COMP-652 Assignment 6 Solutions

Jordan Frank

01/12/08

1 Reinforcement Learning.

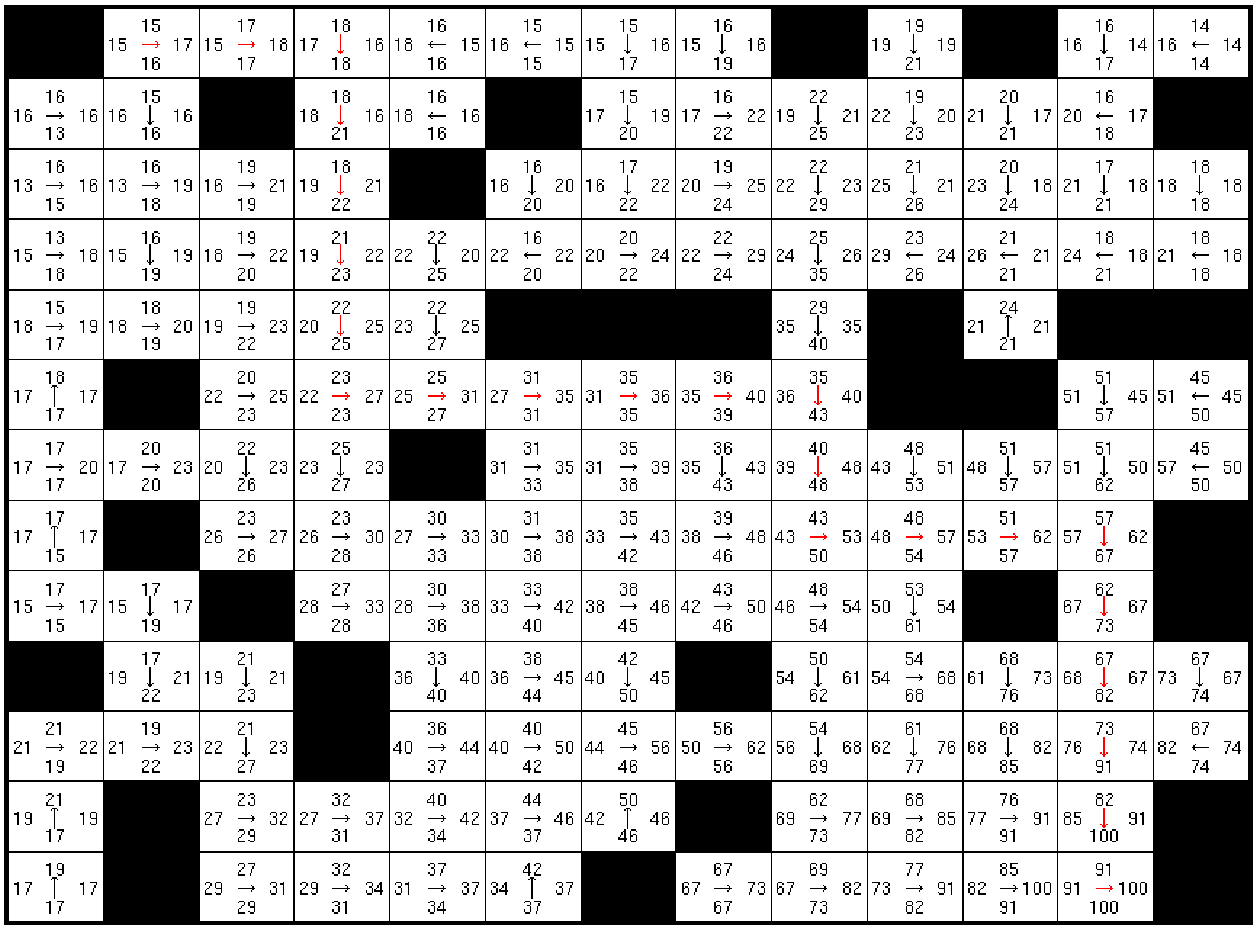

Figure 1: Q-values: Action values shown in positions corresponding to actions. Arrows denote optimal actions, with red arrows denoting the optimal path.

