ME 537 Homework 3

Mitch Colby

November 4, 2010

1 N-Armed Bandit

1.1 Problem Statement

The n-armed bandit problem is a model of a slot machine with multiple levers. When any lever is pulled, it provides a reward obtained from a distribution associated with that lever. Initially, the reward distribution about the levers is unknown. Therefore, the n-armed bandit involves the tradeoff between exploiting the lever with the highest expected reward and exploring other levers in order to gain a better estimate of the lever reward distributions. In this particular case of the n-armed bandit, there are five levers; each lever has a reward distribution which is a Gaussian distribution with a known mean and variance (figure 1). Epsilon-greedy, greedy, and softmax action selection were all compared in this 5-armed bandit domain, using episodes which were 10 or 100 time steps.

Action	Mean Reward	Variance
1	1	5
2	1	1
3	2	1
4	2	1
5	0	10

Figure 1: 5-Armed Bandit Reward Structure

Initially, an expected reward table is initialized to help pick actions. This table may be initialized with all the rewards at zero, an optimistic value (higher than all anticipated mean rewards), or a pessimistic value (lower than all anticipated mean rewards). Every time a lever is pulled and a reward is received, the corresponding expected reward in the table is updated using:

$$Q_{n+1} = Q_n + \alpha(r_{n+1} - Q_n) \tag{1}$$

Where α is the learning rate, Q_n is the previous value in the expected rewards table, r_{n+1} is the reward received, and Q_{n+1} is the updated value in the expected rewards table. The expected reward table, along with an action selection policy, determines which lever is pulled at each time step.

1.2 Solution Methods

1.2.1 Greedy Action Selection

The greedy action selection policy involves pulling the lever with the highest expected reward at each time step. If the expected reward table is accurate (the expected reward values match the actual reward values), then the greedy action selection policy is the optimal policy. However, if the expected reward table is not accurate, or the rewards of the levers change with time, then the greedy action selection policy is (sometimes severely) suboptimal because this policy does not allow for exploration of actions which don't have the highest expected reward.

1.2.2 ϵ-Greedy Action Selection

The epsilon-greedy action selection policy requires setting a small parameter ϵ which is between 0 and 1 (ϵ is generally less than 0.5). At each time step, the lever with the highest expected reward is pulled with probability (1 - ϵ), or a random lever is pulled with probability ϵ. Thus, the epsilon-greedy action selection policy results in exploiting the best known lever most of the time, but allows for exploration of states which don't have the highest expected reward.

1.2.3 Softmax Action Selection

The softmax action selection policy involves probabilistically choosing which lever to pull at each time step. If Q_t is the expected reward table at time t, then the probability of choosing action i is given by:

$$p_t(i) = \frac{e^{\frac{Q_t(i)}{\tau}}}{\sum_j e^{\frac{Q_t(j)}{\tau}}} \tag{2}$$

Where τ is a non-negative parameter called the temperature. If the temperature approaches infinity, then each action will be chosen with equal probability. If the temperature approaches zero, then the action with the highest expected reward is chosen with probability 1. The softmax policy is similar to the epsilon-greedy policy in that it explores states which

don't have the highest expected reward, but it has a significant advantage over the epsilon-greedy policy in some domains. If the domain has actions which are extremely undesirable, these actions have a much higher probability of being taken with an epsilon-greedy policy than with the softmax policy. The softmax policy encourages exploration, but the exploration is more intelligent than the random exploration associated with the epsilon-greedy policy.

1.3 Results

The analysis parameters for the 5-armed bandit problem are listed in figure 2. Figure 3 shows the performance of the epsilon-greedy, greedy, and softmax policies for pessimistic, optimistic, and zero reward initialization over 10 time steps. Figure 4 shows the same comparison, but over 100 time steps.

