CO495 - Coursework 2

Mehdi Bahri

March 10, 2016

Contents

1 Description of the code

We implemented the following algorithms in their respective MATLAB files.

1.1 Generalities

The code is split into subfiles to promote re-usability. We implemented the code for the forward and the forward-backward algorithms separately. The function `normalize.m` is taken from Kevin Murphy's code for its book *Machine Learning: A probabilistic perspective*.

Run `Init.m` to add the test cases and the helper functions to the path.

1.2 Convergence criterion

The convergence criterion for both of the EM implementation is the following:

- Start with initial log likelihood $-\infty$
- At each iteration t, compute the log likelihood $L(t)$ averaged over all the data sequences
- If the log likelihood doesn't change by more than $\epsilon|L(t-1)|$ stop

Where $\epsilon \in \mathbb{R}_+^*$ is typically 10^{-4} or 10^{-5}.

1.3 EM for mixture of Multinoulli

The code for the EM algorithm for a mixture of discrete distributions (multinoulli) is in `EM_estimate_discrete.m`.
Input parameters:

- `Y`: Input data (N by T matrix)
- `Nhidden`: Number of hidden states
- `Niter`: Maximum number of iterations
- `epsilon`: Percentage of change for the convergence criterion

1.4 EM for mixture of Gaussians

The code for the EM algorithm for a mixture of normal distributions is in `EM_estimate_gaussian.m`.
Input parameters:

- `Y`: Input data (N by T matrix)
- `Nhidden`: Number of hidden states
- `Niter`: Maximum number of iterations
- `epsilon`: Percentage of change for the convergence criterion

We implemented farthest clustering[1] for initializing the means. The idea is as follows:

- Sample one data point from the observations and set is a the first mean μ_1 of the mixture model
- For every subsequent mean μ_k

  Compute the squared distances D_i of every data point to all the means $\mu_1, \ldots, \mu_{k-1}$

  Compute $D = \min D_1 \ldots D_{k-1}$ for every data point

  Sample μ_k from the data with probability proportional to D

This gives very good initial means and ensures the initial means "span" the data when the mixtures are sufficiently separated.

1.5 Viterbi decoding

The code for the Viterbi algorithm is in `ViterbiDecode.m`.
Input parameters:

- `Y`: Input data (N by T matrix)
- `Nhidden`: Number of hidden states
- `type`: Type of observations

Input a parameter `type` is mandatory and should be either `gauss` or `multinoulli`.

2 Results for EM

All the tests were performed on 100 sequences of 100 timesteps and averaged over 10 runs.

[1] K. Murphy page 357

2.1 Mixture of Multinoulli

2.1.1 Two hidden states

Parameters:

```
N  = 100;          % number of sequences
T  = 100;         % length of the sequence
pi = [.5; .5]; % inital probability pi_1 = 0.5 and pi_2 =0.5

%%two states hence A is a 2X2 matrix
A  = [0.4 0.6 ; 0.4 0.6 ];         %p(y_t|y_{t-1})

%%alphabet of 6 letters (e.g., a die with 6 sides) E(i,j) is the
E = [1/6 1/6 1/6 1/6 1/6 1/6;      %p(x_t|y_{t})
    1/10 1/10 1/10 1/10 1/10 1/2];

[ Y, S ] = HmmGenerateData(N, T, pi, A, E );
```

Mean results:

A	**B**	π
$\begin{pmatrix} 0.4733 & 0.5267 \\ 0.3156 & 0.6844 \end{pmatrix}$	$\begin{pmatrix} 0.1725 & 0.1740 & 0.1664 & 0.1920 & 0.0975 & 0.1977 \\ 0.1834 & 0.1587 & 0.1133 & 0.2773 & 0.0828 & 0.1844 \end{pmatrix}$	[0.5563, 0.4437]

Table 1: Results for two hidden states

2.1.2 Three hidden states

Parameters:

```
N  = 100;          % number of sequences
T  = 100;         % length of the sequence
pi = [1/3; 1/3; 1/3]; % inital probability pi_1 = 0.5 and pi_2 =0.5

A  = [0.1 0.3 0.6 ; 0.3 0.1 0.6 ; 0.5 0.2 0.3];

%%alphabet of 6 letters (e.g., a die with 6 sides) E(i,j) is the
E = [1/6 1/6 1/6 1/6 1/6 1/6;      %p(x_t|y_{t})
    1/10 1/10 1/10 1/10 1/10 1/2;
    1/3 1/3 1/12 1/12 1/12 1/12];

[ Y, S ] = HmmGenerateData(N, T, pi, A, E );
```

Mean results:

A	**B**	π
$\begin{pmatrix} 0.2946 & 0.2922 & 0.4132 \\ 0.4696 & 0.1993 & 0.3312 \\ 0.3649 & 0.3851 & 0.2500 \end{pmatrix}$	$\begin{pmatrix} 0.1051 & 0.1601 & 0.2609 & 0.1994 & 0.0898 & 0.1847 \\ 0.2101 & 0.2072 & 0.1437 & 0.0961 & 0.2386 & 0.1043 \\ 0.2877 & 0.1782 & 0.1386 & 0.0945 & 0.0809 & 0.2200 \end{pmatrix}$	[0.2712, 0.3468, 0.3821]