Matlab code for batch Q-learning:

```
sCoords = load('HW6_Q1_StateCoords.txt');
traj = load('HW6_Q1_Traj.txt');
alpha = 1.0;
gamma = 0.9;
S = 137;
A = 4;

dQ = ones(S,A);
Q  = zeros(S,A);
while(sum(dQ(:)) > 0)
   Qp = Q;
   for i=1:3:length(traj)-1
       s = traj(i); a = traj(i+1); r = traj(i+2); sp = traj(i+3);
       Q(s,a) = (1-alpha)*Q(s,a) + alpha*(r + gamma*max(Q(sp,:)));
   end
   dQ = abs(Q-Qp);
end
[tmp,pi_opt] = max(Q,[],2);

% Plotting code left out, email me if you want a copy.
```

2 Hidden Markov models.

A) Empirical distributions:

```
data = load('HW6_Q2_Data.txt');
pA = zeros(1,5);
pB = zeros(1,5);
for i=1:5
    pA(i) = sum(data(1,:)==i);
    pB(i) = sum(data(2,:)==i);
end
pA = pA / size(data,2);
pB = pB / size(data,2);

Distribution for source A
1: 0.080 ********
2: 0.085 ********
3: 0.185 ******************
4: 0.300 ******************************
5: 0.350 ***********************************

Distribution for source B
1: 0.380 **************************************
2: 0.255 *************************
3: 0.185 ******************
4: 0.090 *********
5: 0.090 *********
```

B) See Figure 2 for plot of conditional state probabilities.

```
function alphas = fwd(obs, p_trans, p_obs, p_init)
S = size(p_trans,1);
N = length(obs);
alphas = zeros(S,N);
alphas(:,1) = p_obs(:,obs(1)).*p_init;
for i=2:N
    alphas(:,i) = p_obs(:,obs(i)).*(p_trans*alphas(:,i-1));
end
end

function betas = bkwd(obs,p_trans,p_obs)
S = size(p_trans,1);
N = length(obs);
betas = zeros(S,N);
betas(:,N) = 1;
for i=N-1:-1:1
    betas(:,i) = p_trans' * (p_obs(:,obs(i+1)).*betas(:,i+1));
end
end

function [p_s,p_ssp] = viterbi(obs,p_trans,p_obs,p_init)
% based on code submitted by Anqi Xu
alphas = fwd(obs,p_trans,p_obs,p_init);
betas = bkwd(obs,p_trans,p_obs);
p_s = alphas .* betas ./ sum(alphas(:,end));

S = size(p_trans,2);
N = length(obs);
p_ssp = zeros(S,L-1,S);
for i = 1:S
    p_ssp(:,:,i) = ...
        repmat(alphas(i,1:N-1),S,1) .* ...
        repmat(p_trans(:,i),1,N-1) .* ...
        p_obs(:, obs(2:N)) .* ...
        betas(:, 2:N) ./ sum(alphas(:,end));
end
end

data = load('HW6_Q2_Data.txt');
obs = data(3,:);
p_trans = [0.9,0.1;0.1,0.9];
p_init = [0.5;0.5];
% make sure you run part A first
p_obs = [pA;pB];
p_s = viterbi(obs,p_trans,p_obs,p_init);

% p_s contains the conditional state occupancy probabilities for each time step
% Plotting code omitted, email me if you'd like a copy.
```

C) See Figure 2 for plot of conditional state probabilities. This code uses functions from Part B.

```
function [p_s, p_trans] = BW(obs, p_trans, p_obs, p_s)
```

