CSC412 – Assignment #4

Due: March26, in class
Worth: 15%
Late assignments not accepted.

1 Training a Hidden Markov Model

In this question you will implement the EM algorithm for training a hidden Markov model with discrete output symbols. As you know, the model parametrizes a joint distribution over hidden states $x_t \in \{1 \dots K\}$ and observed symbols $y_t \in \{1 \dots M\}$ as follows

$$\begin{aligned} p(x_1 = k) &= \pi_k \\ p(x_t = k | x_{t-1} = j) &= Sjk \qquad t = 2 \dots T \\ p(y_t = m | x_t = k) &= A_{mk} \qquad t = 1 \dots T \end{aligned}$$

The marginal probability of an observed sequence is given by summing over all possible state trajectories:

$$p(\{y_1, \dots y_T\}) = \sum_{\{x_1, \dots x_T\}} p(\{x_1, \dots x_T\}, \{y_1, \dots y_T\})$$

The model is trained to maximize the log likelihood of a set of training sequences
$\{y_1^1, \dots y_{T_1}^1\}, \{y_1^2, \dots y_{T_2}^2\}, \dots, \{y_1^N, \dots y_{T_N}^N\}$,
given by the sum of the log likelihoods for each sequence.

1.1 What to do

- Using the data provided in `a4textdata.mat`, to train hidden Markov models with 1,2,3 and 6 states.

  The data represents character sequences from the book *Decline and Fall of the Roman Empire.* You can use the function `charascii.m` to display the data in a human readable form. Each training or testing sequence is an element of a cell array, for example the tenth training sequence is `textdata_train{10}`.

- For each number of states: (a) train on the 100 training sequences using a few different random initializations and save the parameters and random seeds from the best training run; (b) test on the remaining 50 test sequences; (c) report the average training and test log likelihood *per character.*

- Repeat the same procedure but use each training and testing sequence in reverse, i.e. train and test on *backwards English.*

- Finally, test the forwards English models you trained above on the reversed test sentences, and report the average per character log likelihood, then test the backwards English models you trained above on the forwards test sentences, and report the average per character log likelihood.

1.2 What to hand in

- A table showing the training and testing average per-character log likelihood for 1,2,3,6 state models trained on forwards/backwards sentences and tested on forwards/backwards sentences. There should be 16 numbers in your table in all.

- A graphical representation of the output distributions and transition matrices for the various models trained on forwards/backwards sentences. You may find the matlab program `chardispH.m` helpful in displaying the output distributions. Display all the output distributions for the different states of the same model on the same line, and display different models on different lines. The transition probabilities can be displayed using `imagesc` with `caxis([0,1])` and `colormap gray`. Make sure to clearly indicate whether you are displaying S_{ij} or S_{ji} since the transition matrix is not symmetric! There should be 8 transition matrices and 8 lines of output distributions in all.
- Your MATLAB code, properly commented (6 pages maximum). Please print out your code AND email it to csc412@cs since we will need to test it to make sure it works.

1.3 Some hints

- Initialize the model parameters by setting the transition and output probabilities to be roughly uniform *with a bit of random noise*. If you don't add random noise, all states in the model will receive identical updates and they will never split apart. Make sure there is enough randomness in your initialization that you sometimes fall into different local maxima when training models with more than one or two states.
- Watch out for numerical difficulties when you are coding the E-step and M-step equations for the HMM learning. You can either represent all probabilities as logarithms and use the `logsum.m` function or else use the rescaling trick mentioned in class.
- Make sure your distributions remain properly normalized. In particular, after every M-step it should be the case that $\sum_k S_{jk} = 1 \quad \forall j$ and that $\sum_m A_{km} = 1 \quad \forall k$.
- You can assess convergence of EM by monitoring the log likelihood and stopping when the fractional change `(newlik-oldlik)/abs(newlik)` is less than one part in a million. Your log likelihood should never decrease!
- You might find it interesting to plot the parameters after every iteration of EM, so you can see how things are going. You should definitely print out the log likelihood as you go so you can ensure that it always increases.

1.4 Reminders

Amongst other things, your code will need to:

- Compute the (log) probability of emitting every symbol in the training sequence under each state.

$$b_{tk} = p(y_t|x_t = k)$$

- Run the recursions to compute $\alpha_k(t)$ and $\beta_k(t)$ for all states at all times. Remember that $\alpha_k(1)$ is initialized using π_k and $A_k(y_1)$ and that $\beta_k(T)$ is initialized by a constant value.
- Compute $\gamma_k(t)$ and $\eta_{jk}(t)$ given α and β.
- Compute the average per-character likelihood after each E-step and verify that this never goes down.
- Update the parameters (S, A, π) in the M-step given γ and η.