Table 2: Results for three hidden states

2.2 Mixture of Gaussians

2.2.1 Two hidden states

Parameters:

```
N  = 100;          % number of sequences
T  = 100;         % length of the sequence
pi = [0.5; 0.5]; % inital probability pi_1 = 0.5 and pi_2 =0.5

%%two states hence A is a 2X2 matrix
A  = [0.4 0.6 ; 0.4 0.6 ];         %p(y_t|y_{t-1})

%%one dimensional Gaussians

E.mu    =[ .1  5]; %%the means of each of the Gaussians
E.sigma2=[ .4 .8]; %%the variances
```

Mean results:

A	μ	σ^2	π
$\begin{pmatrix} 0.4022 & 0.5978 \\ 0.3911 & 0.6089 \end{pmatrix}$	[4.9934, 0.0886]	[0.8006, 0.3922]	[0.4800, 0.5200]

Table 3: Results for two hidden states

2.2.2 Three hidden states

```
N  = 100;          % number of sequences
T  = 100;         % length of the sequence
pi = [1/3; 1/3; 1/3]; % inital probability pi_1 = 0.5 and pi_2 =0.5

A  = [0.1 0.3 0.6 ; 0.3 0.1 0.6 ; 0.5 0.2 0.3];         %p(y_t|y_{t-1})

%%one dimensional Gaussians
E.mu    =[ .1 5 20]; %%the means of each of the Gaussians
E.sigma2=[ .4 .6 .8]; %%the variances

[ Y, S ] = HmmGenerateData(N, T, pi, A, E, 'normal');
```

Mean results:

A	μ	σ^2	π
$\begin{pmatrix} 0.2919 & 0.5137 & 0.1944 \\ 0.6117 & 0.1014 & 0.2869 \\ 0.6173 & 0.2930 & 0.0897 \end{pmatrix}$	[19.9929, 0.1167, 4.9936]	[0.7607, 0.3988, 0.6056]	[0.3400, 0.3400, 0.3200]

Table 4: Results for three hidden states

3 Results for Viterbi

3.1 Mixture of Multinoulli

3.1.1 Two hidden states

The typical precision of the most probable sequence of states is 48.5% with 1000 sequences of 100 timesteps.

3.1.2 Three hidden states

The typical precision of the most probable sequence of states is 32.3% with 1000 sequences of 100 timesteps.

3.2 Mixture of Gaussians

3.2.1 Two hidden states

The typical precision of the most probable sequence of states is 99.9% with 1000 sequences of 100 timesteps.

3.2.2 Three hidden states

The precision with 1000 sequences of 100 timesteps is unstable. In our experiments, we found values ranging from 56.98% to $5.0e-3$%. This is yet to be explained as the implementation seems to be working correctly for the other cases we tried.

4 Discussion

One important observation is that the different hidden states can be discovered in any order so any permutation of the transition matrix and of the other parameters is a valid estimation.

Our implementation computes maximum likelihood estimation without any kind of regularization.

The estimation of π is obviously very dependent on the number of sequences.

The performance of our implementation for discrete observations is quite poor. The algorithm is very sensitive to parameters initialization and converges most of the time to local extrema, as we can see with the observation matrix. Our attempt at avoiding these local extrema by adding a small random perturbation to $\mathbf{B}$ before re-normalizing has proved useful but doesn't solve the problem completely.

On the other hand, our implementation for Gaussian observations retrieves the parameters with very good precision for sufficiently separated Gaussians.

Viterbi decoding is very precise for the mixture of two Gaussians that we tried but has a surprisingly erratic behaviour for the case with three Gaussians that we tried. For discrete observations, Viterbi decoding has average performance.

The main obstacle with these algorithms is the initialization of the initial parameters. In particular, if we have implemented an efficient way of initializing the means, we didn't find efficient method of initializing the variance for Gaussian observations.

We actually tested one such method for Gaussian observations, which is based on the idea that the variance of a Gaussian distribution with known mean μ and unknown variance σ^2 follows an Inverse χ^2 distribution with $N-1$ degrees of freedom where N is the number of data points. The method we tested was as follows:

- Find the variance $\hat{\sigma}^2$ of the data points that are in an interval centred around the mean μ (the Gaussian distribution is symmetric), e.g: $[0.5\mu, 1.5\mu]$.
- Count the number N of such data points.
- Sample σ^2 from $(N-1)\hat{\sigma}^2/\chi^2_{N-1}$

This gave good results when the $\hat{\sigma}^2$ was already close to σ^2 and has the advantage over not constraining the initial variance to a pre-defined interval. However, it fails when the variance is poorly estimated by $\hat{\sigma}^2$. This can be the case for instance when the variance of the Gaussian is too large compared to the mean. Consider sampling 100 points from a Gaussian of mean $\mu = 0$ and variance $\sigma^2 = 20$. The variance would be poorly estimated by this technique and we could also have $N = 0$ if the interval is not large enough. If it is too large, then we would capture data from other Gaussians. Moreover, sampling from the distribution didn't improve the results significantly compared to the empirical statistic.

A much more efficient approach would be to use an informative prior. However, this is very much application dependent and seems difficult to automate.