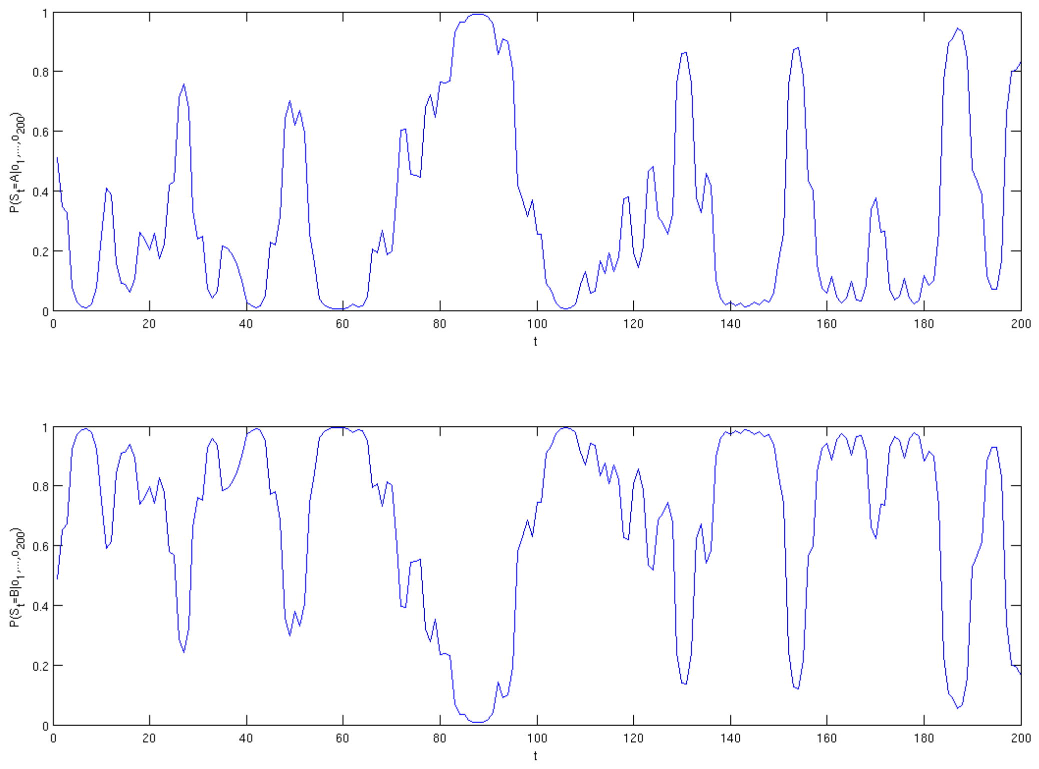

Figure 2: Probabilities of states A and B given observation sequences, computed by forward-backward algorithm.

```
N = length(obs);
S = size(p_trans,1);
step = 1;
while 1
    % E-step
    [p_s_temp, p_trans_temp] = viterbi(obs, p_trans, p_obs, p_s);

    % M-step
    p_s_next = p_s_temp(:,1);
    p_s_total = sum(p_s_temp(:,1:N-1),2);
    p_trans_total = reshape(sum(p_trans_temp(:,:,:),2),S,S);
    p_trans_next = repmat(1./p_s_total',S,1).*p_trans_total;

    % check to see if we are done
    d_p_s = max(abs(p_s_next - p_s));
    tmp = abs(p_trans_next - p_trans);
    d_p_trans = max(tmp(:));
    p_s = p_s_next;
    p_trans = p_trans_next;
    if d_p_s < 1E-12 && d_p_trans < 1E-12
        break;
    end
    step = step + 1;
end
end

data = load('HW6_Q2_Data.txt');
obs = data(3,:);
p_trans = [0.5,0.5;0.5,0.5];
p_init = [0.5;0.5];
% make sure you run part A first
p_obs = [pA;pB];

[p_s, p_trans] = BW(obs, p_trans, p_obs, p_init);
% now recompute state probs using values from Baum-Welch
p_s = viterbi(obs, p_trans, p_obs, p_s);
% p_s contains the conditional state occupancy probabilities for each time step
% Plotting code omitted, email me if you'd like a copy.
```

The initial state occupancy probabilities are $p(S_0 = a) = 1.0$ and $p(S_0 = b) = 0.0$. The transition probability matrix is

$$\begin{array}{c|cc} & a & b \\ a & 0.5194 & 0.4806 \\ b & 0.2179 & 0.7821 \end{array}$$

