Machine Learning Fall 2015 Homework 5

Make sure to explain you reasoning or show your derivations. Except for answers that are especially straightforward, you will lose points for unjustified answers, even if they are correct.

General Instructions

Submit your homework electronically on Canvas. We recommend using LaTeX, especially for the written problems. But you are welcome to use anything as long as it is neat and readable.

Include a README file that describes all other included files. Indicate which files you modified. You are welcome to create additional functions, files, or scripts, and you are also welcome to modify the included interfaces for existing functions if you prefer a different organization.

Since we will work on some of the homework in class, clearly indicate which parts of your submitted homework are work done in class and which are your own work.

Relatedly, cite all outside sources of information and ideas. **List any students you discussed the homework with.**

Written Problems

1. Consider a hidden Markov model (HMM) with Gaussian emission probabilities. This type of HMM represents the joint distribution of a sequence of real valued observations $Y = \{y_1, \ldots, y_T\}$, where each observation is a real-valued vector $y_t \in \mathbb{R}^d$, and a sequence of hidden, latent states $X = \{x_1, \ldots, x_T\}$, where each latent state is a multinomial value $x_t \in \{1, \ldots, K\}$. Using a temporal interpretation of the sequence, the HMM's power comes from the assumption that the past is independent of the future given the present.

   (a) (2 point) Write the factorized probability distribution for the HMM in terms of (1) the prior state probability $p(x_t)$, (2) the transition probabilities $p(x_t|x_{t-1})$, and (3) the observation probabilities $p(y_t|x_t)$.

   (b) (2 points) Using the factorized form, write the formula for computing the joint probability of the first observation and the first latent state $p(x_1, y_1)$. In other words, show how marginalizing out all the variables except x_1 and y_1 leads to a simple formula.

   (c) (3 points) Letting $\alpha_1(x_i) := p(x_1, y_1)$, and more generally $\alpha_t(x_t) := p(x_t, y_1, \ldots, y_t)$, derive the formula for α_2 (i.e., $p(x_2, y_1, y_2)$) in terms of α_1. Note that this time you will need to marginalize out x_1.

   (d) (2 points) Show the generalization of the formula you derived in the previous subproblem: the formula for $p(x_t, y_1, \ldots, y_t)$ or $\alpha_t(x_t)$ in terms of α_{t-1}. You should get the standard forward update of the famous forward-backward algorithm.

   (e) (3 points) Returning to the full joint probability, write the formula for $p(y_T|x_{T-1})$ and simplify. We will call this quantity $\beta_{T-1}(x_{T-1})$.

   (f) (4 points) Write the formula for $\beta_{T-2}(x_{T-2})$, which will need to sum over all values of x_{T-1} (i.e., $p(y_{T-1}, y_T|x_{T-2})$), and then write the general formula for $\beta_{t-1}(x_{t-1})$ (i.e., $p(y_t, \ldots, y_T|x_{t-1})$). You should get the standard backward update of the famous forward-backward algorithm, which uses the values of $\beta_t(x_t)$. Show how this formula comes from marginalizing out the irrelevant variables.

   (g) (3 points) Using α and β, write the expression for $p(x_t, y_1, \ldots, y_T)$, or the joint probability of any one state x_t and all the observations $y_1, \ldots, y_T$. Then show the formula for the conditional probability $p(x_t|Y)$.

   (h) (6 points) When running forward-backward, in practice, one must normalize the α and β messages to avoid numerical underflow. For forward messages α, normalization amounts to computing $p(x_t|y_1, \ldots, y_t)$ instead of the joint probability $p(x_t, y_1, \ldots, y_t)$. There is not as natural an interpretation for the normalized β message $\tilde{\beta}_t(x_t) := \beta_t(x_t) / \sum_{x'_t} \beta_t(x'_t)$. Show that normalizing the α and β messages to $\tilde{\alpha}$ and $\tilde{\beta}$ then normalizing the final product still yields the correct conditional probability $p(x_t|Y)$. Hint: use the conditional definition of $\tilde{\alpha}$, and write the definition of $\tilde{\beta}$ in terms of the unnormalized β. Use a placeholder Z_t for the normalizing constant of each β_t and show that terms cancel in the last stage of the normalization.

2. (5 points) Project status report. Write a 1-2 page summary of your progress so far on the project. Describe what you have accomplished so far on your project, whether any project goals have evolved since the proposal, and what is planned for the rest of the project period.

Programming Assignment

For this programming assignment, we have provided a lot of starter code. Your tasks will be to complete the code in a few specific places, which will require you to read and understand most of the provided code, but will only require you to write a small amount of code yourself.