Parameter	Value(s)
α	0.25
E (epsilon-greedy)	0.1
Time Steps	10; 100
Initial Expected Reward Values	-2; 0; 2
Statistical Runs	1000
Temperature (softmax)	0.7

Figure 2: Parameter Settings for 5-Armed Bandit Analysis

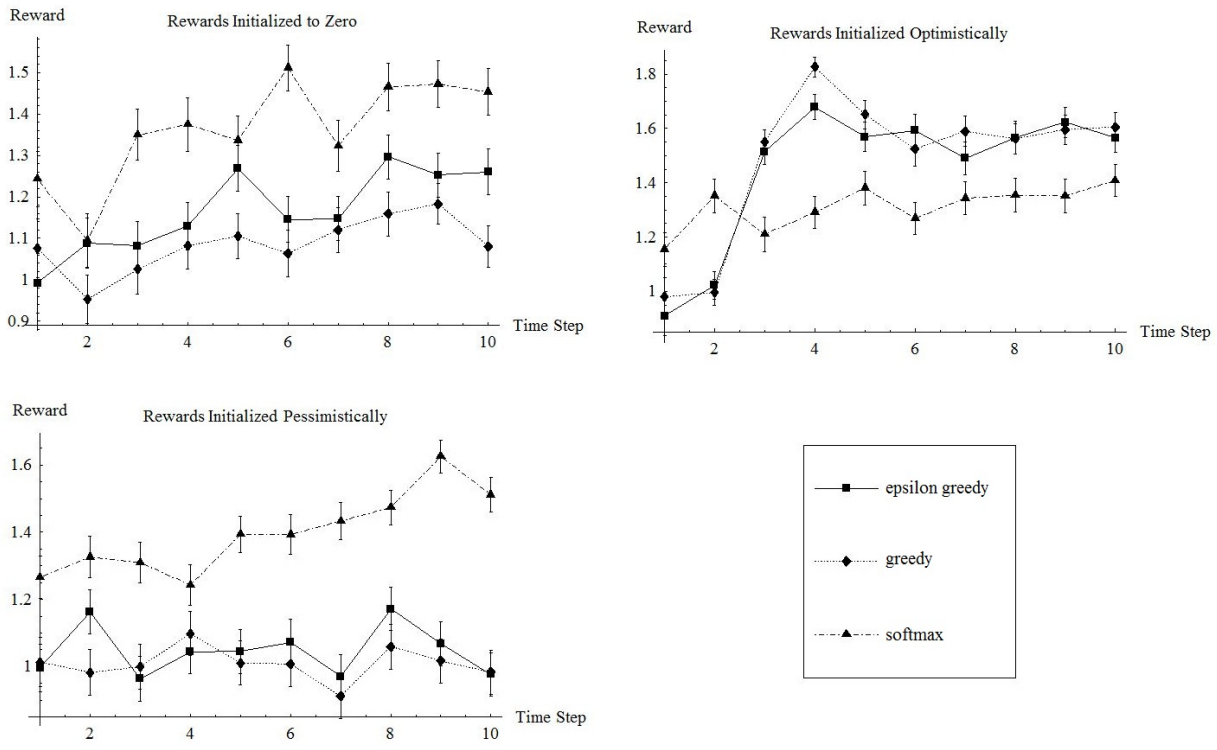

Figure 3: 5-Armed Bandit, 10 Time Steps - comparison of optimistic, pessimistic, and zero reward initialization

