University of Washington
Department of Computer Science and Engineering / Department of Statistics

CSE 547 / Stat 548 Machine Learning (Statistics) for Big Data

Homework 4

Winter 2014

Issued: Tuesday, February 25, 2014 **Due:** Thursday, March 6, 2014

Suggested Reading: Assigned Readings in Case Study III and IV (see website).

Problem 4.1
[75 points]
Comparing ALS and Gibbs Sampling for Matrix Factorization
In this problem, you will look at a portion of the Netflix movie rating dataset, which has ratings that users have given to movies. In this example, we have a matrix $X \in \Re^{n\times m}$ for $n = 50$ users and $m = 20$ movies. Although many of the entries of the full dataset are zero, since users do not rate every movie, this particular subset has ratings for every user/movie pair. We have held out 25% of the existing ratings as a validation set as well 25% as a test set in order to assess performance of different methods for matrix factorization.The remaining 50% of the data will be used as the training set. These files can be found in the zip file "submatrix.zip" in matrix market format. The goal is to find $L \in \Re^{n\times k}$ and $R \in \Re^{m\times k}$ that best predict the true ratings $r_{u,v}$ in the test set. To prevent overfitting on the training set, we will regularize L and R via the L2 norm. Our objective function, then, is to find L and R such that

$$f(L,R) = \frac{1}{2}\sum_{u,v:r_{u,v}\neq 0}(r_{u,v} - L_u \cdot R_v)^2 + \frac{\lambda_u}{2}\|L\|_2^2 + \frac{\lambda_v}{2}\|R\|_2^2 \tag{1}$$

is minimized on the training set. We can then use the validation set to help choose λ_u and λ_v before predicting the test set (which would normally be unknown).

(a) As was discussed in class, one algorithm that solves this optimization problem is Alternating Least Squares (ALS). Recall that this algorithm iteratively solves the minimization over L and R, holding the other fixed. Conveniently, this can be done independently for each row of L and R. So our algorithm is to iterate over the following 2 steps until convergence.

1. Holding R fixed, for each row L_u of L, find

$$\arg\min_{L_u} g(L_u; X, R). \tag{2}$$

2. Holding L fixed, for each row R_v of R, find

$$\arg\min_{R_v} h(R_v; X, L). \tag{3}$$

i. [**5 points**] Write down $g(L_u; X, R)$, the objective function to be minimized for a single user in step 1 of the algorithm above.

ii. [**15 points**] Use ALS to find the L and R that minimize the objective. Use $k = 5$ as the latent dimensionality, and $\lambda_u = \lambda_v = \lambda$ for $\lambda = 0, .1, 1, 10, 100$. Initialize L and R to all 1's. Run ALS for 50 iterations for each value of λ.
(*Running time: less than a minute*)

* Plot $f(L, R)$ vs iteration for $\lambda = 1$.
* Plot the RMSE of the training set and the validation set against λ (on the same plot).
* Based on your results from the previous part, choose the λ that you will use on the test set and report the RMSE on the test set using that value of λ.

The equation for calculating RMSE is

$$RMSE(L, R, X) = \left(\frac{1}{\{\#X_{u,v} \neq 0\}} \sum_{u,v:X_{u,v}\neq 0} (X_{u,v} - L_u \cdot R_v)^2 \right)^{1/2}. \quad (4)$$

iii. [**5 points**] What value of λ performs best on the training set? Does this match your expectations? Why or why not?

iv. [**5 points**] For the sake of comparison to part (b), report the validation and test RMSE for $\lambda = 1$.

(b) A Bayesian approach to this problem uses Gibbs sampling to approximate the full posterior of the parameters of interest rather than getting a single estimate. In this setting, we model the L and R matrices as (independently for each user and movie):

$$L_u \sim N(\mu_L, \Sigma_L), \quad u = 1, \ldots, n, \quad (5)$$
$$R_v \sim N(\mu_R, \Sigma_R), \quad v = 1, \ldots, m. \quad (6)$$

Then, conditional on L and R, we model our observations

$$r_{u,v} \sim N(L_u \cdot R_v, \sigma_r^2). \quad (7)$$

i. [**5 points**] Write down the full conditional distributions for L_u and R_v.

ii. [**5 points**] With $k = 5$ as in part (a) and assuming $\Sigma_L = \Sigma_R = \sigma^2 \mathbf{I}_{k\times k}$, with $\sigma^2 = 0.25$, specify the values of μ_L, μ_R, σ_r^2 for the model corresponding to the objective function (1) you minimized in part (a) with the value $\lambda = 1$.

