Labs
Machine Learning Course
Fall 2019

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
www.epfl.ch/labs/mlo/machine-learning-cs-433

Problem Set 10, Nov 21, 2019
(Matrix Factorizations and Recommender Systems)

Goals. The goal of this exercise is to

- Build a recommendation system.
- Learn to evaluate its performance.
- Implement and understand matrix factorization using SGD.
- Implement and understand the alternating least-squares (ALS) algorithm.
- Choose the appropriate number of *factors*, as well as regularization parameters.
- Compare your system to a few baselines.

Setup, data and sample code. Obtain the folder `labs/ex10` of the course github repository

github.com/epfml/ML_course

We will use the dataset `movielens100k.csv` in this exercise, and we have provided sample code templates that already contain useful snippets of code required for this exercise.

1 Visualization and Creating Train-Test Splits

Since our goal is to predict the *unseen* ratings, we can create a test set by "punching holes" in the matrix, i.e. we randomly select some of the ratings in the matrix as test points, while we leave the rest for training.

Figure 1 shows the number of ratings for each user and each movie. We can see that the number of ratings varies quite a lot among users and movies. This is very typical of datasets for recommendation systems, and Big Data in general.

Exercise 1:

Fill in the notebook function `split_data` to split dataset into train and test. We only consider users and movies that have more than 10 ratings, and will discard all other users and movies. Among the remaining valid ratings, you should randomly select a rating to become a test rating with probability 10%, and training with prob. 90%. This way, we keep some minimum amount of training data for users and movies that have few ratings.

We can visualize the resulting split as in Figure 2.

2 Performance Evaluation

We will use the root mean-square error (RMSE) to measure the performance. Given true test ratings x_{dn} for the d'th movie and n'th user, and our prediction $\hat{x}_{dn} = \boldsymbol{W}_{d:}\boldsymbol{Z}_{:n} = (\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}$, we compute the RMSE as follows:

$$\mathrm{RMSE}(\boldsymbol{W}, \boldsymbol{Z}) := \sqrt{\frac{1}{|\Omega|} \sum_{(d,n)\in\Omega} \tfrac{1}{2}\big[x_{dn} - (\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}\big]^2} \tag{1}$$

and $\boldsymbol{W} \in \mathbb{R}^{D\times K}$, $\boldsymbol{Z} \in \mathbb{R}^{N\times K}$. Here $\Omega \subseteq [D] \times [N]$ is the set of the indices of the observed ratings of the input matrix $\boldsymbol{X}$. The RMSE can be computed either on a training set Ω, or on a hold our hold-out test set Ω.

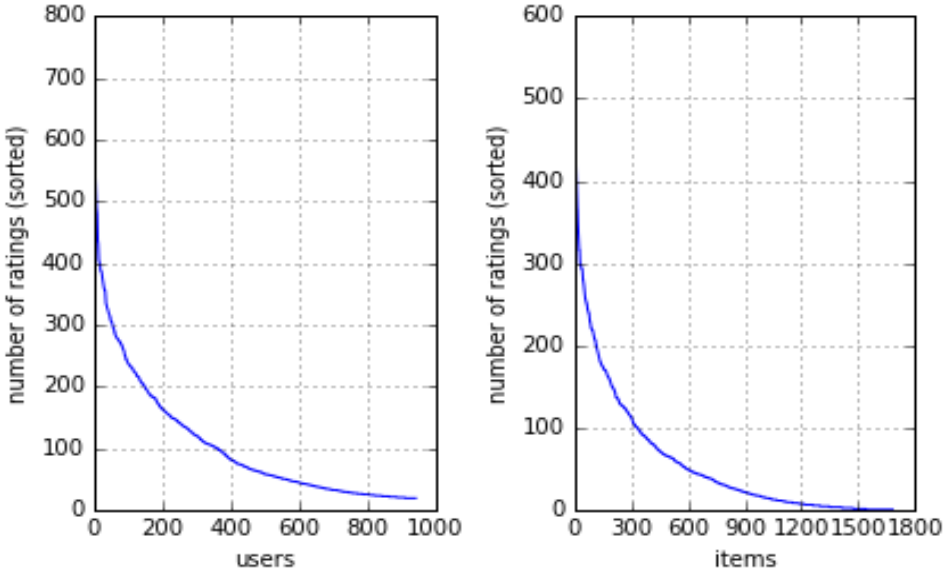

Figure 1: The left plot shows the (ordered) number of ratings for each user, while the right plot shows the (ordered) number of ratings for the movies. Both numbers are showed in descending order for clarity.

