Assignment 3

Yann Debain
debain@kth.se

Harsha Holenarasipura Narasanna
harshahn@kth.se

May 6, 2019

1 Introduction

The goal of this assignment is to implement and verify the forward algorithm to train the HMM in Matlab. In addition to the forward algorithm, we will code and verify a method to calculate the probability of a feature Sequence.

The code is enclosed into the package **PattRecClasses-2**, you will find in this report the different tests implemented to verify that the code is correct (the forward and logprob algorithm are in the Annexes section).

2 Verify the Forward algorithm

2.1 Finite HMM

For the given test example, the parameters of the HMM source are :

$$q = \begin{pmatrix}1\\0\end{pmatrix}; \quad A = \begin{pmatrix}0.9 & 0.1 & 0\\0 & 0.9 & 0.1\end{pmatrix}; \quad B = \begin{pmatrix}b_1(x) = \mathcal{N}(0,1)\\b_2(x) = \mathcal{N}(3,2)\end{pmatrix}$$

We assume that the observed feature sequence is $x = [-0.2, 2.6, 1.3]$.

The results of the forward algorithm for this feature sequence is given in the paper. We have implemented the forward algorithm by following the different steps in the course and obtained the following outputs :

$$\hat{\alpha} = \begin{pmatrix}1 & 0.3847 & 0.4189\\0 & 0.6153 & 0.5811\end{pmatrix}; \quad c = \begin{pmatrix}1 & 0.1625 & 0.8266 & 0.0581\end{pmatrix}$$

The outputs are consistent with the results in the paper, thus we can assume the forward algorithm for a finite HMM is well implemented.

2.2 Infinite HMM

For the infinite-duration HMM, we remove the end state in the transition matrix and we will consider the model from the exercise 5.1. The parameters of the HMM source become :

$$q = \begin{pmatrix}1\\0\\0\end{pmatrix}; \quad A = \begin{pmatrix}0.3 & 0.7 & 0\\0 & 0.5 & 0.5\\0 & 0 & 1\end{pmatrix}; \quad B = \begin{pmatrix}1 & 0 & 0 & 0\\0 & 0.5 & 0.4 & 0.1\\0.1 & 0.1 & 0.2 & 0.6\end{pmatrix}$$

We assume that the observed feature sequence is $x = [1, 2, 4, 4, 1]$.

To verify the algorithm is well implemented we will verify that the outputs of the algorithm give the same results of the parameters computed by hand.

Results computed by hand

$$\hat{\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{7} & \frac{1}{79} & 0 \\ 0 & 0 & \frac{6}{7} & \frac{78}{79} & 1 \end{pmatrix}; \quad c = \begin{pmatrix} 1 & 0.35 & 0.35 & \frac{79}{140} & \frac{157}{1580} \end{pmatrix}$$

Results with Matlab

$$\hat{\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.1429 & 0.0127 & 0 \\ 0 & 0 & 0.8571 & 0.9873 & 1 \end{pmatrix}; \quad c = \begin{pmatrix} 1 & 0.35 & 0.35 & 0.5643 & 0.0994 \end{pmatrix}$$

We can verify :

— $\frac{1}{7} \simeq 0.1429$
— $\frac{6}{7} \simeq 0.8571$
— $\frac{1}{79} \simeq 0.0127$
— $\frac{78}{79} \simeq 0.9873$
— $\frac{79}{140} \simeq 0.5643$
— $\frac{157}{1580} \simeq 0.0994$

The outputs are consistent with the results computed by hand, thus we can assume the forward algorithm for an infinite HMM is well implemented too.

3 Verify the probability of a feature sequence

To verify the probability of a feature sequence we will follow the same procedure as the previous section.

3.1 Finite HMM

Using the parameters of the finite HMM, we should have $ln[P(X = x|q, A, B)] \simeq -9.1877$ (results in the paper).

By running the logprob function we obtain $ln[P(X = x|q, A, B)] \simeq -9.1877$ so we can assume that the function is well implemented for finite HMM.

3.2 Infinite HMM

For the infinite HMM, we will use the result from the exercise 5.1 :
$P(X = x|q, A, B) \simeq 6.86875e^{-3} \Rightarrow ln[P(X = x|q, A, B)] \simeq -4.9808$.

By running the logprob function we obtain $ln[P(X = x|q, A, B)] \simeq -4.9808$ so we can assume that the function is well implemented for infinite HMM.

4 Conclusion

As a conclusion, we have seen how to implement the forward algorithm to train the HMM. In addition, we have implemented a method to compute the probability of a feature sequence.

For a given set of parameters, we have verified if the codes have been well implemented for the different case of HMM (finite/infinite duration).

