Devin Koepl
November 5, 2010
HW 3
ME 537

1 N-Arm Bandit Problem

The N-arm bandit problem extends the simple one-arm bandit casino game, in which a player pulls the arm and receives some payout based on an unknown random distribution. However the N-arm bandit gives a player a choice of arms to pull, each arm samples payouts from its own unknown random distribution. By playing the N-arm bandit a player can begin to guess at arm's distributions and favor those arms with higher expected payouts.

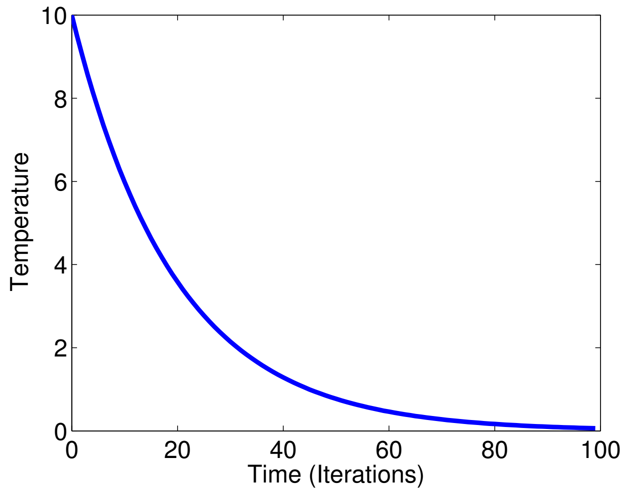
We test our simple action value reinforcement learner on a five-arm bandit in simulation with normal payout distributions on each of its arm. The mean and variance of each of our five-arm bandit's arms is as follows:

Arm	A1	A2	A3	A4	A5
Mean	1	1	2	2	0
Variance	5	1	1	1	10

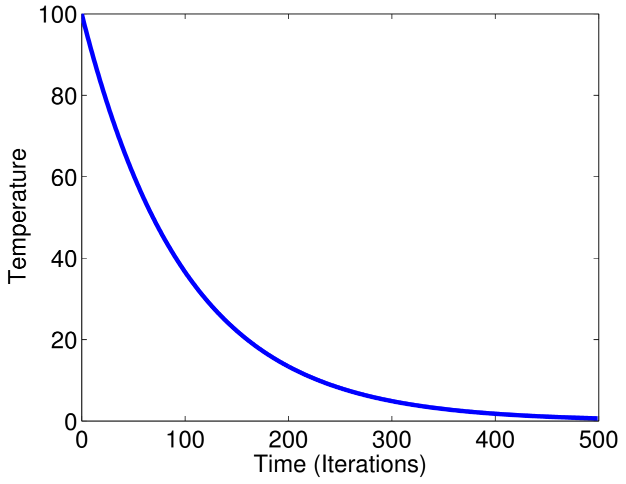
We initialize our learner's expected values for the five actions at the beginning of each trial by sampling a random variable with a gaussian distribution, a mean of 1, and a variance of 0.1. Our learner updates its expected values, Q_k, for the five available actions by playing the N-arm bandit and updating its expected values according to a fixed learning rate of $\alpha = 0.1$ and the payout received for an action, r, as follows:

$$Q_k = Q_k + \alpha\,(r - Q_k)$$

We expect a good reinforcement learner solution to first play the five arms at random, but favor arms three and four after repeated iterations, since these arms have the highest expected payouts.



(a) The temperature curve used for the N-arm bandit problem.



(b) The temperature curve used for the gridworld problem.

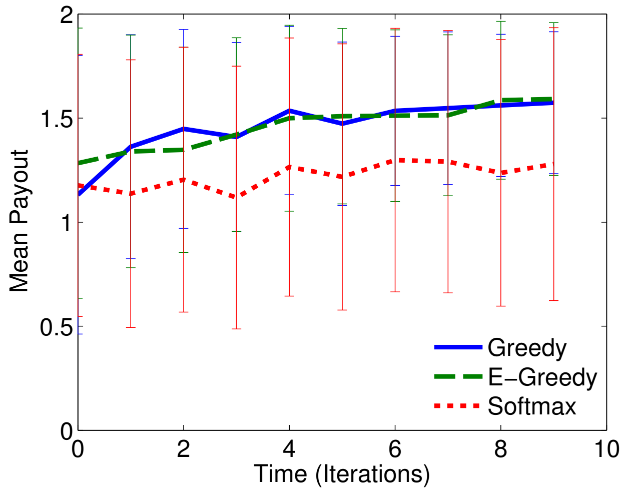
Figure 1: The temperature curves used for softmax action selection. The temperature decayed over time, making non-greedy actions less likely later in a trial.

