Hong Kong University of Science and Technology
COMP 5212: Machine Learning
Fall 2016

Project 3
Due: 30 November 2016, Wednesday, 11:59pm

1 Objective

The objective of this project is twofold:

1. To acquire a better understanding of matrix factorization and collaborative filtering by implementing a version of probabilistic matrix factorization (PMF).
2. To investigate how the performance of PMF is affected by varying some hyperparameters of the model and the degree of sparsity of the data.

2 Major Tasks

The project consists of the following tasks:

1. To implement PMF with maximum a posteriori (MAP) estimation.
2. To conduct empirical study to compare the performance of PMF by varying the number of latent factors K and the regularization hyperparameters λ_u and λ_v.
3. To conduct empirical study to compare the performance of PMF on dense and sparse data.
4. To write up a project report.

Each of these tasks will be elaborated in the following subsections.

2.1 PMF with MAP Estimation

Matrix factorization (MF) is one of the most commonly used approaches in all sorts of recommender systems (e.g., for book recommendation, music recommendation, and friend recommendation). In this project, you are asked to implement a variant of MF, called PMF, from scratch.

As discussed in class, the learning of PMF with MAP estimation is equivalent to minimizing the sum of squared errors with quadratic regularization terms. The sum of squared errors corresponds to the likelihood term and the zero-mean spherical Gaussian priors correspond to the regularization terms.

The learning of PMF involves iterative update of $\mathbf{U}$ and $\mathbf{V}$ with fixed hyperparameters. Although the objective function is not jointly convex with respect to $\mathbf{U}$ and $\mathbf{V}$, it is convex with respect to $\mathbf{U}$ when $\mathbf{V}$ is fixed and with respect to $\mathbf{V}$ when $\mathbf{U}$ is fixed. Given this property, we can apply block coordinate descent to alternate between the update of **U** and **V** via update rules corresponding to gradient descent. In order to keep the algorithm *linear* in the number of observed ratings, sparsity of the rating matrix should be taken advantage of when you implement the model.

In this project, we will use the MovieLens 100K dataset[1] to evaluate the performance of PMF.[2] The dataset consists of 100,000 ratings from 943 users on 1,682 movies. The dataset will be split into three disjoint sets: a training set, a validation set, and a test set. For example, you will use 80% of the dataset to perform cross validation to obtain the best hyperparameters, similar to Project 1. After that, you can use the optimal hyperparameters to train the PMF on 80% of the dataset and evaluate the performance on the test set (i.e., the remaining 20%).

Different from the objective function, we use root mean squared error (RMSE)[3] as the evaluation metric. In the report you will have to show the RMSE during cross validation and testing.

You are expected to do the implementation all by yourself from scratch so you will gain a better understanding of the method. MATLAB/Octave is the preferred language choice which facilitates fast prototyping, possibly at the expense of run-time efficiency.[4] You may also use some other programming language, such as C++, Java, Python, or R, if you so wish, but then you may not be able to take advantage of the powerful and convenient matrix manipulation capabilities and built-in functions provided by MATLAB.

2.2 Sensitivity to Hyperparameters

In this part, you will investigate how the performance of PMF changes by varying the hyperparameters. The dataset will first be split into a training/validation set (80% of the data) and a test set (20%). Cross validation will be performed on the training/validation set to determine the best values for the hyperparameters K, λ_u, and λ_v. For simplicity, the following scheme will be used:

1. **Choosing regularization hyperparameters**: First we set $K = 2$. The generalization performance of PMF is estimated for each candidate value of $\lambda_u \in \{0.1, 1, 10, 100\}$ and $\lambda_v \in \{0.1, 1, 10, 100\}$ (16 combinations in total). This is done by randomly sampling 80% of the training/validation data (that is 64% of the whole dataset) to train PMF and then testing it on the remaining 20% of the training/validation data. Five such random data splits are performed and the average over these five trials is used to estimate the generalization performance. The values λ_u^* and λ_v^* that give the best performance among the 16 configurations can then be found.

