Assignment 5: Collaborative Filtering

Arash Vahdat

Fall 2015

Readings You are highly recommended to check the following readings before/while doing this assignment:

- Slope One Algorithm: `https://en.wikipedia.org/wiki/Slope_One`.
- Alternating Least Squares (ALS) Method: Chapter 3, "Advanced Analytics with Spark", Ryza, Laserson, Owen and Wilss, O'Reilly Media, 2015.
- "A Comparative Study of Collaborative Filtering Algorithms", Joonseok Lee, Mingxuan Sun, and Guy Lebanon, arXiv preprint 1205.3193 (2012).
- "A Survey of Collaborative Filtering Techniques", Xiaoyuan Su and Taghi M. Khoshgoftaar, Advances in artificial intelligence, Vol 2009.
- "Improving regularized singular value decomposition for collaborative filtering", Arkadiusz Paterek, Proceedings of KDD cup and workshop. Vol. 2007. 2007.

Provided Code and Data In this assignment, we will use the MovieLens dataset which contains 100K movie ratings available at:

`/cs/vml2/avahdat/CMPT733_Data_Sets/Assignment5`

1 Introduction

Assume you are running an online retail store. Being able to predict your users interests can significantly boost your sales by providing personalized suggestions based on their profiles. Recommendation systems are designed to address this problem.

Generally, existing algorithms for recommendation systems can be divided into two main categories: Content-based filtering algorithms that use products' or users' characteristics to recommend products to users. And Collaborative filtering algorithms that use user's past behavior, e.g. items bought, rated, or visited. Collaborative filtering algorithms can be further sub-categorized in two groups based on the data they rely on. The first group consists of collaborative filtering systems based on *implicit feedback* data. This type of data relies on the assumption that users only visit or brows through products of their interest. The second group refers to systems using *explicit feedback* data. In this type of data, users provide their ratings explicitly for known products.

In this assignment, we will examine two collaborative filtering algorithms using explicit feedback data. First, we will try the Alternating Least Squares method which is a parametric matrix decomposition-based

algorithm. Second, we will implement the Slope One algorithm which is a simple non-parametric model that uses item-to-item comparisons to infer a user's rating for a given item.

In order to train and test your models, you are provided with train and test splits extracted from the 100K movie ratings of the MovieLens dataset. These splits can be found in the directory mentioned above (`MovieLens100K_train.txt, MovieLens100K_test.txt`). The dataset is organized as follows: `"UserID \t MovieID \t Rating \t Timestamp"` where UserIDs range between 1 and 943, MovieIDs range between 1 and 1682, and, Ratings are based on a 5-star scale (whole-star ratings only). You can ignore the timestamp data for this assignment.

2 Alternating Least Squares (ALS) Method

Assume you have access to a movie descriptor which represents a movie using semantically meaningful measurements. For example, your feature vector could have multiple dimensions. Some feature dimensions could represent the movie genre (e.g. if it is an action movie or not, if it is a comedy movie or not, and so on). You could also include other features such as the production date, the cast, the list of directors etc.

Given a movie representation (q_j), a collaborative filtering algorithm can be modeled as a regression problem where the goal is to find a weight vector for each user (p_i) which maps the movie features to the user's ratings (r_{ij}). In our feature vector example, the weight vector represents how much the user likes different movie genres, actors or directors. During training the model learns a preference weight vector for each user such that the predicted ratings ($p_i^T q_j$) are as close as possible to the user's provided ratings (r_{ij}).

The Alternating Least Squares (ALS) method assumes that there exist a hidden representation for each movie. It represents all user ratings as a linear combination of movie features where the weights of the linear combination is user-specific. The goal of the ALS algorithm is to find both movie features and user preferences alternatively by solving one parameter at a time (remaining parameters being fixed).

Question 1: (5 marks) ALS algorithm can be easily used in Spark. Follow the example in the Spark documentation[1] to train a recommendation model using the provided training data.

Two main hyperparameters are used to tune the ALS model. The first parameter is a "rank" variable which controls the size of a movie representation (item factor vector). The second parameter is a variable "lambda_" (λ), which balances the model parameter regularization. Try different values, exponents of two $(2, 4, 8, 32, ..., 256)$ for tuning the hyperparameter "rank" and different exponents of 10 $(0.01, 0.1)$ for tunning "lambda". Fix the number of iterations to 20 for each of your experiments. Measure your performance for each configuration using the Root Mean Squared Error (RMSE) criterion.

Visualize the error versus ranks on the test data while fixing lambda to the best value for each rank. Submit your plot along with the implemented code used to generate your results. What is the lowest test error achieved, and what are the *rank* and *lambda_* values for which you have achieved the reported lowest error?

[1] `http://spark.apache.org/docs/1.3.1/mllib-collaborative-filtering.html`

How does the error change as the rank increases? How can you explain the irregular patterns observed when visualizing the error vs rank?

Question 2: (15 marks) Training a recommendation system using the ALS method, will result in two big matrices, more specifically, two RDDs in Spark. The first matrix will contain a factor vector for each movie, and the second matrix will contain user preferences or the weight vector.

Paterek, in the paper "improving regularized singular value decomposition for collaborative filtering" proposed a very interesting idea. In the Sec.3.6 of the paper, the author proposes to feed the hidden movie features trained using ALS-type algorithm to a non-linear regression model (in their case, kernel ridge regression). Due to the non-linearity of this regression model, there is a chance to train a better recommendation model.

