly though, it also outperforms Boltzmann Exploration for convergence. It would be interesting to see if BE outperforms local temperature if BE were to use a faster temperature drop rate.

To ensure that these results were valid, the same tests were run on a second world which was the first world with an extra pit in the top right corner (square *(0, 4)*). This had the expected effects of slowing convergence and lowering the overall accuracy of all of the algorithms (Figure 6). Otherwise, the results were consistent with Figure 5.

7. Conclusion and Future Work

Four distinct exploration/exploitation methods were presented and compared in this paper. It was shown that, for the limited subset of parameter combinations tested, the local temperature policy performs the best on average, considering both convergence time and policy accuracy. However, Boltzmann exploration looks promising with different parameters. The global temperature policy looks like it is dominated by the local temperature policy; however, more statistics are required for the local temperature policy so global temperature will perform sufficiently if there are other limitations.

An interesting next step would be to compare Q-learning to a model-based reinforcement learning algorithm and then to add Dyna (planning capabilities) to both the Q-learner and the model-based learner. Finally, developing a POMDP solution that uses the stench and breeze percepts would be extremely instructive.

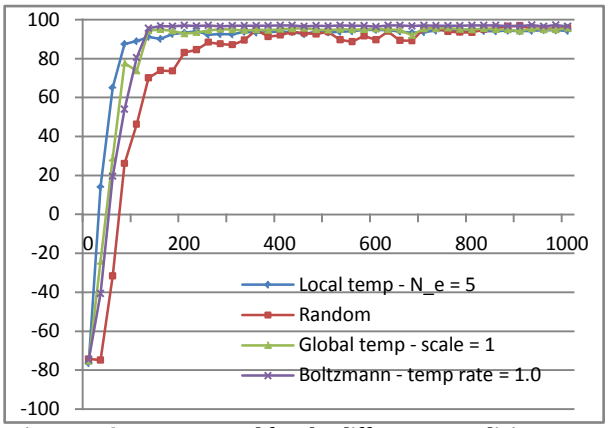

Figure 5: Average reward for the different EE policies over 1000 iterations (averaged over 10 runs) for the first world

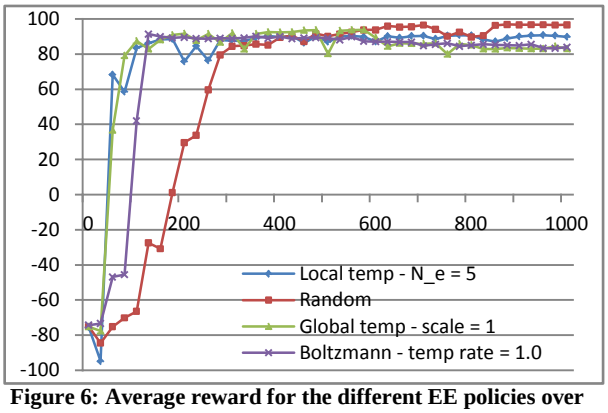

Figure 6: Average reward for the different EE policies over 1000 iterations (averaged over 10 runs) for the second world

8. References

[1] C. Watkins, P. Dayan, *Machine Learning 8*, Kluwer Academic Publishers, Boston, 1992.

[2] S. Russell, P. Norvig., *Artificial Intelligence: A Modern Approach*, Pearson Education, Inc., New Jersey, 2003.

[3] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming", in *Proceedings of the Seventh International Conference on Machine Learning*, Morgan Kaufman: San Mateo, CA, pp. 216-224, 1990.