iii. [**15 points**] Using the parameter values you came up with in (ii), run the Gibbs sampler for 300 iterations and plot the log probability on the training data vs iterations. Using the single sample corresponding to the maximum log probability, i_{max}, predict the validation data and report $RMSE(L^{*(i_{max})}, R^{*(i_{max})}, X^{\text{validation}})$. Also predict the test data and report $RMSE(L^{*(i_{max})}, R^{*(i_{max})} X^{\text{test}})$.
Initialize your L and R matrices from a sample directly from the prior.
(*Running time: may be up to 5 minutes*).

iv. [**5 points**] In theory, what is the relationship between L^* and R^* found in part (a) with $\lambda = 1$ and $L^{*(i_{max})}, R^{*(i_{max})}$ found in part b(iii), where in part (a) L^*, R^* is the ALS solution and in part b(iii) $L^{*(i_{max})}, R^{*(i_{max})}$ correspond to the sample maximizing log-probability?

v. [**5 points**] Using the Gibbs samples from (iii), throw out the first 150 samples as burn in and compute an average RMSE with the remaining 150 samples on the validation data.

$$RMSE_{ave} = \frac{1}{150} \sum_{i=151}^{300} RMSE(L^{*(i)}, R^{*(i)}, X^{\text{validation}}) \tag{8}$$

vi. [**5 points**] Note that part (iii) provides a way of determining a point estimate for the completed matrix by using the L and R that maximize the log probability. We might also consider using the point estimate given by the posterior expectation of the completed matrix LR^T:

$$\mathbb{E}[LR^T|\phi, X^{\text{train}}] = \int_{L,R} LR^T P(L, R|\phi, X^{\text{train}}) dL dR \tag{9}$$

In terms of $L^{*(i)}$ and $R^{*(i)}$, write an expression for the Monte Carlo estimate of the posterior expectation of LR^T, using only the samples after burn-in.

vii. [**5 points**] Using the ratings estimates given by the Monte Carlo estimate from part (vi), report the RMSE on the validation data and the test data.

Problem 4.2
[25 points]
Collaborative Filtering and Clustering on the Netflix Dataset

**Matrix notation: We will use subscript(superscript) to denote the column(row) of a matrix; e.g. for any matrix X, X_j is the jth column, X^i is the ith row, and $X_{i,j}$ is the entry (i,j).*

This problem contains 2 parts. In the first part, you will learn to use GraphChi (`graphchi.org`) to perform matrix factorization on the Netflix Dataset. In the second part, you will use the latent factor matrix from part 1 to perform kmeans and spectral clustering.

The famous Netflix dataset contains 100M ratings from $n = 480K$ users to $m = 17770$ movies. Let X be the $n \times m$ rating matrix, and k be the number of latent factors, the goal is to find $U \in R^{n\times k}$ and $V \in R^{m\times k}$ such that

$$f(U,V) = \frac{1}{2}\sum_{i,j:X_{i,j}\neq 0}\left(X_{i,j} - U^{i^T}V^j\right)^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 \tag{10}$$

is minimized.

One algorithm that solves the above optimization problem is the Alternating Least Squares method discussed in class. This algorithm is already implemented in GraphChi.

- Download “netflix_mm.gz” from the course website.
- Run “gunzip netflix_mm.gz” and you will get the rating matrix in sparse Matrix Market format.
- Follow the instructions at `bickson.blogspot.co.il/2013/02/setting-up-java-graphchi-development.html` to setup graphchi-java.

The latent dimension k trades off computation with accuracy.

(a) **[15 points]** Run ALSMatrixFactorization with $nshards = 10$ and $k = 2, 4, 6, 8, 10$.

  – Plot the training RMSE against k.
  – Plot the engine runtime against k.

The resulting latent factor matrices U and V are stored in “netflix_mm_U.mm” and “netflix_mm_V.mm” in sparse Matrix Market format.

In the next part, you will use the latent factor matrix V as movie features, and cluster the movies using Kmeans. If you use Matlab, you do not need to implement anything – it is all handled by the starter code. For simplicity and better interpretability, we only cluster the first 3000 movies. (The first 3000 rows in matrix "netflix_mm_V").

(b) **[10 points]** Run "kmeans_run.m" and report the 20 closest movies for each cluster. (No coding required, if using the starter code.)