Homework #3 problem 5 revisited - Optional

CSE 546: Machine Learning
Prof. Kevin Jamieson
Due: 12/12 11:59 PM

*This homework assignment is an **optional** replacement for problem 5 on Homework 3. If you choose to complete this assignment and turn it in on gradescope, you will receive the maximum of the score you originally received for problem 5 on homework 3 and the score you receive on this homework assignment. If you do not submit this homework assignment, you will receive the score you received on the original assignemnt. If you received a score of $x \in \{1, \ldots, 8\}$ on the originally assigned problem, and you choose to do this assignment and receive a score of $y \in \{1, \ldots, 8\}$, the score you will be assigned for problem 5 on homework 3 will equal $\max\{x, y\}$. That is, if you already received a score of 8 on the original assignment, you cannot achieve additional points by completing this assignment.*

1 Matrix Completion

5. *[8 points]* You will build a personalized movie recommender system. There are $m = 500$ movies and $n = 1000$ users. As historical data, every user watched a subset of movies and rated them. The goal is to recommend movies the users haven't seen. The historical rating is represented by a matrix $R \in \mathbb{R}^{n \times m}$. The entry $R_{i,j}$ represents the user i's rating on movie j. The rating is a real number in $[-10, 10]$: a higher value represents that the user is more satisfied with the movie. You are provided with two files:

- `train.txt` contains the user-movie-score data representing the training set. Each line takes the form "`i,j,s`", where `i` is the user index, `j` is the movie index, and `s` is the user's score in $[-10, 10]$ describing how much they liked the movie (higher is better).
- `test.txt` has the same format, with the same users rating movies held out from the training set.

Latent factor model is the state-of-the-art method for personalized recommendation. It learns a vector representation $u_i \in \mathbb{R}^d$ for each user and a vector representation $v_j \in \mathbb{R}^d$ for each movie, such that the inner product $\langle u_i, v_j \rangle$ approximates the rating $R_{i,j}$. You will build a simple latent factor model. We will evaluate our learnt vector representations by two metrics

- Mean squared error: $\frac{1}{|S|} \sum_{(i,j) \in S} (\langle u_i, v_j \rangle - R_{ij})^2$ where S (and the corresponding $R_{i,j}$ values) are from the test set
- Mean absolute error: $\frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} |\langle u_i, v_j \rangle - R_{ij}|$ where $\mathcal{N}_i$ are the movies rated by user i in the test set

You will implement multiple estimators and use the inner product $\langle u_i, v_j \rangle$ to predict if user i likes movie j in the test data. You will choose hyperparameters like d or the amount of regularization by creating a validation set from the training set.

a. The first estimator pools all the users together and just predicts what the average user in the training set rated the movie. This is equivalent to $d = 1$ with u as the all ones vector and v minimizing least squares.

b. Now replace all missing values in $R_{i,j}$ no in the training set by zero. Then use singular value decomposition (SVD) to learn a lower dimensional vector representation for users and movies. Recall this means to project the data vectors to lower dimensional subspaces of their corresponding spaces, spanned by singular vectors. Refer to the lecture materials on SVD, PCA and dimensionality reduction. You should use an efficient solver, I recommend `scipy.sparse.linalg.svds`. Try $d = 1, 2, 5, 10, 20, 50$ and plot the error metrics on the train and test as a function of d.

c. For sparse data, replacing all missing values by zero is not a completely satisfying solution. A missing value means that the user has not read the movie, but doesn't mean that the rating should be zero. A more reasonable choice is to minimize the MSE only on rated movies. Let's define a loss function:

$$L\Big(\{u_i\},\{v_j\}\Big) := \sum_{(i,j)\in T} (\langle u_i, v_j\rangle - R_{i,j})^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda \sum_{j=1}^{m} \|v_j\|_2^2,$$

where T and $R_{i,j}$ here are from the training set and $\lambda > 0$ is the regularization coefficient. Implement an algorithm to learn vector representations by minimizing the loss function $L(\{u_i\},\{v_j\})$. Try $d = 1, 2, 5, 10, 20, 50$ and plot the error metrics on the train and test as a function of d. Note that you may need to tune the hyper-parameter λ to optimize the performance.

Hint: you may want to employ an alternating minimization scheme. First, randomly initialize $\{u_i\}$ and $\{v_j\}$. Then minimize the loss function with respective to $\{u_i\}$ by treating $\{v_j\}$ as constant vectors, and minimize the loss function with respect to $\{v_j\}$ by treating $\{u_i\}$ as constant vectors. Iterate these two steps until both $\{u_i\}$ and $\{v_j\}$ converge. Note that when one of $\{u_i\}$ or $\{v_j\}$ is given, minimizing the loss function with respect to the other part has closed-form solutions. You should never be allocating an $m \times n$ matrix for this problem.