CSCI 1950-F Homework 9: HMMs & Topic Models

Brown University, Spring 2011

Homework due at 12:00pm on May 10, 2011

We begin by learning hidden Markov models (HMMs) which describe the statistics of English text. In this application, each discrete "time" point corresponds to a single letter. For training, we use a chapter from Lewis Carroll's *Alice's Adventures in Wonderland*, available in `aliceTrainRaw.txt`. To simplify the modeling task, we first converted letters to lowercase and removed all punctuation. The resulting text, stored in `aliceTrain.txt`, is a sequence composed of 27 distinct characters (26 letters, as well as whitespace encoded via an underscore '_').

In many applications of HMMs, there is insufficient data or computational resources to select the model order via cross-validation. In these situations, the state dimension is often selected via either the *Akaike information criterion (AIC)* or *Bayesian information criterion (BIC)*. Let $y = (y_1, \ldots, y_T)$ denote the observed training sequence, $x = (x_1, \ldots, x_T)$ a hidden state sequence, and $\hat{\theta}_M$ an ML estimate of the parameters for an HMM with M states:

$$\hat{\theta}_M = \arg\max_{\theta_M} p(y \mid \theta_M) = \arg\max_{\theta_M} \sum_x p(y \mid x, \theta_M) p(x \mid \theta_M) \qquad x_t \in \{1, \ldots, M\}$$

For this model, the AIC and BIC take the following form:

$$\mathrm{AIC}_M = \log p(y \mid \hat{\theta}_M) - d(M)$$
$$\mathrm{BIC}_M = \log p(y \mid \hat{\theta}_M) - \frac{1}{2} d(M) \log(T)$$

Here, d(M) is the *number* of parameters (degrees of freedom) for an HMM with M states. The "best" model is then the one for which AIC_M or BIC_M is largest. You will need Murphy's Matlab HMM toolbox, which is distributed with the pmtk toolbox or available at

`http://people.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`

An example script providing partial solutions, as well as auxiliary functions useful for dealing with character data, are posted here:

`/course/cs195f/asgn/hmm`

WARNING: Training HMMs of multiple orders via the EM algorithm may take a few hours of computation time, so start early!

Question 1:

a) The method `dhmm_em.m`*, provided by the HMM toolbox, is an implementation of the expectation maximization (EM) algorithm for ML parameter estimation in HMMs. Use this*

package to learn HMMs with different hidden state dimensions (for example, try models with $M = 1, 5, 10, 15, 20, 30, 40, 50, 60$ states). Note that the model with $M = 1$ is a unigram model, which assumes that characters are independent. For each of these models, use a single random initialization, and run the EM algorithm for at most 500 iterations, or until the change in log-likelihood falls below 10^{-6}. Compute the log–likelihood which each model assigns to the training sequence. Save these models for later sections.

b) *Derive a formula for the number of parameters d in an HMM with M hidden states, and observations taking one of W discrete values. Remember to account for normalization constraints (for example, a discrete distribution on 4 events has only 3 degrees of freedom, since the probabilities of these events must sum to one). Plot the training log–likelihood* $\log p(y \mid \hat{\theta}_M)$, AIC_M, *and* BIC_M *versus M for the HMMs learned in part (a). Which criterion favors simpler models?*

c) *To test our learned HMMs, we use the text from a different chapter of* Alice's Adventures in Wonderland, *available in* `aliceTest.txt`. *Using* `dhmm_logprob.m`, *evaluate the test chapter's log–likelihood with respect to each HMM learned in part (a). Plot these test log–likelihoods versus M. Which model selection criterion better predicted test performance?*

d) *Using the method* `sampleText`, *generate a random 500–character sequence from four different HMMs: the model with no sequential dependence ($M = 1$), the model with the highest* BIC_M, *the model with the highest* AIC_M, *and the most complex model. Compare and contrast these sequences. What aspects of English text do they capture? What do they miss?*

In addition to computing likelihoods, HMMs lead to an efficient forward–backward algorithm which estimates the posterior probabilities of unobserved state sequences. This problem uses this method to estimate the identities of characters which have been *erased* from a text document.

Let $x_t \in \{1, \ldots, M\}$ denote the hidden state at position t, and y_t the "true" character at position t of some document. Suppose that instead of observing y, we observe an alternative sequence z in which some letters have been erased. We assume that each letter is independently erased with probability ϵ, so that

$$\mathrm{P}[Z_t = y_t \mid Y_t = y_t] = 1 - \epsilon \qquad \mathrm{P}[Z_t = * \mid Y_t = y_t] = \epsilon$$

where '*' is a special erasure symbol. Figure 1 shows a graphical model describing this generative process. Note that we never observe an "incorrect" letter; z_t is always either identical to y_t, or the erasure symbol '*'.