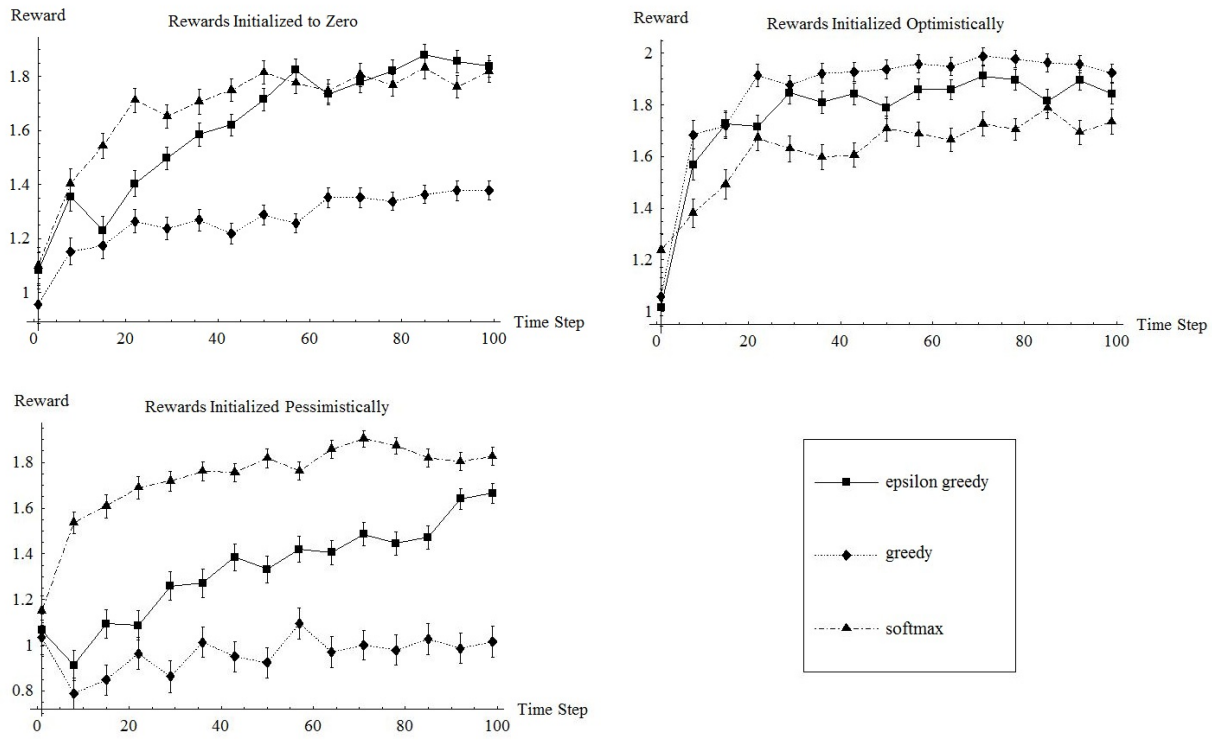

Figure 4: 5-Armed Bandit, 100 Time Steps - comparison of optimistic, pessimistic, and zero reward initialization

1.4 Discussion

1.4.1 Expected Rewards Initialized Optimistically

Initializing the expected rewards to an optimistic value (higher than any actual value received) encourages exploration early on in the search process. This is because as any action is taken, the actual reward is less than the expected reward, which results in the action which was taken having a lower expected reward than the other actions. This process continues until each action has been taken at least once. In the 10 time step episode, the epsilon-greedy and greedy policies performed equally well, because the exploration resulting from the optimistic reward initialization makes both of these policies nearly identical. The softmax policy performed poorly over the 10 time step episode with optimistic reward initialization, because each action had nearly equal probabilities of being taken. In the 100 time step episode, the epsilon-greedy and greedy policies performed equally well, again because of the exploration that was encouraged early in the episode. The softmax policy still didn't perform as well as the epsilon-greedy and greedy policies over the 100 time step episode, but it performed much better with more time steps because the expected reward table was closer to converging to

the actual rewards toward the end of the episode (See figures 3 and 4).

1.4.2 Expected Rewards Initialized Pessimistically

Initializing the expected rewards to a pessimistic value (lower than any actual reward received) encourages exploitation. After the first action is taken, the expected reward for that action becomes higher than any other value in the expected reward table. For the greedy policy, this action is taken for every remaining time step in the episode. For the epsilon-greedy policy, this action will continue to be taken with probability (1 - ϵ). In the 10 time step episode, both the greedy and epsilon-greedy policies perform quite poorly because they are extremely exploitative policies in such a short episode. In the 10 time step episode, the softmax policy outperforms both the epsilon-greedy and greedy policies, because the probabilistic action selection of the softmax policy still promotes exploration of the state space early in the episode. In the 100 time step episode, the greedy algorithm performs extremely poorly, because it continues to choose the first action taken, due to the exploitative nature of the algorithm. However, the epsilon-greedy policy performs much better over 100 time steps than 10 time steps because it has more time to explore the action space in longer episodes. The softmax policy performs best when the expected rewards are initialized pessimistically. Again, this is because the softmax policy gives a greater probability of exploration, which results in the expected reward table being more accurate than with the epsilon-greedy or greedy policies (See figures 3 and 4).

