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

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

Homework 4
Spring 2015

Issued: Saturday, May 16, 2015 **Due:** Thursday, May 28, 2015

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

Problem 4.1
[70 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 $X \in \mathbb{R}^{n\times m}$ for $n = 216$ users and $m = 100$ 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 $L \in \mathbb{R}^{n\times k}$ and $R \in \mathbb{R}^{m\times k}$ that best predict the true ratings $X_{u,v}$ in the test set. To prevent overfitting on the training set, we will regularize L and R via the ℓ_2 norm. Our objective function, then, is to find L and R such that

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

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. [**4 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 $\lambda = 0, 0.1, 1, 10, 100$. Initialize L and R to all 1's (with $\lambda = 0$, add some Uniform$(-0.01, 0.01)$ noise to avoid issues with inversion of singular matrices). Run ALS for 50 iterations for each value of λ.
(*Running time: a few minutes*)

   * Plot $\mathcal{L}(L, R)$ 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. [**4 points**] What value of λ performs best on the training set? Does this match your expectations? Why or why not?

iv. [**4 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

$$X_{u,v} \sim N(L_u \cdot R_v, \sigma_\epsilon^2). \quad (7)$$

i. [**6 points**] Write down the full conditional distributions for L_u and R_v.

ii. [**4 points**] With $k = 5$ as in part (a) and assuming $\Sigma_L = \Sigma_R = \sigma^2 \mathbf{I}_{k \times k}$, with $\sigma^2 = 0.1$, specify the values of $\mu_L, \mu_R, \sigma_\epsilon^2$ for the model corresponding to the objective function (1) you minimized in part (a) with the value $\lambda = 10$.

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: around 10 minutes*).

iv. [**4 points**] In theory, what is the relationship between $\hat{L}$ and $\hat{R}$ found in part (a) with $\lambda = 10$ and $L^{(i_{max})}, R^{(i_{max})}$ found in part b(iii), where in part (a) $\hat{L}, \hat{R}$ 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. [**4 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:

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

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
[30 points]
Collaborative Filtering with Graphlab Create
For this problem, we will use Graphlab Create to run parallel matrix factorization. For this whole problem, we will use the formulation that does not address cold start—namely the one that just looks at the ratings matrix (Slide 11, small numbers, on the SGD for matrix factorization, NMF, and cold start slides[1]). Make sure to install GraphLab Create[2] and download the starter code for this part (mf_starter.py).

Graphlab Create has data structures that allow you to manipulate data that do not fit into the computer's memory. We will be using one of those structures, called SGraph. To make things simpler, we have already transformed the training data you used in the previous question into an SGraph, called "training_graph.sgraph".

The graph is constructed as follows: there is an edge between a user u and a movie v if u rated v. The edge has value r_{uv}. Instead of doing node-centric computations (as Carlos presented in class), we will be using a newer feature in Graphlab: edge-centric computations. We will be using the triple_apply function (read carefully about it[3]), where we apply a function to each edge and its associated nodes. Each edge is visited once and in parallel. The point of this function is to modify vertex data—and modification to vertex data is protected by a lock. In this case, the vertex data we will be modifying are the latent factors L and R for the user and movie associated with the edge.

(a) [**6 points**] Can we perform the SGD updates for two edges r_{ab} and r_{cd} in parallel without any interference (i.e. as if we had run the updates sequentially) in the following cases? Explain why or why not.

   - $a = 1, b = 1 , c = 2 , d = 1$
   - $a = 2, b = 1 , c = 1 , d = 2$
   - $a = 1, b = 2 , c = 1 , d = 3$

(b) [**4 points**] If we implement SGD using the triple_apply function, is the result we get equivalent to running SGD in some sequential order (i.e. do we have sequential consistency)? Explain why or why not.

Implement SGD for matrix factorization, completing the *sgd_update* function in the starter code. Run it for 10 passes over the dataset, using $\eta = 0.1$, $k = 5$ (latent factor dimension), with $\lambda_u = \lambda_v \in [0, 0.001, 0.01, 0.1, 1]$.

(c) [**8 points**] Plot the RMSE on the validation data for the model you learn after the 10 passes for each value of λ (use a logscale on the x axis). Which λ would you choose in this case?

[1]`http://courses.cs.washington.edu/courses/cse547/15sp/slides/sgd-nmf-coldstart-annotated.pdf`
[2]`https://dato.com/download/`
[3]`https://dato.com/products/create/docs/generated/graphlab.SGraph.triple_apply.html`

(d) [**8 points**] Plot the RMSE on the training data after each pass over the data for the value of λ you chose in the previous question (including the RMSE before you do any pass). Does the result make sense?

(e) [**4 points**] Report the RMSE on the test data for the value of λ you chose, after the 10 passes over the data.