Question 2:

a) *Starting with the test sequence from* `aliceTest.txt`, *generate a "noisy" text sequence by randomly erasing letters with probability $\epsilon = 0.2$. See the sample script for code which does this. Print out the first 500 characters of the noisy sequence.*

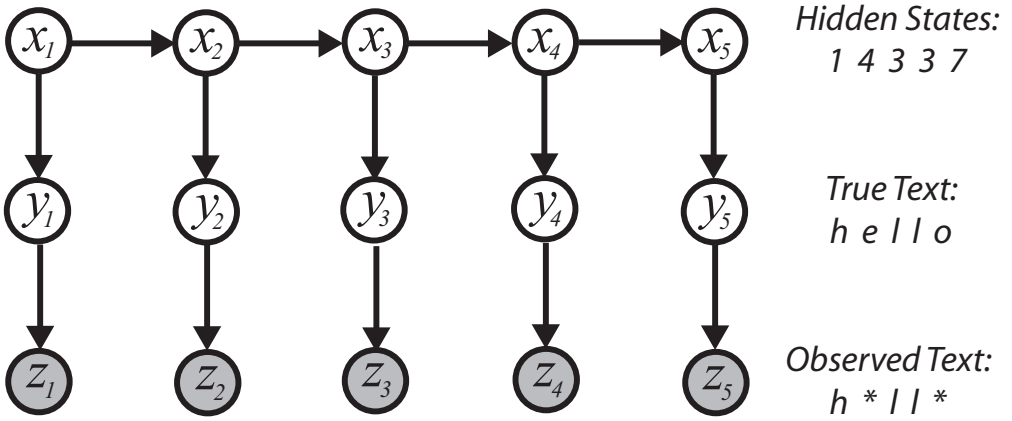

Figure 1: Graphical model illustrating an HMM with hidden states x_t which generate letters y_t. We observe a sequence z in which some of these letters have been erased.

b) Using the method `fwdback.m`*, compute the posterior distributions* $p(x_t \mid z)$ *for the four models from problem 1(d). To do this, exploit the fact that*

$$p(z_t \mid x_t) = \sum_{y_t} p(z_t \mid y_t) p(y_t \mid x_t)$$

This implies that if we sum or marginalize over the possible values of the letters y_t*, we recover a standard hidden Markov model in which the observations* z_t *are independent given the hidden state sequence* x*.*

c) Suppose that we observe a letter $z_t \neq *$ *at position* t*. Argue that this implies that* $y_t = z_t$ *with probability one.*

d) Suppose that we observe an erasure at position t*, so that* $p(z_t = * \mid y_t) = \epsilon$ *remains* constant *as* y_t *is varied (since erasures provide no information about the underlying letter). Using the factored form of the generative model (as illustrated by the graph in Fig. 1), and the form of the observation model, show that the posterior distribution of* y_t *is*

$$\begin{aligned} p(y_t \mid z, z_t = *) &\propto p(z_t \mid y_t) p(y_t \mid z_1, \ldots, z_{t-1}, z_{t+1}, \ldots, z_T) \\ &\propto \sum_{x_t} p(y_t \mid x_t) p(x_t \mid z) \end{aligned}$$

where $p(x_t \mid z)$ *is the posterior distribution of* x_t *given the* full *noisy sequence* z*.*

e) Using the marginal distributions $p(x_t \mid z)$ *from part (b), and the equation from part (d), determine the most likely missing letter for each erasure. Or, equivalently, implement the decision rule which minimizes the expected number of incorrect characters.*

f) Determine the percentage of missing letters which were correctly estimated by each model. What would chance performance be for this task? Print the first 500 characters of the denoised text produced by each model from problem 1(d), and comment on any differences.

This problem will explore the latent Dirichlet allocation (LDA) model, a popular "topic" model. Each of the K topics has a categorical distribution ϕ_k over a fixed vocabulary of W words, sampled from a symmetric Dirichlet prior:

$$\phi_k \sim \text{Dir}(\beta, \beta, \ldots, \beta), \qquad k = 1, \ldots, K.$$

Each document has a categorical distribution over the K topics, also sampled from a symmetric Dirichlet prior:

$$\theta_d \sim \text{Dir}(\alpha, \alpha, \ldots, \alpha), \qquad d = 1, \ldots, D.$$

The i^{th} word in document d, which we denote by w_{di}, is then sampled from a topic z_{di} determined as follows:

$$z_{di} \sim \text{Cat}(\theta_d)$$
$$w_{di} \sim \text{Cat}(\phi_{z_{di}})$$

Let $\phi = \{\phi_k\}_{k=1}^K$ and $\theta = \{\theta_d\}_{d=1}^D$. Let $\boldsymbol{z}$ denote the assignments of words to all topics, $\boldsymbol{z}_{-di}$ the set of all assignments excluding z_{di}, and define $\boldsymbol{w}$ and $\boldsymbol{w}_{-di}$ similarly. In the following questions, assume that the hyperparameters α and β are set to known, fixed values.

