Elliot Greenlee
egreenle@vols.utk.edu

Assignment 2

1 Collaborators

Discussed cross validation functions in Spark with Razieh Baghbaderani.

2 Approach

2.1 Background

This report covers the design and implementation of a recommendation system for personalized music. After preprocessing the data, non-negative matrix factorization using alternating least squares is used to predict a user's implicit rating of a given artist. The performance of these personalized music recommendations is evaluated for various hyperparameter settings, with the goal of minimizing mean squared error.

Recommendation algorithms are ubiquitous among media and shopping platforms, providing personalized suggestions for movies, shows, songs, and products. This seemingly inexplicable process for a human: observing action and extracting preference, is performed well by computer systems. However, more advanced algorithms are challenging to build, causing companies like Netflix, Amazon, Spotify, Pandora, and Hulu to develop secret in house solutions.

Depending on the platform and situation, these solutions can be either explicit or implicit. Explicit systems rely on ratings chosen by the user, such as a thumbs up or down, or four out of five stars. Unfortunately, users rate much less than they use the system, so instead algorithms extract implied relationships in user decisions. This provides a much large implicit dataset with richer information. In the case of music recommendation, this can mean how often a song is played, with user-artist relationships observed and predicted.

2.2 Data

The original dataset for this project is from 2005 and provided by Last.fm at http://bit.ly/1KiJdOR. However, that large source of 140,000 users, 1.5 million artists, and 24 million entries is deprecated. Another source comes from our class, but due to issues with the server environment, the data size had to be reduced. The smaller dataset has 50 users, 30,537 artists, and 49,481 entries. Unfortunately, these sizes lead to poor predictive performance.

The dataset consists of three files. One contains a list of the useful data, which matches a player ID with an artist ID and the number of times the user has played a song by that artist. Additionally, there is a file that matches the artist ID to the real name of the artist. Finally, there is a file that matches the IDs of nonstandard or misspelled names of artists to their canonical name IDs. In order to parse the data into PySpark, extra spaces were removed from the documents, and the values were converted into csv format.

2.3 Methods

Luckily, all of the user IDs are positive and below 2147483647, the max value for an integer, which is required for the non-negative matrix factorization (NMF) by the alternating least squares (ALS) method to be used for recommendation, so no processing is needed. First, the artist ID column is cleaned by matching any alias IDs that appear to a canonical ID. Next, a column of human readable names is generated based on the appropriate ID. Finally, because the number of artist plays for some users is an improbable number, values are clipped to 4800, which is 10 entire days of listening to a given artist, assuming an average song length of 3 minutes. At this point, the data is ready for prediction.

Non-negative matrix factorization provides characterizations of items and users by vectors of latent factors. For items, the vector q_i gives whether certain factors are possessed, such as comedy or romance. For users, the vector p_u gives whether certain factors are desired. There is a high correspondence between the latent factors of users and items, but the exact latent factors are unknown. The dot product $q_i^T * p_u = r_{ui}$ characterizes the overall interest. In order to optimize the vectors, methods minimized the regularized squared error on the set of known ratings across all pairs of factors

$$min(\sum_{u,i} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2))$$

Explicit datasets are sparse which is acceptable for stochastic gradient descent as the runtime scales linearly with increases in the data; however, implicit datasets are generally dense so another technique is needed. Alternating Least Squares (ALS) is more optimized and is fully parallelizable. If one of q_i and p_u is fixed, the problem becomes quadratic and be solved optimally. ALS switches between a fixed q_i and p_u to solved the least-squares problem, decreasing to convergence [1]. As for many prediction algorithms, unexpected or unseen data is challenging for ALS to handle. In the case of this project, new users or items that have no rating history on which the model has been trained can use a cold start strategy to drop rows that contain NaN values.

In order to test the performance of the recommender, the data must be divided into a training and testing set. The training set is used to create the model, and the testing set is used to evaluate the learned performance. Unfortunately, many models suffer because the training sample is not infinite, so the distributions are not accurately estimated. This leads to problems with overfitting, where the model works well on the training data but poorly on the testing data. Cross validation attempts a solution to this problem by using the data to create multiple, randomly inter-selected sets of training and testing data. With m-fold cross validation, the data is broken into m samples of equal size for testing, with the rest of the data not selected for testing being used for training. This method provides more realistic accuracy numbers for the data, and lessens the effects of overfitting.

The ALS method has a few hyperparameters in the algorithm which can be varied depending on the data to improve performance. In the case of this project, two grid searches, one rough and one fine, are run across ranges for all the hyperparameters to find the best combination. The metric used is mean squared error (MSE), which provides a measure of how concentrated the data is around the model. Better performance is

indicated by a lower value, with zero being the best possible performance. Formally,

$$MSE = \sum_i (f_i - y_i)^2$$

where y_i are the real values and f_i are the predicted values.

3 Results

The first verification of the results is a spot check of three users: 2064012, 1000647, and 2023686. Rough visual comparison between the predictions and real implicit ratings seems to confirm that the algorithm is working. Many of the predictions are very close, and almost all are the same order of magnitude. The top twenty results for each user can be obtained from the accompanying notebook file.

For this project, 10 fold cross validation was used to evaluate MSE across various hyperparameters. The results of varying each hyperparamter are described below. Two levels of grid search were used, one rough and one smooth. The results of these grid searches, including the best result, are described below.

There are four hyperparameters that control the model. Rank is the number of latent factors in the model, and the number of columns in the user and artist feature matrices. Iterations is the number of iterations the factorization runs, alternating between users and items. Lambda is the overfitting parameter. Alpha controls the relative weights of the observed versus the unobserved user-artist interactions in the factorization.

Iterations was never varied, and no change was observed when varying alpha, most likely due to the cold start strategy of dropping unseen users or artists. However, when the rank is varied from 10 to 70 while holding iterations at 10, the regularization parameter at 1.0, and alpha at 1.0, there is an error minimum around rank 50. Additionally, it appears that as the regularization parameter is increased, the error decreases to a plateau. This can be seen in tables 1 and 2.

Rank	10	20	30	40	50	60	70
MSE	339k	356k	318k	294k	265k	275k	284k

Table 1: The mean squared error averaged over 10 folds of cross validation when varying the ALS rank.

Rank	3	1	$1e^{-1}$	$1e^{-2}$	$3e^{-3}$	$1e^{-3}$	$3e^{-4}$	$1e^{-4}$	$1e^{-5}$
MSE	247k	264k	472k	331k	302k	293k	304k	321k	341k

Table 2: The mean squared error averaged over 10 folds of cross validation when varying the ALS rank.

Two searches were run across the four parameters. The results of the first search revealed that a rank of 50 and a regularization parameter higher than 1 achieved the best results. The second, finer grain search is shown in table 3. The best result obtained was an MSE of 238k when using a rank of 52, a regularization parameter of 100, 10 iterations, and an alpha of 1.0.

rank/regParam	46	48	50	52	54
100	239k	239k	239k	238k	239k
30	242k	241k	242k	241k	241k
10	242k	242k	242k	242k	243k
3	245k	245k	247k	246k	244k
1	263k	279k	265k	275k	269k

Table 3: The mean squared error averaged over 10 folds of cross validation when varying ALS rank (columns) and regularization parameter (rows).

4 Code

The code for this project is written in a Jupyter notebook using Python3 and PySpark. Findspark is used to integrate Spark, and will need to be altered or removed to run on another system. A pip requirements file is provided as requirements.txt.

References

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.