CS6350: BIG DATA ANALYTICS and MANAGEMENT

Spring 2016

HW #3

Related to: Data Analytics and Recommendation System using Spark

Due: April 8, 2016 by 11.55 p.m.

This homework consists of two parts. Here, we focus on K-means clustering (data analytics), classification and recommendation systems.

Part A:

Q1

Using spark machine learning library spark-mlib, **use kmeans to cluster the movies using the ratings given by the user,** that is, use the item-user matrix from **itemusermat File provided** as input to your program.

Dataset description.

Dataset: Itemusermat File.

The **itemusermat file contains** the ratings given to each movie by the users in **Matrix format.** The file contains the ratings by users for 1000 movies.

Each line contains the movies id and the list of ratings given by the users.

A rating of 0 is used for entries where the user did not rate a movie.

From the sample below, user1 did not rate movie 2, so we use a rating of 0.

A sample **Itemusermat file** with the item-user matrix is shown below.

	user1	user2
movie1	4	3

movies2	0	2

Set the number of clusters (**k**) to 10

Your Scala/python code should produce the following output:

- For each cluster, **print any 5 movies in the cluster. Your output should contain the movie_id, movie title, genre and the corresponding cluster** it belongs to. **Note:** Use the **movies.dat** file to obtain the movie title and genre.

  For example
  cluster: 1
  123,Star wars, sci-fi

Q2 Classification

Using spark MLlib, use the supervised learning (decision tree and Naive Bayes) algorithms to classify types of glass based on the dataset "glass.data"

The dataset comprises of the following attributes.

Attribute Information:

1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as

   are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium

8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)

    -- 1 building_windows_float_processed

    -- 2 building_windows_non_float_processed

    -- 3 vehicle_windows_float_processed

    -- 4 vehicle_windows_non_float_processed (none in this database)

    -- 5 containers

    -- 6 tableware

    -- 7 headlamps

Please use 60% of the data for training and 40% for testing and give the accuracy of the classifiers.

Part B Recommendation System [Latent Factor Model]

Q3a.

Recommendation system using Matrix factorization with Alternating Least Squares (ALS).

Description of the approach can be found at:

Source: Stanford Distributed Algorithms and Optimization class.

http://stanford.edu/~rezab/dao.

Instructor: Dr. Reza Zadeh, Databricks and Stanford.

Here is the relevant part:

therefore be to fix Y and optimize X, then fix X and optimize Y, and repeat until convergence. This approach is known as **ALS(Alternating Least Squares)**. For our objective function, the alternating least squares algorithm is as follows:

Given below is the ALS algorithm.

14 Matrix Completion via Alternating Least Square(ALS)

14.1 Introduction

A common problem faced by internet companies is that of recommending new products to users in personalized settings (e.g. Amazon's product recommender system, and Netflix movie recommendations). This can be formulated as a learning problem in which we are given the ratings that users have given certain items and are tasked with predicting their ratings for the rest of the items. Formally, if there are n users and m items, we are given an $n \times m$ matrix R in which the $(u, i)^{th}$ entry is r_{ui} – the rating for item i by user u. Matrix R has many missing entries indicating unobserved ratings, and our task is to estimate these unobserved ratings.

14.2 Matrix Factorization: Objective and ALS Algorithm on a Single Machine

A popular approach for this is **matrix factorization**, where we fix a relatively small number k (e.g. $k \approx 10$), and summarize each user u with a k dimensional vector x_u, and each item i with a k dimensional vector y_i. These vectors are referred to as *factors*. Then, to predict user u's rating for item i, we simply predict $r_{ui} \approx x_u^\intercal y_i$. This can be put in matrix form: Let $x_1, \dots, x_n \in \mathbb{R}^k$ be the factors for the users, and $y_1, \dots, y_m \in \mathbb{R}^k$ the factors for the items. The $k \times n$ user matrix X, and the $k \times m$ item matrix Y are then defined by:

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix}, Y = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_m \\ | & & | \end{bmatrix}$$

Our goal is then to estimate the complete ratings matrix $R \approx X^T Y$. We can formulate this problem as an optimization problem in which we aim to minimize an objective function and find optimal X and Y. In particular, we aim to minimize the least squares error of the observed ratings (and regularize):

$$\min_{X,Y} \sum_{r_{ui}\, observed} (r_{ui} - x_u^\intercal y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right) \tag{1}$$

Notice that this objective is non-convex (because of the $x_u^T y_i$ term); in fact it's NP-hard to optimize. Gradient descent can be used as an approximate approach here, however it turns out to be slow and costs lots of iterations. Note however, that if we fix the set of variables X and treat them as constants, then the objective is a convex function of Y and vice versa. Our approach will

Algorithm 1 ALS for Matrix Completion

```
Initialize $X, Y$
repeat
  for $u = 1 \dots n$ do
      $$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i \qquad (2)$$
  end for
  for $i = 1 \dots m$ do
      $$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u \qquad (3)$$
  end for
until convergence
```

Equation 2 updates the latent vectors for each user. Equation 3 updates the latent factors for each item.

You will be required to fill in the code that implements Equation 2 and 3 in the given als.scala file to update the latent vectors for each users and movies/item.

You will be using ratings.dat as input to your program.

Given below is the Dataset description:

ratings.dat

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040

- MovieIDs range between 1 and 3952

- Ratings are made on a 5-star scale (whole-star ratings only)

- Timestamp is represented in seconds since the epoch as returned by time(2)

- Each user has at least 20 ratings

movies.dat

MovieID::Title::Genres.

Output:

1. Print out the learned latent vector for user id 1,1757 and 1759.

2. Print out the learned latent vectors for movies 914, 1777 and 231.

Note: Your code should be able to retrieve any of the latent vectors by taking the user_id or movie_id from command line argument.

Q3b

Using the factorized matrix obtained from Q3a, Provide the predicted rating for the given users and movies below.

user 1 and movieid 914,

user 1757 and movieid 1777,

user 1759 and movieid 231.

Submission:

Please upload your submission via e-learning. Please upload the following to eLearning:

1. Submit the answer to Q1 as Q1.txt
2. Please submit the source code and the accuracy of the classifiers.

3. Submit the answer to Q3 as completed als.scala file and Q3b answer. If you use python/scala, then submit all source files.