AI2 - Assignment 2

Arve Nygård

10.02.2014

Part A

Unobserved variable X_t: Weather at time `t`. `Rain` or `Sun`. Observable variable E_t: Did the guy bring an umbrella or not, at time `t`.

Dynamic model

	R_t	S_t
R_{t-1}	.7	.3
S_{t-1}	.3	.7

R_t = P(rain = true), S_t = P(rain = false)

Observation model

	U_t	N_t
R_{t-1}	.9	.1
S_{t-1}	.2	.8

U_t = P(Umbrella = true), N_t = P(Umbrella = false)

Assumptions

1. Markov assumption - The current state depends only on a fiexd number of earlier states.
2. Stationary process - The transition model is fixed. Depending on location, this might not be reasonable. Weather patterns might change based on for example season.
3. Sensor Markov assumption - Sensor readings are only dependant on the state of the world (i.e. the unobservable variables), not earlier sensor values. This is pretty reasonable in this case.

Part B

```
function [ result ] = FORWARD( initialState,  transitionModel, observationModel, observations )
% FORWARD Hidden Markov-model FORWARD algorithm.
%   initialState is a row vector containing the probability distribution
%   we have assumed for the inital X_0 state
%
%   transitionModel is a n*n matrix encoding the transition model, where
%   "from" state is chosen as row number, and "to" state is chosen as
%   column number.
%
%   observationModel is a n*n matrix encoding the observation model, where
%   row number represents the causing state of the world, and
%   column number represents the resulting observable value.
%
%
    result = initialState;
    for i=1:size(observations,2)
        % prediction step
        result = result * transitionModel;

        % update observed values
        observation = observations(i);
        observationProbabilities = observationModel(1:end, observation).';
        % slice observationModel to get probability distr. matching the observations%
        updatedState = result .* observationProbabilities;

        %normalize
        result = updatedState / sum(updatedState)
    end
end
```

Running the program, gives the expected output for `{Umbrella, Umbrella}`: "'matlab initial = [.5 .5] obs = [1 1] % [Umbrella Umbrella] observationModel = [.9 .1; .2 .8] transitionModel = [.7 .3; .3 .7]

FORWARD(initial, transitionModel, observationModel, obs); ans = [0.8834 0.1166]

"'

Running the program again, with `obs = [1 1 2 1 1]` gives:

```
result = [0.8182    0.1818]
result = [0.8834    0.1166]
result = [0.1907    0.8093]
result = [0.7308    0.2692]
result = [0.8673    0.1327]

ans = [0.8673    0.1327]
```

Part C

```
function [ sv ] = ForwardBackward(initialState, transitionModel, observationModel, observations)
    t = size(observations, 2);

    % n is number of states in prob. distribution
    % initialize backward to 1's
    n = size(initialState, 2);
    b= ones(1, n);

    % column vector
    fv = zeros(t+1, n);
    sv = zeros(t, n);

    fv(1, 1:end) = initialState;

    % store forward messages in fv
    for i=1:t,
        observation = observations(i);
        fv(i+1, 1:end) = FORWARD(fv(i, 1:end), transitionModel, observationModel, observation);
    end

    % backward
    for i=t:-1:1,
        tmp = fv(i+1, 1:end) .* b;
        normalized = tmp / sum(tmp);
        sv(i, 1:end) = normalized;

        % update backwards message
        observation = observations(i);
        b = observationModel(1:end, observation).' .* b * transitionModel;

        % VERY unsure about this normalization. Shuold it be here?
        b = b / sum(b);

        %print backward message to console
        b
    end
end
```

Running the program, gives the expected output for `{Umbrella, Umbrella}`:

```
initial = [.5 .5]
obs = [1 1] % [Umbrella Umbrella]
observationModel = [.9 .1; .2 .8]
transitionModel = [.7 .3; .3 .7]

ForwardBackward( initial,  transitionModel, observationModel, obs)
% backwards messages
b = 0.6273    0.3727
b = 0.6533    0.3467

% smoothed values
ans=0.8834    0.1166
    0.8834    0.1166
```

I find it weird that both smoothed values should be the same here, and I expect there is some error in my program.

Running the program again, with `obs = [1 1 2 1 1]` gives:

```
% backwards messages
b = 0.6273    0.3727
b = 0.6533    0.3467
b = 0.3763    0.6237
b = 0.5923    0.4077
b = 0.6469    0.3531

% smoothed values
ans =
    0.8673    0.1327
    0.8204    0.1796
    0.3075    0.6925
    0.8204    0.1796
    0.8673    0.1327
```