Movie Recommendations from User Ratings

Hans Byström
Stanford University
bystrom@stanford.edu

ABSTRACT
This paper demonstrates an implementation of k-means clustering and softmax regression classification for movie recommendation based on the hetrec2011-movielens-2k dataset. Root-Means-Square Error (RMSE) is used for evaluation of results. Reasonable performance is demonstrated.

1. INTRODUCTION
A recommendation system is an important component in many online services. The recommendation system can help the user find interesting content or goods to consume which increases user satisfaction, and for the service provider this leads to the value of increased usage of the service.

Movie recommendation systems have attracted significant interest in recent years. One example of this was the Netflix Prize [1], where contestants were rated based on their algorithm's Root-Mean-Square-Error (RMSE) score. The Root-Mean-Square Error is a measure of the algorithm's error in the predictions of users' ratings of movies.

This paper is exploring the approach of using k-means to cluster and label users, and to use a softmax regression classifier to predict to which cluster the end-user belong, based on user ratings of movies. The recommendation system will then suggest the highest rated movies from that cluster. Prediction accuracy of the classifier is calculated and compared with the Root-Mean-Square Error of the assigned ratings compared to ratings in a test set.

2. METHODS
K-means clustering algorithm [2] is used to group users in to clusters based on their ratings. This is the k-means clustering algorithm:

1. Initialize centroids $\mu_1, \dots, \mu_k \in \mathbb{R}^n$ randomly.

2. Repeat until convergence : {

For every *i*, set

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

For each *j*, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$$

}

Users in the complete dataset were labeled based on cluster assignment.

For classification, softmax regression [3] was used, with hypothesis function

$$h_\theta\left(x^{(i)}\right) = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

The dataset was divided into training set and test set, and the model parameters θ were trained on the training set to minimize cost function J(θ):

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)}=j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n} \theta_{ij}^2$$

Prediction accuracy of the classification of users to clusters was measured as

$$accuracy = \frac{number\ of\ correct\ classifications}{total\ number\ of\ classifications}$$

Root-Mean-Square Error (RMSE) was used for evaluation and is defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|S_{test}|}\sum_{u,i \in S_{test}} (\hat{r}_{ui} - \mathrm{r}_{ui})^2}$$

Here, S_{test} is the set of all user ratings in the test set, $\hat{r}_{ui}$ is the by system predicted rating of item (movie) *i* by user *u*, and r_{ui} is the ratings in the test set. In the approach used in this paper, the predicted ratings $\hat{r}_{ui}$ are the ratings represented by the centroid that was assigned in the classification stage, and these are compared with ratings in the test set, only for those movies that each user has rated.

For comparison, a baseline prediction [4] was calculated as

$$b_{ui} = \mu + b_u + b_i$$

where μ is the average rating in the dataset, b_u is the bias in user *u* ratings, and b_i is the offset in average of ratings for movie *i*.

3. DATASET
The hetrec2011-movielens-2k dataset was used [5], which is a subset and extension of MovieLens10M dataset [6]. The dataset contains ratings of 10109 movies by 2113 users. Movies are rated with score 1-5.

4. IMPLEMENTATION
4.1 Dataset preprocessing
Dataset was loaded using python pandas into a matrix represented as numpy array of size *users* x *movies* (2113 x 10109). Element a_{ij} represents the rating of movie *j* by user *i.* If this user did not rate this specific movie the element is set to 0.

Ratings from each user was centered around zero by removing mean: element $a_{ij} = a_{ij} - \overline{a_i}$, where $\overline{a_i}$ is the mean of all ratings by user *i* (same as bias b_u in the baseline predictor).

The dataset was divided 70/30 into training set and test set. The training set consisted of ratings from 1480 users and the test consisted of ratings from 633 users.

4.2 Algorithms

K-means was implemented in python using numpy.

Softmax regression implementation was done with numpy and scipy following the Ufldl tutorial [3]. The ground truth matrix was represented as a scipy.sparse matrix. The scipy.optimize L-BFGS-B solver implementation was used to solve for the minimum of the cost function J(θ).

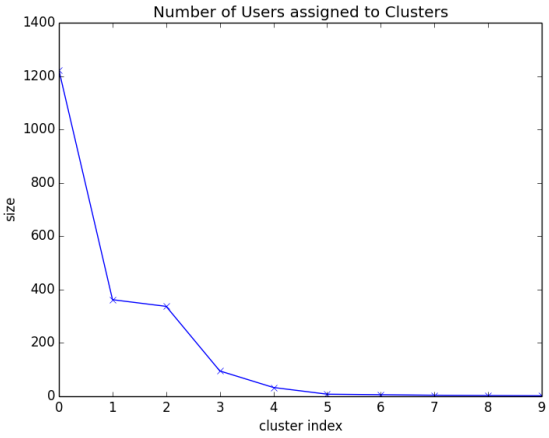
4.3 Processing

Running the k-means clustering algorithm with k=100 took 554 seconds on a MacBook Air 1.8 GHz Intel Core i5 with 4GB RAM.