In this question, we are going to reproduce this idea using the following steps:

1. Train an ALS model using fixed hyperparameters selected from the previous question.
2. Extract movie features using `model.productFeatures()`. This command will return an RDD containing <movieID, feature> pairs.
3. Collect all features for movies rated by each user in the training data. While considering the corresponding user ratings as target values for each movie feature, train a kernel ridge regression model for each user. You can use the kernel ridge regression class from the Sklearn package [2]. Try different kernel types (RBF, Polynomial, Sigmoid, etc) as your kernel, and cross validate the kernel parameters as well as the regularizer coefficient for the regression model.
4. For test, apply the corresponding user-based regression model to the movie feature collected for each user-movie pair .
5. Evaluate your performance using the RMSE criterion.

Implement this idea and compare your performance to the best model trained in the previous question. Can you improve your performance using the above-mentioned trick? What kind of kernel did you use for this question and which values did you set for the kernel parameters?

Note: To perform steps 3 and 4, you will need to combine multiple RDDs[3]. For this, use Spark's RDD operations. Your answer to this question should include **how you used RDD operations to implement steps 3 and 4.**

3 Slope One

Slope One is a family of non-parametric algorithms proposed for collaborative filtering. It was originally proposed by Daniel Lemire and Anna Maclachlan[4]. Slope One is a fairly simple yet powerful algorithm. Learn about this algorithm by following the instructions seen during the lecture or in the following wiki page

[2] `http://scikit-learn.org/stable/modules/kernel_ridge.html`

[3] Familiarize yourself with the main RDD operations: **map, reduce, groupByKey, reduceByKey, join, flatMap, mapValues**.

[4] Daniel Lemire, Anna Maclachlan, Slope One Predictors for Online Rating-Based Collaborative Filtering, In SIAM Data Mining (SDM'05), Newport Beach, California, April 21?23, 2005. available at: http://arxiv.org/pdf/cs/0702144v2.pdf

https://en.wikipedia.org/wiki/Slope_One. Lemire and Maclachlan's original paper is also a good source to familiarize yourself with the concepts introduced in this algorithm.

In this section, we are going to implement the weighted Slope One algorithm using the RDD operations available in Spark. For implementing this algorithm we will need to compute the average deviation of the j^{th} movie with respect to the i^{th} movie using the following equation:

$$dev_{j,i} = \frac{1}{c_{j,i}} \sum_{u \in U_{j,i}} r_{u,j} - r_{u,i} \tag{1}$$

where $U_{j,i}$ represents the set of all users rating both movies, $c_{j,i}$ is the cardinality of this set ($c_{j,i} = |U_{j,i}|$) and $r_{u,j}$ and $r_{u,i}$ are the ratings provided by the user u for the j^{th} and i^{th} movie, respectively, in the training data.

At test time, we can predict the rating of the user a for the movie j using:

$$P_{a,j} = \frac{\sum_{i \in N} (dev_{j,i} + r_{a,i}) c_{j,i}}{\sum_{i \in N} c_{j,i}} \tag{2}$$

where N represents the set of all movies rated by user u.

Question 3: (5 marks) To solve Eq.2, we need to compute both $dev_{j,i}$ and $c_{j,i}$ for any pair of movies with a common rating. Explain how to compute these two quantities using the RDD operations available in Spark.

Question 4: (5 marks) Briefly explain how to predict users movie ratings in the test dataset using Eq.2, given an RDD containing both $dev_{j,i}$ and $c_{j,i}$ for each movie pair and another RDD containing the training user-movie pairs.

Question 5: (10 marks) Implement the Slope One algorithm. Use your analysis from Q.3 to compute $dev_{j,i}$ and $c_{j,i}$ on the training data, then, use your analysis from Q.4 to predict ratings for every user-movie pair in the test dataset. Report your performance on the test dataset using the root mean squared error. Does your Slope One implementation outperforms the ALS method?

Question 6: (10 marks) The above recommendation system algorithms can be improved in multiple ways. The following is a short list of candidate approaches:

1. Model fusion: this consists in the aggregation of different models. In our case, we could simply combine the models learned with Slope One and ALS algorithm by taking a weighted average of their predictions for the test data.
2. Hybrid approach: In Q.2 you were asked to extract hidden feature vectors for movies. Some datasets include semantic descriptors that represent genre or casts for a given movie. The hidden movie features can be concatenated with the semantic features for training the non-linear regression model in Q.2. In our dataset, we have access to movie genres (provided to you in the data directory). You could try to concatenate these two feature types before training your kernel ridge regression.

Implement a subset of these approaches. Design experiments to validate your new model on the test dataset. Report how your performance changes. You are highly encouraged to propose approaches other than the ones listed above and provide your own analysis on what you think might boost your recommendation system performance.

4 Submission

Your assignment should be submitted online at `https://courses.cs.sfu.ca/`. You should submit two files:

1. **Report:** Create your report in PDF format including your answers for each question.
2. **Code:** Create an archive file containing all scripts implemented in this assignment. You should design your scripts such that your results can be easily reproduced. Please include a readme.txt file in your archive briefly explaining which script was used for each question.