Given that the first observation is 4, it is unsurprising that the initial state is estimated to be state a, since we are more likely to see a 4 when in state a than in state b. The transition probabilities are estimated fairly poorly compared to the known transition probabilities. However, the estimated model fits the data better, which we see if we calculate the likelihood of the data given the parameters $\theta = (p_a, p_b, p_{aa}, p_{ab}, p_{ba}, p_{bb})$, $p_o = p(o_1, \ldots, o_{200}|\theta)$. Given the model parameters in B, $p_o \approx 5.0^{-139}$, while the parameters in C give us $p_o \approx 5.0^{-137}$.

One thing we notice between the two plots in B and C is that although the plot in C varies far more than in B, it varies in a similar manner and C can be seen as an exaggerated version of B. In other words, the most pronounced extrema occur at the same times. It is obvious why plot C would oscillate more frequently, given that it is trying to fit the transition model to the data. For instance, if we are in state b and we see a 1, then given the observation probabilities it is reasonable to expect that we have transitioned to state a rather than stayed in b and emitted such an unlikely observation. Since we have no prior on the transition model, the algorithm is more likely to account for observations with higher probabilities of being emitted from specific states with transitions to those states. This is exactly what we see, a more uniform transition distribution.

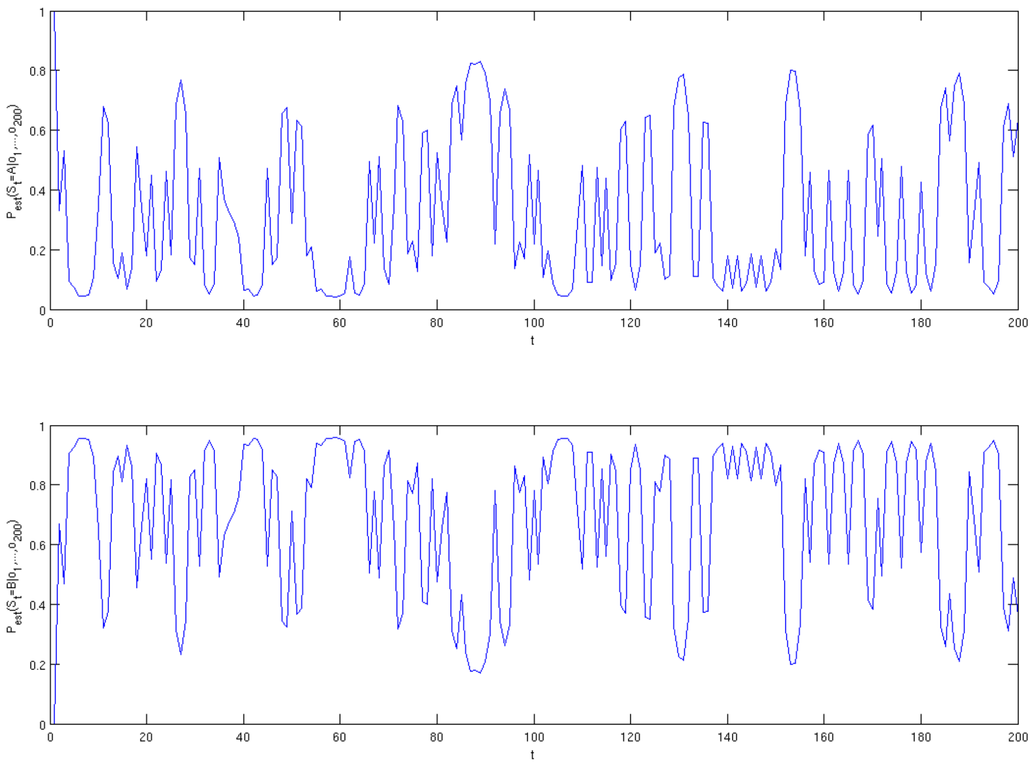

Figure 3: Probabilities of states A and B given observation sequences, computed by Baum-Welch.