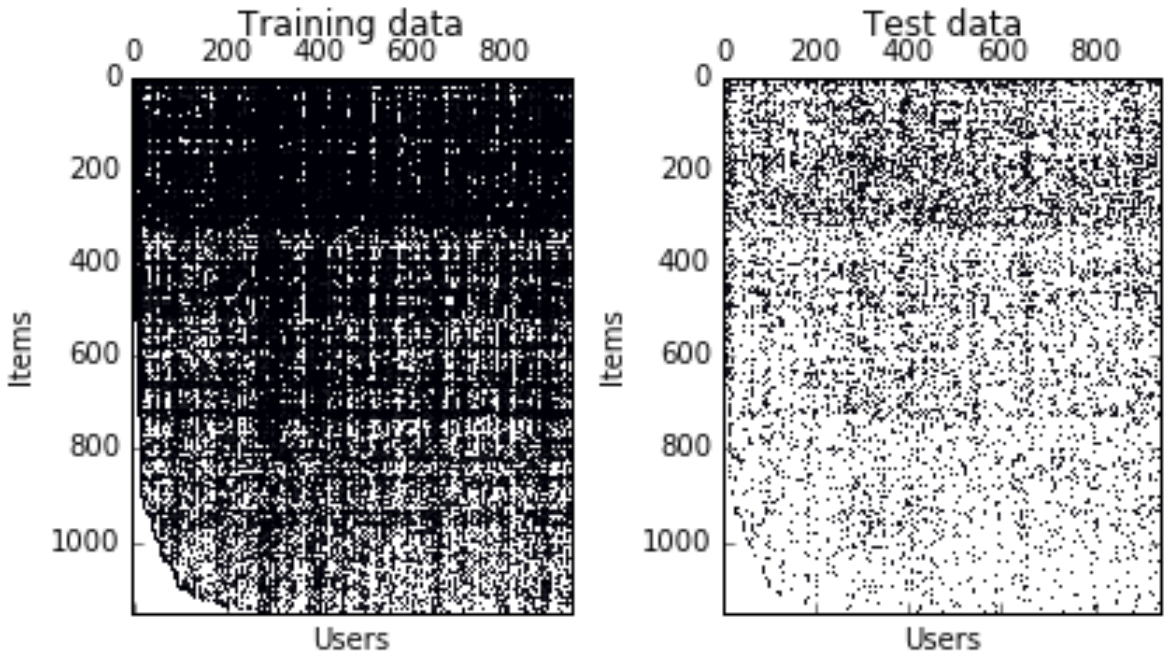

Figure 2: This figure shows obtained train-test split. The left plot shows the training data and the right figure shows the test data. In each plot, a dot indicates a user-movie pair with a non-zero rating.

3 Baseline Models

We will use the following models, that use the mean to predict, as baselines:

$$\text{Global Mean: } \hat{x} := \frac{1}{|\Omega|} \sum_{(d,n)\in\Omega} x_{dn} \quad = \frac{1}{|\Omega|} \sum_{n=1}^{N} \sum_{d\in\Omega_{:n}} x_{dn} = \frac{1}{|\Omega|} \sum_{d=1}^{D} \sum_{n\in\Omega_{d:}} x_{dn}, \tag{2}$$

$$\text{User Mean: } \hat{x}_n := \frac{1}{|\Omega_{:n}|} \sum_{d\in\Omega_{:n}} x_{dn}, \tag{3}$$

$$\text{Movie Mean: } \hat{x}_d := \frac{1}{|\Omega_{d:}|} \sum_{n\in\Omega_{d:}} x_{dn}, \tag{4}$$

where Ω is the set of non-zero rating indices (d, n) in the training data matrix, $\Omega_{:n}$ is the set of movies rated by the n-th user, and $\Omega_{d:}$ is the set of users who have rated the d-th movie.