1.1 Action Selection Policy

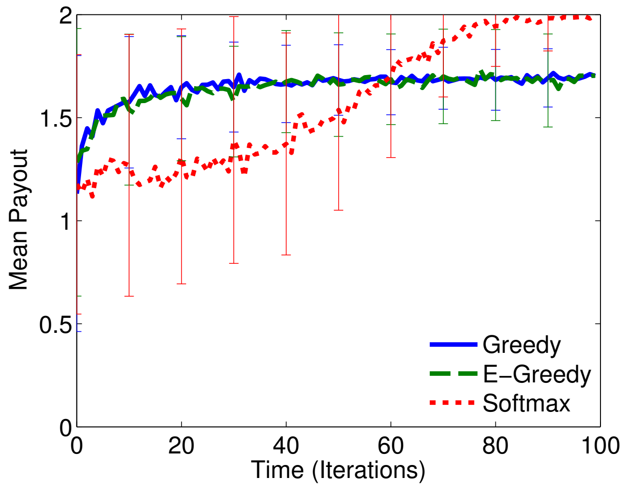
We compared greedy, ϵ-greedy, and softmax action selection policies for our simple action value reinforcement learner.

Greedy Action Selection The first action selection policy we considered was to always choose the arm with the highest expected payout. This action selection policy converged in only about ten iterations, but did not always learn to pull the arms with the highest expected payouts, as shown in Figure 2(b), since the mean payout converged to a value less than the maximum expected payout for the N-arm bandit considered. Greedy action selection does not ever consider actions with lower expected payout, this means that it is unable to push through local maxima to higher rewards.

ϵ-Greedy Action Selection The ϵ-greedy action selection policy closely followed the convergence of the greedy action selection policy, as shown in Figure 2(b). In general, we would expect ϵ-greedy action selection to converge slower than greedy action selection, since some iterations would be spent testing actions with lower expected payouts, but be more capable of overcoming local maxima, and converge to a higher mean payout after sufficient iterations. We chose an ϵ value of just 0.05, which does not seem to be enough for our ϵ-greedy action selection policy to outperform our greedy action selection policy on the N-arm bandit. When ϵ is too low, ϵ-greedy action selection becomes almost indistinguishable from greedy action selection, since both policies wind up choosing the action with the highest expected payout.



(a) Performance of the action value learner on the N-arm bandit problem over ten iterations.



(b) Performance of the action value learner on the N-arm bandit problem over one hundred iterations.

Figure 2: Performance on the N-arm bandit problem was measured by taking the mean payout for an iteration across $10,000$ trials. Errorbars represent one quarter standard deviation from top to bottom in these figures.

Softmax Action Selection The third action selection policy we tested on the N-arm bandit problem was softmax action selection. This action selection policy used a decaying temperature curve, shown in Figure 1(a), and generated by computing the temperature at timestep T_N as:

$$T_N = T_{N-1} \cdot T_{decay} \tag{1}$$

were $T_0 = 10$ and $T_{decay} = 0.95$. The resulting temperature curve and the expected rewards of the five available actions were used to choose actions with higher expected rewards with increasing probability as the number of iterations increased. The temperature curve we chose resulted in a lot of random action selection during early iterations, and lower mean payouts, as shown in Figure 2(a), but a very high probability of identifying the actions with the highest expected payouts in later iterations, as shown in Figure 2(b). The softmax action selection policy approaches two, the highest expected payout of any of the available actions after about eighty iterations. The convergence of the softmax action selection action value learner could probably be expediated by increasing the decay of the temperature curve, but the curve chosen was sufficient to illustrate the power of the softmax action selection policy over greedy and ϵ-greedy action selection policies.

2 Gridworld Problem

The second problem domain we consider is a gridworld with an exit, as shown in Figure 3, the goal in this problem domain is to reach the exit in as few timesteps as possible, starting from a random location within the 5×10 gridworld. At each timestep our agent is allowed to move one square up, down, right, or left or remain at its current position. This conviently provides our agent with five available actions, so the code that we used for the N-arm bandit problem can be used without modification.

For this problem domain, rewards were assessed at each timestep, such that rewards would backpropogate for our Q-learner. If rewards were only assessed when the exit was found, the expected values of the state action pairs leading to the exit's discovery would never be updated. A reward of one hundred was assessed for locating the exit at a timestep, or a reward of zero otherwise. We restricted each iteration to 250 timesteps, or the number of timesteps that our agent required to locate the exit, whichever came first. Each trial consisted of five-hundred iterations, and final results, shown in Figures 4(a) and 4(b), were found by taking the mean number of timesteps required to find the exit during an iteration across one thousand trials.

