CSC 412/2506
Probabilistic Graphical Models
Assignment 3
Out: March 3
Due: March 24

1 Training a mixture model for movie ratings [30 points]

In this assignment you will work with the MovieLens dataset from the first assignment.

First you will build a probabilistic clustering model for the data model, which has a very natural interpretation in the collaborative filtering domain. A cluster can simply be thought of as a collection of users that express similar preferences over the full set of items. In a movie recommender system, for example, clusters may reflect preferences for specific genres like action, drama, science fiction and so on.

Define a finite multinomial mixture model, a probabilistic clustering model for discrete data. In this model, an unobserved or latent variable Z_i is associated with every user i, indicating the cluster to which user i belongs. Assume that there are a fixed, finite number of clusters K. The parameters of each cluster β_k specify the preferences of a prototypical user that belongs to cluster k. Specifically, β_{vjk} is the probability $P(R_{ij} = v|Z_i = k)$ that user i will assign rating v to item j under the assumption that user i belongs to cluster k. Assume that the prior probability that a user will belong to cluster k is uniform across the clusters. The probabilistic model for this basic multinomial mixture model is defined as:

$$P(Z_i = k|\theta) = \theta_k \tag{1}$$

$$P(\mathbf{R}_i|Z_i = k, \beta) = \prod_{j=1}^{M} \prod_{v=1}^{V} \beta_{vjk}^{[r_{ij}=v]} \tag{2}$$

The square bracket notation $[s]$ represents an indicator function that takes the value 1 if the statement s is true, and 0 if the statement s is false.

You will use EM to train the mixture model. You can evaluate a trained model by conditioning on the training set ratings for each user, and predict the ratings for each of that user's test items. You should use cross-validation to tune your hyper-parameters. We have provided a test set of ratings you can use to evaluate your trained models.

Note that in the data split, movies with less than 200 ratings are removed (leaving 90 movies). Otherwise there are too many parameters and the model overfits too easily. Each row in these sets corresponds to users, and each column corresponds to movies. There is no overlap between users in the train/test set.

Also, note that this model is very sensitive to random seeds.

Here are some tips to deal with numerical issues:

- In the E-step make sure to compute everything in log-space. When it comes time to compute the responsibilities, you can use the following identity:

$$a/\sum(a) = \exp(\log(a) - \text{logsumexp}(a))$$

- For the M-step there is a risk that the denominator becomes 0 (this can happen with some of the β values). You can add $1e{-}32$ to make sure it is not exactly 0. Then, if you do the $1e-32$ trick, you should make sure the β values still normalize across ratings. You can do this by adding $1e-32$ to each new β value, and then re-normalizing.
- You should make sure that after implementing any tricks that the log-likelihood still monotonically increases with each EM step (on the training set).
- For python users, they can find logsumexp implemented in the scipy.misc package. For matlab users you can find it in implemented in Mark Schmidt's minfunc package.

(A). [6 points] Evalute different numbers of mixture components and select the best setting using cross validation. Produce a bar chart that describes the performance of the model on the held-out ratings.

(B). [3 points] Run several random restarts, and examine the final log likelihoods. How consistent are your results across different initializations?

(C). [3 points] For your best number of components, produce a plot of the expected complete data log likelihood as a function of EM interation.

(D). [3 points] Examine the mixture components. Did the model find anything interesting in user clusters? Compare them to the information about the users available in the Movielens data, such as their gender, occupation, and age.

(E). [15 points] (Written part.)

Now we will extend the finite multinomial mixture model by introducing a prior over the β parameters. The probabilistic model for this extended multinomial mixture model is defined as:

$$P(\beta_{jk}|\phi_k) = \mathcal{D}(\beta_{jk}|\phi_k) \tag{3}$$

$$P(Z_i = k) = \theta_k \tag{4}$$

$$P(\mathbf{R}_i|Z_i = k, \beta) = \prod_{j=1}^{M}\prod_{v=1}^{V} \beta_{vjk}^{[r_{ij}=v]} \tag{5}$$

where we have adopted a Dirichlet distribution, denoted by $\mathcal{D}$, for the prior over rating predictions.