1. (10 points) Complete `markovChainTrain.m`, which takes a sequence of data as input and learns the maximum likelihood transitions. Given data $\{x_1, \ldots, x_T\}$ with each variable one of K states, the model should train a prior

$$p(x = i) \leftarrow \frac{1}{T + K\alpha} \left(\alpha + \sum_{t=1}^{T} I(x_t = i) \right), \forall i \tag{1}$$

and a transition probability

$$p(x_t = i | x_{t-1} = j) \leftarrow \frac{(\#\text{ time steps where state } j \text{ transitions to state } i) + \alpha}{(\#\text{ of times process is in state } j \text{ in the first } T-1 \text{ steps}) + K\alpha} \tag{2}$$

Test this function using `runObama.m`, which loads a speech transcript by President Obama and trains a Markov chain on his word usage. It should generate somewhat convincing text that sometimes reads like real English. Try running the same procedure on other text documents!

2. (12 points) Complete `myHmmInferStates.m`, which performs the forward-backward algorithm to infer the latent state probabilities of a hidden Markov model. Refer to the textbook and the class videos (and what you derived in the written section) for the formulas you'll need to implement. You are responsible for three segments of this algorithm implementation: the forward message-passing, the backward-message passing, and the fusion of the forward and backward messages to compute the marginal probabilities.

   Once you get this function working, part of the `runSyntheticExperiments.m` script will run. The first test this script runs is to evaluate the log likelihood of the synthetic data using the true models they were generated from. If your code is working, you should see that the models that generated the corresponding data sets will have the highest log likelihood (i.e., model 1 should have the highest for data 1, model 2 for data 2, and model 3 for data 3). The rest of the script will not work yet, until you finish the next step.

3. (6 points) Complete `myHmmTrain.m`. You are only responsible for a small snippet of this function: the M-step update for the Gaussian parameters. It will be very helpful to understand the rest of the provided implementation to write this snippet.

   Once you complete this step, you will be able to run the full synthetic experiment. The experiment trains HMMs on the three data types, then evaluates the log likelihood of each data set (train and test) under each learned HMM. If everything is working correctly, you should see significantly higher likelihoods for the correct HMMs. Finally, the script plots the Gaussians of each state from the trained HMMs under the Gaussians of each state from the true HMM that generated the data. The trained Gaussians should look similar to the true Gaussian if your code is working correctly.

4. (0 points) Your code should run on the real data, which is a time series of a four daily weather measurements (low and high temperature, overall precipitation, and snowfall) taken from the Blacksburg NOAA weather station over approximately 30 years. Run the script `runWeatherExperiment.m` (which should take a few minutes to complete, even if you vectorize your implementations). The script will train HMMs using different numbers of hidden states and evaluate them on held-out data. The script is not set up to run with very high numbers of hidden states, even though the initial experiments indicate that more states will yield a better fit model. Instead, once it sweeps through different numbers of hidden states, it visualizes the trained parameters of the HMM with four hidden states as well as the inferred state probabilities for the test period. The learned states have an intuitive interpretation.

5. (2 points) Prepare a short writeup describing what files you modified, any special instructions for us to understand your submission.

Table 1: Included files and brief descriptions.

File Path	Description
blacksburgWeather.mat	Weather data from Blacksburg NOAA station
obama15.txt	Transcript of 2015 State of the Union Address
synthObservations.mat	Synthetic observations from HMMs
synthStates.mat	States of sampled HMM sequence used to generate observations
trueModels.mat	HMM parameters used to generate synthetic data
src/gaussianLL.m	Script that computes the log likelihood of points under a Gaussian distribution
src/loadWordSequence.m	Function that loads a text file into a sequence of indices from a dictionary of words
src/logsumexp.m	Function that computes log(sum(exp(X))) in numerical stable manner
src/makeSyntheticData.m	Script that generates the synthetic data. You do not need to run this script since all the relevant output is saved in mat files
src/markovChainTrain.m	Function **you will complete** to train a Markov model (without hidden variables) on an observed sequence
src/myHmmInferStates.m	Function **you will complete** that performs inference via the forward-backward algorithm on an HMM
src/myHmmTrain.m	Function **you will complete** to train a hidden Markov model on an observed sequence of observations
src/plotGMM.m	Function that plots ellipses representing the Gaussians in a mixture model.
src/runSyntheticExperiment.m	Script that runs the main HMM experiment.
src/runObama.m	Script that runs the Markov chain experiment .
src/runWeatherExperiment.m	Script that runs the real-data HMM experiment, training HMMs on Blacksburg weather data.
src/sampleGaussian.m	Script that samples data from a multivariate Gaussian distribution. You won't need this, but it's used to generate the synthetic data.
src/sampleMultinomial.m	Script that samples data from a multinomial distribution. You won't need this, but it's used to generate the synthetic data.