2.1 Action Value Learning

We first tested the simple action value reinforcement learner used in the N-arm bandit problem domain in the gridworld domain. This simple action value learner was able to identify actions with higher expected payouts for the N-arm bandit problem, were the optimal actions never changed, but did not extend well using one of the greedy action selection policies to our gridworld domain were the optimal actions changed depending upon the agents location in the gridworld. The solutions learned by our simple action value learner consisted of a position independent policy, such as the one shown in Figure 5(a).

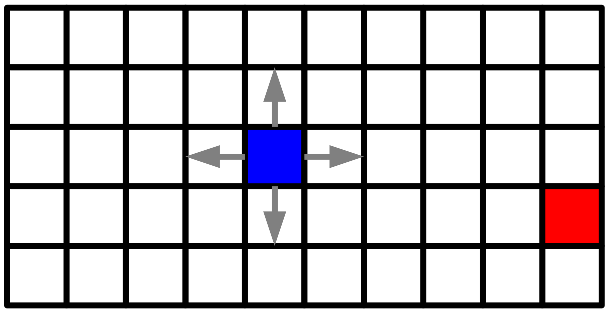
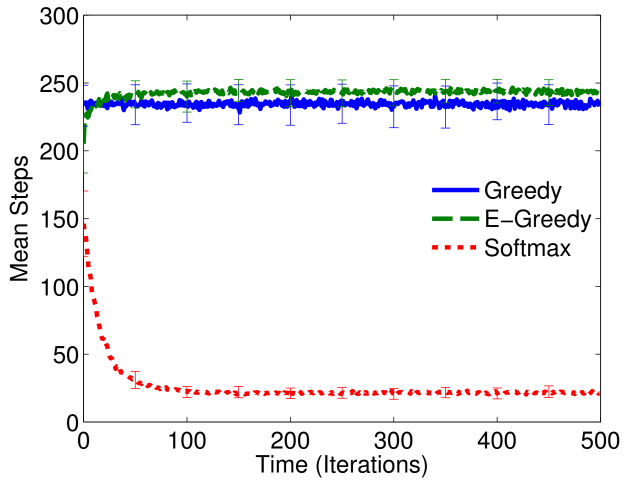


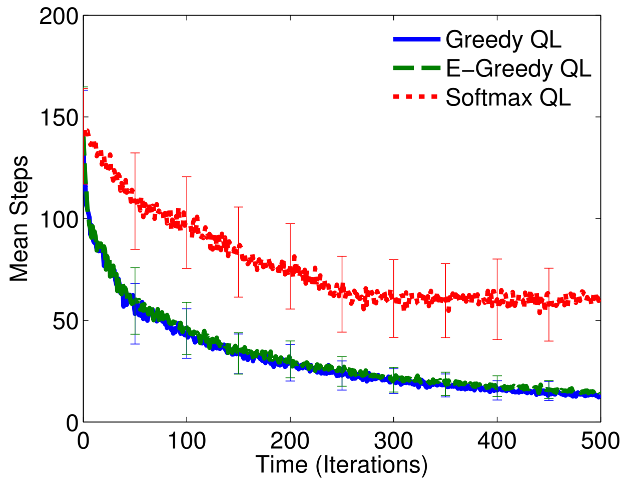
Figure 3: The gridworld used in these experiments had a goal, represented by the red square, and an agent represented by the blue square. The goal's location remained constant across iterations and trials, but at the start of each iteration agent's location was set to a square inside the gridworld.

Greedy Action Selection Greedy action selection aplied to our simple action value learner was unable to learn a policy that reliably led to locating the exit, as shown in Figure 4(a). Action value learning is unable to correlate states with actions, since it has no notion of state. Our simple action value learner was only able to learn policies such as move right, as shown in Figure 5(a), and therefore only found the exit when it started in the same row or column as the exit, and had the appropriate policy to lead it in the right direction.

ϵ-Greedy Action Selection ϵ-greedy action selection encountered many of the same problems as greedy action selection with our simple action value learner. We would expect ϵ-greedy action selection to perform slightly better than greedy action selection, but this was not the case. Our intuition suggests that ϵ-greedy action selection should for the most part choose the greedy action, such as move right, eventually pushing against the right wall, occasionally making random moves up, down, and to the left until finding the exit. This did not seem to occur though as ϵ-greedy action selection performed slightly worse than greedy action selection. We used an ϵ constant of 0.1, which may have been too low for the problem. If we had chosen a higher ϵ value, our learner using ϵ-greedy action selection would have made more random moves, possibly increasing its likelihood of locating the exit.



(a) Performance of the simple action value learner inside the gridworld with different action selection policies.



(b) Performance of the Q-learner inside the gridworld with diffent action selection policies.

