CS 294-40, Fall 2008
Problem Set #1: Value iteration, contractions

Due before 11am on Tuesday, November 18.

NOTE: Please refer to the class webpage (http://inst.eecs.berkeley.edu/~cs294-40/fa08/) for the homework policy.

1 Value iteration, Policy iteration, Q learning, Reward shaping

The code referred to below is available from the class website. For all questions, hand in the code you wrote as well as the plots asked for and your answers to the questions.

In this problem you will implement policy iteration and Q-learning for an agent living in a 2D gridworld. The code available from the class website already contains an implementation of value iteration for documentation/illustration purposes.

1. [**4 points**] **Policy Evaluation**

   Implement policy evaluation by filling in the file policy_evaluation.m. Compute the policy evaluation by repeatedly performing Bellman back-ups for the fixed policy.

   You can now run the file ps3_vi.m. Hand in the learning curve (the utility of the attained policy versus the number of value iterations).

2. [**4 points**] **Gauss-Seidel back-ups**

   Implement Gauss-Seidel value iteration into value_iteration_GS.m. Run ps3_vi_GS.m and hand in the learning curve.

   Recall the pseudo-code for Gauss-Seidel value iteration:

   ```
   Iterate:
   for $s = 1, 2, \ldots, |S| : V(s) = \max_a (R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s'))$
   ```

3. [**4 points**] **Policy iteration**

   Implement policy iteration into ps3_pi.m. Recall, policy iteration alternates between a policy evaluation step, and a policy improvement step (picking the greedy policy w.r.t. the current value function; you still have to implement this functionality in greedy_policy.m). In fact, you have already implemented the policy evaluation step for question 1. In the policy evaluation step, one can use an approximate evaluation by running the evaluation for a "small" number of iterations.

   Plot the learning curves for 1, 5, and 10 iterations in the policy evaluation. Plot the performance of the attained policies as a function of the total number of Bellman iterations.

4. **[4 points] Q-learning**

   Implement Q-learning into ps3_q.m

   Plot the learning curve.

5. **[8 points] Reward shaping**

   Implement Q-learning with reward shaping into ps3_q_w_shaping.m. Using a shaping potential ϕ, i.e., the shaped reward for a transition (s, a, s') becomes $R(s, a) + \gamma\phi(s') - \phi(s)$.

   (a) Use the value function obtained from value iteration as the shaping potential ϕ.

   (b) Use the value function offset by the constants +10, and -10 respectively.

   (c) Use the value function scaled by a factor .01 and 100 respectively.

   Plot the learning curves. Comment on the results.

2 Kalman filtering, smoothing, EM

A skeleton for the code below is provided from the class website.

6. **[4 points] Kalman filtering**

   Implement a Kalman filter into ps3_kf.m.

   Plot the true state and the state estimate of the KF over time.

7. **[4 points] Kalman smoothing**

   Implement a Kalman smoother into ps3_kf.m

   Plot the true state, the filtered state estimates, and the smoothed state estimates.

8. **[4 points] EM**

   Implement the EM algorithm to learn the process and observation covariance matrices. Plot the log-likelihood of the data (vertical axis) versus number of EM iterations (horizontal axis).

   Once the EM algorithm has converged. Run the filter and the smoother over the data and plot the true state and the according state estimates. Also report the learned covariance matrices.