Question 3:

a) Suppose that the parameters θ and ϕ were known. Derive an expression for the conditional distribution $p(z_{di} \mid \boldsymbol{z}_{-di}, \boldsymbol{w}, \theta, \phi)$.

b) Using formulas for the marginal likelihood of Dirichlet-multinomial models (see Sec. 4.8.2 of the textbook), derive an explicit formula for the following marginalized (collapsed) prior distribution:

$$p(\boldsymbol{z} \mid \alpha) = \int_\Theta p(\boldsymbol{z} \mid \theta) p(\theta \mid \alpha)\, d\theta$$

When expressing your answer, use n_{dk} to denote the number of observed word tokens assigned to topic k in document d.

c) Again using formulas for the marginal likelihood of Dirichlet-multinomial models, derive an explicit formula for the following marginalized (collapsed) likelihood function:

$$p(\boldsymbol{w} \mid \boldsymbol{z}, \beta) = \int_\Phi p(\boldsymbol{w} \mid \boldsymbol{z}, \phi) p(\phi \mid \beta)\, d\phi$$

When expressing your answer, use m_{kw} to denote the number of instances of vocabulary word w assigned to topic k.

d) Finally, derive a formula for the following conditional probability distribution:

$$p(z_{di} \mid \boldsymbol{z}_{-di}, \boldsymbol{w}, \alpha, \beta) \tag{1}$$

This defines the conditional distributions required by the collapsed Gibbs sampler. Progressively sampling from these distributions will provide a sample approximately distributed according to the posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{w}, \alpha, \beta)$.

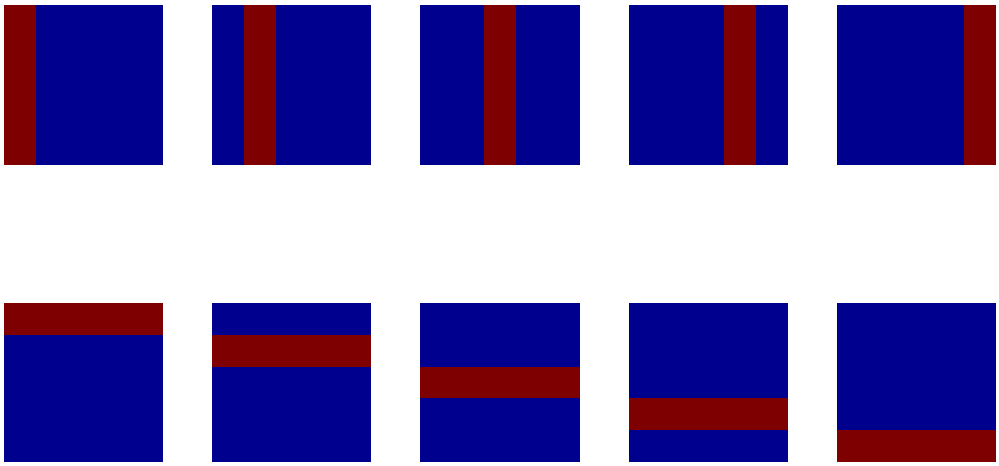

Figure 2: K=10 topics used to generate the toy bars dataset. Each topic is a distribution over 25 words, which are visualized in a 5×5 grid.

e) Implement the collapsed Gibbs sampler yourself, and run it on the artificial bars dataset, stored in `bars.mat`*. The dataset contains documents generated as a mixture of the topics in Figure 2. Set the number of topics to* $K = 10$*, and the hyperparameters as* $\alpha = 1$*,* $\beta = 1$*. From a random initialization, run the sampler for 100 iterations, where one iteration resamples all variables once. Visualize the recovered topics using* `visualizeBars.m`*. How well is the original topic structure captured?*

f) Repeat part (e) with the number of topics set to $K = 5$*. Discuss how the results change.*

g) Finally, we analyze a collection of papers from the Neural Information Processing Systems (NIPS) conference, stored in `NIPS.mat`*. Set the number of topics to* $K = 10$*, and the hyperparameters as* $\alpha = 0.1$*,* $\beta = 0.01$*. From a random initialization, run the sampler for 100 iterations, where one iteration resamples all variables once. For the final topic assignments, report the top 10 most likely words associated to each topic.*

Dataset and implementation hints Data is posted in the following directory:
`/course/cs195f/asgn/topics`
Both datasets contain two data structures, WS and DS, which are $1 \times N$ arrays. WS(i) contains the vocabulary index of the i^{th} word token, and DS(i) contains the corresponding document index. Additionally, the NIPS dataset contains 'vo', a W-dimensional cell array giving the character string associated with each vocabulary word. You will need to access vo only when displaying the top words assigned to each topic.

To sample from a categorical distribution, you can use `sample.m`, which is distributed with the PMTK toolbox. See the course locker for an example script which visualizes the bars dataset. Make sure that your Gibbs sampler returns topic-word and topic-document co-occurrence counts for the final, sampled state. You will need the topic-word counts to visualize the topics.