CSCI B659: Machine Learning through Approximate Inference in Graphical Models
Spring 2023
Roni Khardon
Assignment 2

In this assignment you will implement and test the variational probabilistic matrix factorization algorithm that you analyzed in assignment 1. The assignment is due by Wednesday 2/22, 11:59pm on canvas as a ZIP file. Submission instructions below.

Implementation

Please read the paper carefully for implementation details. Your algorithm will optimize the variational parameters (means $\bar{u}_i$ and $\bar{v}_j$ and covariance matrices Φ_i, Ψ_j) and model parameters (vector of variances σ_l^2 and scalar variance τ^2). Note that (as specified on page 4) we keep $\rho_l^2 = 1/d$ where d is the dimension or rank of the approximation (n in the paper).

Initialize covariance matrices Φ_i and Ψ_j to the identity matrix, and σ_l^2, τ^2 to 1. Initialize all entries in the mean vectors by sampling independently from a normal distribution with mean 0 and variance 1.

Run the variational EM algorithm for up to 100 iterations. For the E step the algorithm iterates over rows of U and V updating one at a time using Eq 17,18 or 20,21. For the M step the algorithm updates the parameters using Eq 22 and 24. You can, but are not required to, implement the space saving technique that avoids storage of Φ_i which is explained in section 4.1 of the paper.

The algorithm can stop early if the maximum over all i,j of the difference in $\|\bar{u_i} - \bar{u_i}^{old}\|_2$ and $\|\bar{v_j} - \bar{v_j}^{old}\|_2$ between iterations is smaller than 0.01.

For prediction for entry i, j we follow the paper and use the inner product between $\bar{u}_i$ and $\bar{v}_j$. For any predictor we measure the RMSE on the corresponding dataset, i.e., $\sqrt{\frac{1}{N}\sum_i(\hat{t_i} - t_i)^2}$ where the dataset has N examples and t_i, $\hat{t}_i$ are the true and predicted value for the i'th example.

Baseline

In addition to the matrix factorization algorithm implement the following baseline from the training set only. **Preparation:** (1) Calculate c: the mean score of all data points in the training set. (2) For each user i calculate a_i: the mean of ratings for that user. (3) For each item j calculate b_j: the mean of scores for that item. (4) If user i or item j do not have any ratings their score is set to c. **Prediction:** To predict an entry i, j predict the average of a_i and b_j.

Evaluation

We have provided 3 datasets via canvas: 40-20-2, 500-500-3, ml-100K where the first two are generated from the model (with corresponding users, items and rank) and the third is the 100K movie-lens dataset (a relatively small real world dataset which was widely used for evaluating algorithms) which is reformatted to have the same notation. You can use 40-20-2, which is smaller, to debug your implementation and then run the evaluation on the other two datasets.

The first row in each dataset gives the number of (users, items, examples) in the file. Then the examples follow where each row gives an entry in the matrix in the form $i, j, score$. The rows in the file are already randomized. Split the data into two equal parts, train on the first and test on the second part.

For 500-500-3 run the algorithm with rank d=3 and plot the train and test RMSE as a function of iterations as well as the performance of the baseline. Does the algorithm converge? and how does it compare to the baseline? Then run the algorithm with rank 1,2,3,5,10,20 and tabulate or plot the performance as a function of rank (at convergence or 100 iterations). How does the performance vary with the selected rank? how does the run time behave as a function of rank (consider both total time, and time per iteration)?

For ml-100k perform the same evaluation but with d=5 for the first part and ranks 1,2,3,5,20,100 for the second part.

Note that the evaluation will require some time (especially for large d) so please plan your time accordingly.

Useful Values for Debugging with the 40-20-2 Dataset

You can expect a baseline error of ~ 1.6. For this dataset, the MF algorithm seems to have multiple optima with error of ~ 1.56 or ~ 1.33 depending on initialization. For the large datasets the results are stable so you do not need to worry about multiple restarts.

To further help debug your code, if you initialize all entries in mean vectors to 1 (instead of random values), then after one iteration of E-step and M step updates you can expect values as follows. First two entries of u: [0.26 0.26] [0.16 0.16], first two entries of v: [0.09 0.09] [0.17 0.17], σ^2: [0.39 0.39], and τ^2: 1.74. Note, however, that the random initialization is important to break symmetry and yield good performance, so only use this option for debugging.

For Extra Fun and Credit

Experiment with different initialization strategies as proposed in the paper and test the effect on the performance of the algorithm. Do these affect convergence speed? Do they affect the outcome?

Submitting Your Assignment

Please pack the following 3 items into a directory: `YourNameHW2/`, pack the directory into `YournameHW2.zip` and upload the zip file on canvas. The items are:
(1) Please write a report on the experiments and results as requested above and put this into a PDF file - please call it `report.pdf`.
(2) The code, including a README file/portion that will tel me how to run it. The code should assume that data is in sub-directory `hw2data/`. Please make sure the code is well structured and easy to read (i.e., document it as needed). This portion can be a single file or multiple files.
(3) One PDF file which includes a "printout" of: the report, the code and README so that I do not need to print many different files for grading. Please call this `YourNameHW2Everything.pdf`.

Submission notes: Python is preferred and Jupyter notebooks are fine. In fact, combining all 3 items into a Jupyter notebook and its PDF printout can make submission easy, but this is not required.

Grading Notes: I will use item 3 as the main tool for grading so submission of the "everything" file is crucial. The assignment will be graded based on the clarity of the code, its correctness, and the presentation and discussion of the results. I may or may not run your code. Following the naming conventions will make the management of grading a lot easier so please remember to follow it.