5 Annexes - Code

5.1 Forward algorithm

```
function [alfaHat, c]=forward(mc,pX)
%[alfaHat, c]=forward(mc,pX)
%calculates state and observation probabilities for one single data sequence,
%using the forward algorithm, for a given single MarkovChain object,
%to be used when the MarkovChain is included in a HMM object.
%
%Input:
%mc= single MarkovChain object
%pX= matrix with state-conditional likelihood values,
%   without considering the Markov depencence between sequence samples.
%       pX(j,t)= myScale(t)* P( X(t)= observed x(t) | S(t)= j ); j=1..N; t=1..
    T
%       (must be pre-calculated externally)
%NOTE: pX may be arbitrarily scaled, as defined externally,
%   i.e., pX may not be a properly normalized probability density or mass.
%
%NOTE: If the HMM has Finite Duration, it is assumed to have reached the end
%after the last data element in the given sequence, i.e. S(T+1)=END=N+1.
%
%Result:
%alfaHat=matrix with normalized state probabilities, given the observations:
%       alfaHat(j,t)=P[S(t)=j|x(1)...x(t), HMM]; t=1..T
%c=row vector with observation probabilities, given the HMM:
%       c(t)=P[x(t) | x(1)...x(t-1),HMM]; t=1..T
%       c(1)*c(2)*..c(t)=P[x(1)..x(t)| HMM]
%   If the HMM has Finite Duration, the last element includes
%   the probability that the HMM ended at exactly the given sequence length, i
    .e.
%   c(T+1)= P( S(T+1)=N+1| x(1)...x(T-1), x(T)   )
%Thus, for an infinite-duration HMM:
%   length(c)=T
%   prod(c)=P( x(1)..x(T) )
%and, for a finite-duration HMM:
%   length(c)=T+1
%   prod(c)= P( x(1)..x(T), S(T+1)=END )
%
%NOTE: IF pX was scaled externally, the values in c are
%   correspondingly scaled versions of the true probabilities.
%
%-------------------------------------------------------------------
%Code Authors:
%-------------------------------------------------------------------

T = size(pX,2); %Number of observations
N = nStates(mc); %Number of states

q = mc.InitialProb;
A = mc.TransitionProb;

```

```
c = zeros(1, T);
alfa = zeros(N, T);
alfaHat = zeros(N, T);

isFinite = (N ~= size(A, 2)); %Test finite case

% Initialization
alfa(:, 1) = q.*pX(:, 1);
c(1) = sum(alfa(:, 1));
alfaHat(:, 1) = alfa(:, 1) / c(1);

% Loop
for t = 2:T
    alfa(:, t) = A(:, 1:N)'*alfaHat(:, t-1).*pX(:, t);
    c(t) = sum(alfa(:, t));
    alfaHat(:, t) = alfa(:, t)/c(t);
end

% Termination
if isFinite
        c(1, end+1) = alfaHat(:, T)' * A(:, N+1);
end

end
```

5.2 HMM logprob

```
%logP=logprob(hmm,x) gives conditional log(probability densities)
%for an observed sequence of (possibly vector-valued) samples,
%for each HMM object in an array of HMM objects.
%This can be used to compare how well HMMs can explain data from an unknown
    source.
%
%Input:
%hmm=    array of HMM objects
%x=      matrix with a sequence of observed vectors, stored columnwise
%NOTE:  hmm DataSize must be same as observed vector length, i.e.
%       hmm(i).DataSize == size(x,1), for each hmm(i).
%       Otherwise, the probability is, of course, ZERO.
%
%Result:
%logP=  array with log probabilities of the complete observed sequence.
%logP(i)=   log P[x | hmm(i)]
%           size(logP)== size(hmm)
%
%The log representation is useful because the probability densities
%exp(logP) may be extremely small for random vectors with many elements
%
%Method: run the forward algorithm with each hmm on the data.
%
%Ref:    Arne Leijon (20xx): Pattern Recognition.
%
%-------------------------------------------------------------
%Code Authors:
```

```
%-----------------------------------------------------------------

function logP=logprob(hmm,x)
hmmSize=size(hmm);%size of hmm array
T=size(x,2);%number of vector samples in observed sequence
logP=zeros(hmmSize);%space for result
for i=1:numel(hmm)%for all HMM objects
    %Note: array elements can always be accessed as hmm(i),
    %regardless of hmmSize, even with multi-dimensional array.
    %
    %logP(i)= result for hmm(i)

    mc = hmm(i).StateGen;
    B = hmm(i).OutputDistr;
    [pX, logS] = prob(B, x);

    [~, c] = forward(mc, pX);
    logP(i) = sum(log(c)) + sum(logS);

end
end
```