1.4.3 Expected Rewards Initialized to Zero

Initializing the expected rewards to zero results in system performance which lies between optimistic and pessimistic expected reward initialization. The expected rewards initialized at zero is slightly pessimistic, because the lowest average reward for any of the five actions is 0. This initialization encourages exploitation early, but the expected rewards converge more quickly than with true pessimistic reward initialization because the expected rewards are initialized closer to the actual rewards. In the 10 time step episode, the greedy algorithm performed very poorly, because it was exploiting poor rewards rather than exploring the state space to find the optimal action. The epsilon-greedy policy performed better than when the rewards were initialized pessimistically, but still performed poorly due to the exploitative nature of the policy with the rewards initialized to zero. The softmax policy performed best during the 10 time step episode, due to the fact that the softmax policy explores during the early part of the episode, because the actions all have similar expected rewards. In the 100 time step episode, the softmax and epsilon-greedy policies both converge to approximately

the same value; this means that the epsilon-greedy policy didn't have enough time to explore in the 10 step episode, but 100 steps allowed for exploration to increase the epsilon-greedy policy's performance. The greedy policy still performs poorly over 100 time steps; this is simply because initializing the expected rewards to zero encourages the greedy policy to repeat the first action it took for the entire episode, unless a negative reward is received early in the episode (see figures 3 and 4).

1.4.4 Comparison of ϵ-Greedy, Greedy, and Softmax Policies

Over a short time horizon, as in the case of the 10 time step episodes, the greedy algorithm has the potential to perform well because there aren't enough time steps for exploration to pay off by the end of the episode. However, as the time horizon increases, the greedy action selection policy becomes less desirable (compared to the epsilon-greedy and softmax policies) because exploration increases long-term rewards when the time horizon is large. When rewards are initialized pessimistically, the greedy action selection policy breaks down because no exploration takes place. When expected rewards are initialized optimistically, the greedy policy behaves similarly to the epsilon-greedy policy because exploration is forced by the expected reward initialization. In general, over long time horizons both epsilon-greedy and softmax policies are preferable to a greedy policy because exploration is so important in maximizing rewards in a long time horizon. The performance of the softmax policy is extremely sensitive to the selection of temperature. The ideal temperature setting would encourage exploration when the uncertainty surrounding the expected rewards is high, and would encourage exploitation once the expected rewards begin to converge. The performance of epsilon-greedy policies is very sensitive to the choice of ϵ, but this choice is simpler to make than the temperature choice because the effect of ϵ on the algorithm is obvious. Ultimately, exploration is crucial to maximize rewards in the n-armed bandit problem, especially over long time horizons; for this reason, the softmax and ϵ-greedy policy generally perform better than the greedy policy.

2 Gridworld

2.1 Problem Statement

A 5 by 10 gridworld has been created, and the goal state is in position (4, 10) (Figure 5). At the beginning of each episode, the agent starts in a random location in the gridworld. At every time step, the agent has five potential actions: moving left, right, up, down, or staying in place. The agent receives a reward of 100 for reaching the goal state. The problem

involves implementing a learning algorithm to train the agent to find the goal state in the shortest time possible. For any initial state, an optimal route distance is known, and is given in equation 3. The performance metric used is given in equation 4, and gives the ratio of optimal number of actions vs. the number of actions actually taken.

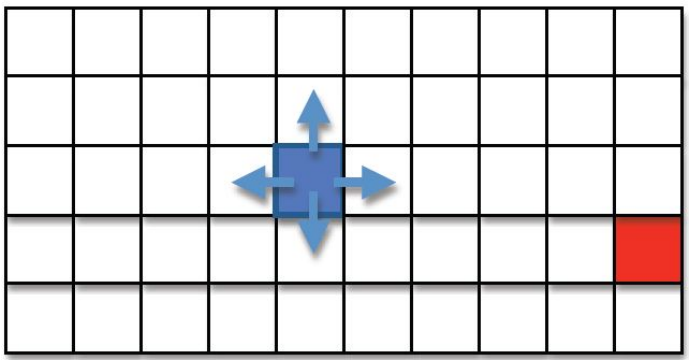

Figure 5: 5 by 10 Gridworld- the agent starts in a random position and must travel to the goal state (lower right)

$$d_{optimal} = |x_{goal} - x_{initial}| + |y_{goal} - y_{initial}| \tag{3}$$

$$Performance\,Metric = \frac{d_{optimal}}{d_{actual}} \tag{4}$$

2.2 Solution Methods

2.2.1 Stateless Action Selection

The first action selection policy is the ϵ greedy policy utilized to solve the n-armed bandit problem. No differentiation between states is made with this policy, so the agent has no knowledge of its position in the gridworld.