Derive the EM updates in this extended model. What are the potential advantages of introducing this Dirichlet prior?

2 Comparing approximate inference methods [70 points]

Now you will extend the model to include latent factors for movies. Movies also have natural underlying features, such as genre. One way to model both users and movies is with a directed graphical model, where the latent features are decomposed into user-specific and movie-specific factors U_i and V_j.

Each rating can then be modeled as a noisy version of the inner product (where we have switched to Gaussian observation noise):

$$r_{ij} \sim \mathcal{N}(\mathbf{u}_i^T \mathbf{v}_j | 0, \sigma)$$

The prior distributions over these latent factors are D-dimensional diagonal Gaussians:

$$p(U|\sigma_U) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U I) \tag{6}$$

$$p(V|\sigma_V) = \prod_{j=1}^{M} \mathcal{N}(U_j|0, \sigma_V I) \tag{7}$$

The predictive distribution for this model can be obtained by marginalizing over the model parameters and hyperparameters, given some set of observed ratings $\mathbf{r}^o$:

$$p(r_{ij}|\mathbf{r}^o) = \int\int p(r_{ij}|U_i, V_j) p(U, V|\mathbf{r}^o, \sigma_U, \sigma_V) d(U, V)$$

Due to the complexity of this posterior distribution over U and V, we need to use some approximate inference method to evaluate this predictive distribution. In this question you will explore various approximate inference methods for this model.

There is some code online for this model that you may find useful; you can follow the pointers from https://github.com/chyikwei/recommend. Note that this code has some bugs – we suggest you use it as a guide for your own code rather than completely relying on it.

(A). [15 points] First approximate the complicated posterior with a point estimate. You can get point estimates by maximizing $p(U, V|\mathbf{r}^0, \sigma_u, \sigma_v)$, and then estimate log-likelihood using these U and V by directly computing $p(r_{ij}|U, V)$. Experiment with the latent dimension D, trying a few values in the range of 1 to 20. You will need to tune the Gaussian standard deviations, which effectively act as regularization parameters. Plot how the likelihood of held-out ratings—in training, validation and test—varies as a function of the observed ratings. Also examine the mean squared error between the true ratings and the mean of your predictive distribution, for the different data splits.

(B). [15 points] (Written part). Rather than simply using point estimates for U and V, we can adopt a variational approach. For this part we suggest a mean-field variational approximation. You need to define factors to approximate the posterior $p(U, V|\mathbf{r}^o, \sigma_U, \sigma_V)$, and then derive update equations for those factors, and for the resulting predictive distribution.

(C). [20 points] Experiment with the variational approximation, using settings and tasks similar to what you did in part A. You can make predictions by computing the mode of $E_q[\log p(r_{ij}|U, V)]$; estimating the likelihood $p(r_{ij}|\mathbf{r}^o)$ directly is hard if we want to integrate over $q(U)$ and $q(V)$, but taking the log simplifies things.

(D). [20 points] An alternative is to use MCMC to approximate the predictive distribution:

$$p(r_{ij}|\mathbf{r}^o) = \frac{1}{S}\sum_{s=1}^{S} p(r_{ij}|U_i^{(s)}, V_j^{(s)})$$

You can generate the samples via Gibbs sampling. Here you can make predictions by averaging the means of $p(r_{ij}|U_i^{(s)}, V_j^{(s)})$ over the samples.

You can find some instructions on how to formulate Gibbs sampling for this model in the *Bayesian Probabilistic Matrix Factorization using MCMC* paper included in the assignment zip file.

Again track the squared-error objective across epochs of training, and validation. Also, you should investigate performance of your best model as a function of the number of samples.

For this assignment, turn in a pdf, by 6pm on March 24. Include a few plots that accurately represent the experimental results you obtained. Also turn in the EM derivation from Section 1B, and a description of your variational approximation from Section 2B. Finally, you should also include a few observations about the models, and a brief comparison of their behavior.