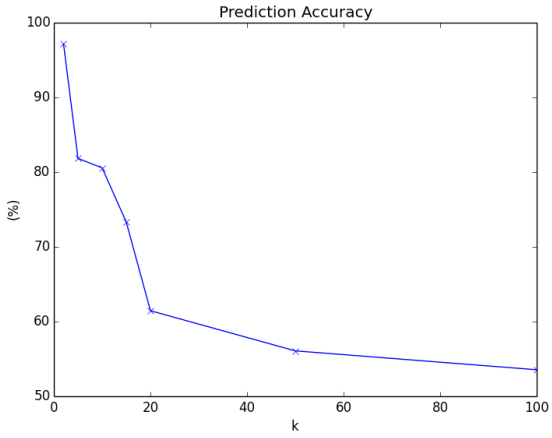
Training the softmax classifier with k =100 on the training set took 196 seconds. Classifying 633 users took 0.59 seconds (k = 100).

5. RESULTS

After running k-means, the plot of the number of assigned users to each cluster showed a power-law curve for all values of $k \geq 5$, where the majority of users were assigned to first cluster and then a bump on the curve with 2-3 equally sized clusters, and then a long tail with small clusters. The plot here below shows the number of users assigned to each cluster for k = 10.

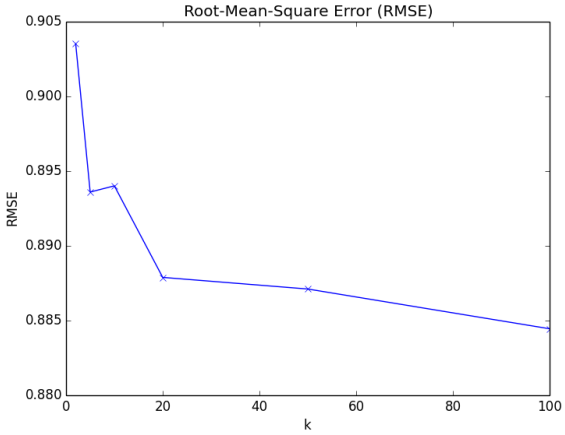


The accuracy in prediction of labels by the softmax regression classifier showed a drastic fall in the accuracy of predicted labels with an increased k.



The baseline predictor showed an RMSE = 0.900.

The best achieved result with the approach described in this paper was RMSE = 0.884, which is an improvement of 1.81% compared with baseline.



The low accuracy of the prediction of which cluster a user belongs to, while still maintaining a reasonable RMSE, shows that several cluster centroids are close to each other. Each centroid can be considered representing a certain preference or taste, and for this dataset, users' tastes does not seem to differ that much for some groups of the users.

6. DISCUSSION

Implementation and evaluation of k-means clustering and softmax classification for movie recommendation was made based on the hetrec2011-movielens-2k dataset.

The softmax predictor showed good processing performance also for large number of classes (k = 100). A good prediction processing performance is a requirement for implementation in a commercial Movie Recommendation service.

Comparing the result of the accuracy in prediction of user ratings, measured as the RMSE improvement relative to baseline, with results from Bao and Xia [7] who tested a number of different methods on a variant of the same dataset, the approach in this paper gives better result (1.81%) than K-Nearest Neighbor (1.15%), but performs less well than SVD and its variants (4.82%).

As further work, a comparison test with SVD, and in particular Asymetric SVD, could be made on this specific dataset to get a firmer comparison between the methods.

7. REFERENCES

[1] Netflix, "Netflix Prize Webpage", 2009, http://www.netflixprize.com/

[2] A. Ng, "CS229 Lecture Notes", 2013, Stanford University.

[3] Ng, et al, "UFLDL Tutorial", 2013, Stanford University, http://ufldl.stanford.edu/wiki/index.php

[4] Ekstrand, et al, "Collaborative Filtering Recommender Systems", 2010, Foundations and Trends in Human–Computer Interaction.Vol. 4, No. 2, p 81–173

[5] http://www.grouplens.org/datasets/hetrec-2011/

[6] http://www.grouplens.org/datasets/movielens/

[7] Z. Bao, H. Xia, "Movie Rating Estimation and Recommendation", CS229 2012 project, Stanford University.