2. **Choosing the number of factors**: By setting $\lambda_u = \lambda_u^*$ and $\lambda_v = \lambda_v^*$, you will estimate the generalization performance of PMF for each candidate value of $K \in \{1, 2, 3, 4, 5\}$. A procedure similar to '**choosing regularization hyperparameters**' above can be used to find the optimal value K^*.

Having found λ_u^*, λ_v^*, and K^*, you can set λ_u, λ_v, and K to their corresponding optimal values

[1]You can download the dataset at `http://files.grouplens.org/datasets/movielens/ml-100k.zip`. Note that only the file `u.data` is needed for this project.

[2]Other larger MovieLens datasets are in `http://grouplens.org/datasets/movielens/`. We encourage you to try them out as well. To keep the project simple, however, here we only use the smallest dataset.

[3]See `http://www.kaggle.com/wiki/RootMeanSquaredError` for details.

[4]When we refer to MATLAB hereafter, we mean either MATLAB or Octave.

and train PMF on the whole training/validation set. The learned **U** and **V** are then used to evaluate the generalization performance on the test set.

2.3 Dense and Sparse Data

In the previous subsection, evaluation of PMF is performed on a dense training/validation dataset. In practice, the data available for model training is often much sparser, especially for newly built web services. In this part of the project, you will repeat all the experiments with a different data split. Specifically, you will use only 20% of all the data to form the training/validation set and the remaining 80% as the test set. The hyperparameters K, λ_u, and λ_v will be chosen by following the same procedure as in Section 2.2 before evaluating the trained PMF on the test set. You are expected to compare the performance of PMF on the dense data (from the previous part) and on the sparse data (from this part) in your report.

2.4 Report Writing

In your report, you are expected to present the parameter settings and the experiment results. Besides reporting the recommendation performance (for both training/validation and test data) in numbers (using RMSE), graphical aids should also be used to compare the performance of different configurations visually. For the CPU time information, you may just report it in numbers.

3 Some Programming Tips

As is always the case, good programming practices should be applied when coding your program. Below are some common ones but they are by no means complete:

- Using functions to structure program clearly
- Using meaningful variable and function names to improve readability
- Using indentation
- Using consistent styles
- Including concise but informative comments

For MATLAB in particular, you are highly recommended to take full advantage of the built-in functions which can keep your program both short and efficient. Note that using loops to index individual elements in matrices and arrays should be avoided in MATLAB as much as possible. Instead, block indexing without explicitly using loops is much more efficient. Proper use of these implementation tricks often leads to speedup by orders of magnitude.

4 Project Submission

Project submission should only be done electronically using the Course Assignment Submission System (CASS):

`http://cssystem.cse.ust.hk/UGuides/cass/student.html`

There should be two files in your submission with the following naming convention required:

1. **Project report** (with filename `report`): preferably in PDF format.
2. **Source code and a README file** (with filename `code`): all necessary code for running your program as well as a brief user guide for the TA to run the programs easily to verify your results, all compressed into a single ZIP or RAR file. The data should not be submitted to keep the file size small.

When multiple versions with the same filename are submitted, only the latest version according to the timestamp will be used for grading. Files not adhering to the naming convention above will be ignored.

5 Grading Scheme

This project will be counted towards 9% of your final course grade. The maximum scores for different tasks are as follows:

- Implementation of PMF [30 points]
- Empirical study on hyperparameters [20 points]
- Empirical study on sparse data [20 points]
- Project report [30 points]

Late submission will be accepted but with penalty. The late penalty is deduction of one point (out of a maximum of 100 points) for every minute late.

6 Academic Integrity

Please read carefully the relevant web pages linked from the course website.

While you may discuss with your classmates on general ideas about the project, your submission should be based on your own independent effort. In case you seek help from any person or reference source, you should state it clearly in your submission. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.