2.2.2 Q-Learning

To compare with the stateless action selection, a Q-learning policy is implemented to train the agent to take the shortest path to the goal state. Initially, the agent is placed in a random state. The state in this context is the position of the agent, and the actions are to move in one of four directions or to stay in place. After each action, the agent receives a reward. If the next state is not the goal state, then the agent receives a time penalty of -1. If the next state is the goal state, the agent receives a reward of 100. Once the reward is received, the

Q-table is updated using equation 5. An ϵ-greedy policy is used for action selection in the Q-Learning algorithm.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{5}$$

2.3 Results

The learning parameters used in the gridworld analysis are shown in figure 6. Figure 7 shows the performance of the stateless action selection policy and the Q-Learning policy.

Parameter	Value
ε	0.01
α	0.5
γ	0.5
Episodes	1000
Initial expected reward value	0
Statistical Runs	100

Figure 6: Parameters Used in Gridworld Analysis

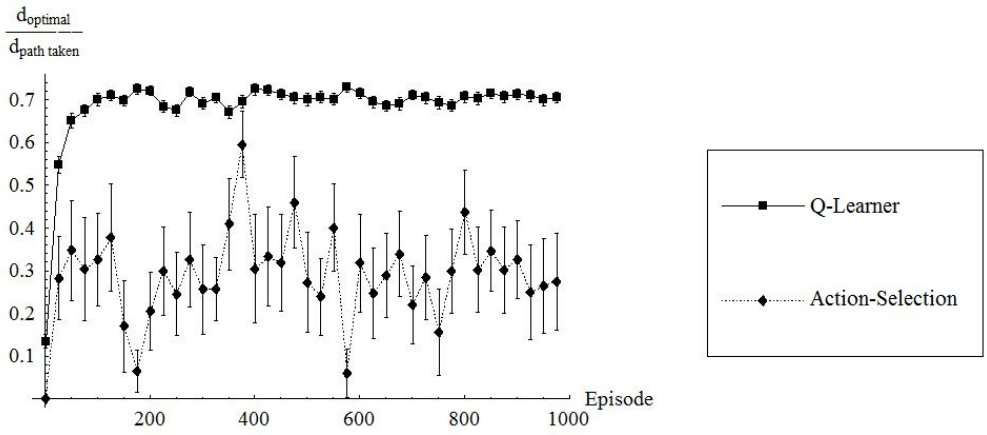

Figure 7: Performance of Q-Learning and Action-Selection in Gridworld

2.4 Discussion

The Q-learning algorithm significantly outperformed the simple action-selection policy (Figure 7). The Q-Learner learned its converged policy after about 200 episodes, while the action-selection policy did not lead to any learning. The reason for this is by implementing a stateless action selection policy in the gridworld, the problem became non-Markov. With no knowledge of what state it is in, the agent can't learn that an action which is good in one state isn't necessarily desirable in another state. In contrast, the gridworld problem is

Markov when a Q-learner is implemented. Thus, the Q-learner is able to learn a policy which allows it to find the goal state much faster than the action selection policy. Ultimately, in the grid world, the stateless action selection policy is equivalent to a random search policy, while the Q-learning algorithm allows for learning to take place.

3 Conclusion

The main implication of the data acquired during this analysis is that the action-selection policy and learning algorithm used in a domain must be chosen such that the Markov property is valid. Although an algorithm may work in one domain, there is no guarantee that it will work in another domain without modification. The action-selection policies performed well while solving the n-armed bandit, but performed very poorly in the gridworld problem. It would be expected, however, that the Q-learner which performed well in the gridworld would also perform well in the n-armed bandit problem, which can be thought of as the gridworld problem with only one state and 5 actions. As long as the Markov property is satisfied by an algorithm, it should be capable of learning.