Figure 4: Performance of the simple action value reinforcement learner and Q-learner were measured as the number of timesteps an agent took to reach the goal square. Results were found by averaging the performance over one thousand trials, and the errorbars represent one half a standard deviation of error from top to bottom.

Softmax Action Selection Softmax action selection was the only action selection policy for the simple action value learner that was able to find the exit with any sort of consistency, as shown in Figure 4(a). The temperature curve we chose, shown in Figure 1(b), and generated using (1) with $T_0 = 100$ and $T_{decay} = 0.98$, kept the probability of making moves with a lower expected payout sufficiently high to allow our agent to blindly stumble upon the exit during most iterations with only a simple policy such as move right, but occasionly move down, and with decreasing probability, move up, left, or stay in place. However, our temperature curve also decayed quickly enough to give our softmax action selection policy a very fast rate of convergence by discouraging moves such as stay in place or move left early on that never result in finding the exit. The zero reward for moves that do not directly lead to the exit resulted in the expected values of moving left and staying in place to decay rapidly during early iterations.

2.2 Q-Learning

We implemented a Q-learner with its state as our agent's position in the gridworld. Our Q-learner was able to learn state specific actions, making solutions with different actions depending on our agent's position possible, as shown in Figure 5(b). We arbitrarily chose to use a learning rate of $\alpha = 0.1$ and $\gamma = 0.1$ for our

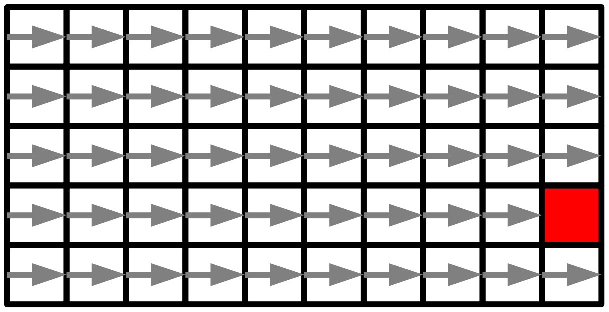
Q-learning algorithm which updated the expected values of state-action pairs, $Q(s_t, a_t)$ as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left(r(s_t, a_t) + \gamma \cdot maxQ(s_{t+1}, a) - Q(s_t, a_t) \right)$$

At the beginning of each trial we initialized our Q-learners expected values by sampling a gaussian distribution with a mean of 1 and a variance of 0.1.

Greedy Action Selection The first action selection policy we tested with our Q-learning algorithm was greedy, which resulted in the best convergence of any of the action selection policies tested on our Q-learner, as shown in Figure 5(b).

ϵ-Greedy Action Selection ϵ-greedy action selection gave a performance curve comparable to greedy action selection, as shown in Figure 5(b). Greedy action selection seems to perform slightly better than ϵ-greedy action selection on this problem because ϵ-greedy action selection made random, not necessarily good moves, with probability $\epsilon = 0.1$.


(a) Example solution generated by the simple action value learner.

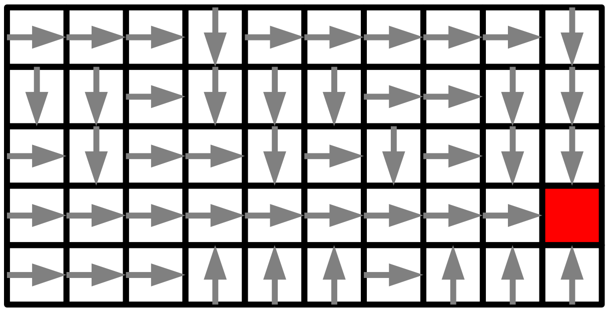

(b) Example solution generated by the Q-learner.

Figure 5: These example solutions were generated by running our action value and Q-learning algorithms for 500 iterations using ϵ-greedy action selection. After 500 iterations the resulting Q values (predicted values) were fed into a greedy action selection routine, the resulting policies are shown in the figures above.

Softmax Action Selection Our softmax action selection policy did not perform as well as our greedy or ϵ-greedy action selection policies. We attribute the poor performance of this selection policy to the temperature curve used, shown in Figure 1(b), and generated using (1) with $T_0 = 100$ and $T_{decay} = 0.98$. While this temperature curve resulted in good convergence for the simple action value learner using softmax action selection, it seems to decay too quickly for our softmax algorithm to explore the available state-action pairs, or form good approximations for their values. In general, with a carefully tuned temperature curve, we would expect the performance of softmax action selection to eventually overtake greedy or ϵ-greedy action selection policies, since greedy action selection is subject to local minima, and ϵ-greedy action selection is prone to making random moves even after the learner has converged to an optimal action policy.