Exercise 2:

We will compare the above three baselines first.

- Before implementing, think about the following: which of the three models will give the best performance, and why?
- Implement the notebook functions `baseline_global_mean()`, `baseline_user_mean()` and `baseline_item_mean()`.
- Compare the resulting models. Which model gives you the lowest training RMSE? Which one gives lowest test RMSE?
  Hint: You can change the random seed of your train/test split, and thereby generate several estimates.

4 Matrix Factorization with SGD

Exercise 3:

Task: Implement Stochastic Gradient Descent

1. Derive the full gradient $\nabla_{(\boldsymbol{W},\boldsymbol{Z})} f(\boldsymbol{W}, \boldsymbol{Z})$ for $f = \mathrm{RMSE}(\boldsymbol{W}, \boldsymbol{Z})$. Note that since we have $(D + N) \times K$ variables, the gradient will have $(D+N)\times K$ entries (each corresponding to an entry of $\boldsymbol{W}$ or $\boldsymbol{Z}$ respectively).
2. Derive a stochastic gradient $\mathbf{G}$ using the sum structure of f over the Ω elements. We want to do this in such a way that $\mathbf{G}$ only depends on a single observed rating $(d, n) \in \Omega$.
3. Implement Stochastic Gradient Descent as described in the lecture, for our objective function being RMSE as defined in Equation (1).
4. Implement SGD for the regularized cost function

   $$\mathcal{L}(\boldsymbol{W}, \boldsymbol{Z}) := \tfrac{1}{2} \sum_{(d,n)\in\Omega} \left[x_{dn} - (\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}\right]^2 + \frac{\lambda_w}{2}\|\boldsymbol{W}\|_{\text{Frob}}^2 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_{\text{Frob}}^2$$

   where $\lambda_w, \lambda_z > 0$ are scalars.
5. Experimentally find the best stepsize γ to obtain the lowest training error value.
6. Does the test error also decrease monotonically during optimization, or does it increase again after some time? Try for different choices of K.
7. (OPTIONAL: Can you speed up your code, by for example maintaining the set of values $(\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}$ for the few observed values $(d, n) \in \Omega$, and thereby avoiding the computation of the matrix multiplication $\boldsymbol{W}\boldsymbol{Z}^\top$ in every step?)

5 Alternating Least Squares

Exercise 4:

Theory Question: Alternating Least Squares with Missing Entries.

1. Mathematically derive the precise updates for $\boldsymbol{W}$ and $\boldsymbol{Z}$, for the Alternating Least Squares (ALS) algorithm for the more general setting, when only the ratings $(d, n) \in \Omega$ contribute to the cost, i.e.

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{Z}) := \tfrac{1}{2} \sum_{(d,n)\in\Omega} \left[x_{dn} - (\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}\right]^2$$

   Hint: Compute the gradient with respect to each group of variables, and set to zero.

2. Do the same for the regularized cost function

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{Z}) := \tfrac{1}{2} \sum_{(d,n)\in\Omega} \left[x_{dn} - (\boldsymbol{W}\boldsymbol{Z}^\top)_{dn}\right]^2 + \frac{\lambda_w}{2}\|\boldsymbol{W}\|_{\mathsf{Frob}}^2 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_{\mathsf{Frob}}^2$$

   where $\lambda_w, \lambda_z > 0$ are scalars.

Exercise 5:

From our provided template, implement the ALS algorithm for regularized matrix completion, as in the above defined objective function.

- **Step 1** Initialize the item matrix $\boldsymbol{W}$ by assigning the average rating for that movie as the first row, and small random numbers for the remaining entries;
- **Step 2** Fix $\boldsymbol{W}$, Solve for $\boldsymbol{Z}$ by minimizing the objective function (the sum of squared errors);
- **Step 3** Fix $\boldsymbol{Z}$, solve for $\boldsymbol{W}$ by minimizing the objective function similarly;
- **Step 4** Repeat Steps 2 and 3 until a stopping criterion is satisfied.

Try different values of the latent dimension K.