An Elastic Group Recommendation System Designed for Multivariate Dynamic Attributes

Naveen Mysore
EMC Corporations
Santa Clara, CA
Email: navimn1991@gmail.com

***Abstract*—In this paper an introduction to an elastic group recommendation system is made where recommendation happens by collaborative and content filtering at three phases for multivariate dynamic attributes. Most regular recommendation systems work on static data contents where as an elastic group recommendation is designed to work for dynamic data inputs and on different use cases. recommendation systems which are commercially available today focuses on individual behavior however for events like movies and restaurants we need a group recommendation systems where group requirements are prioritized. The system introduced here is a breed model of content based and collaborative filtering techniques and filtering is done at three phases so that we can make a recommendation which works best for an individual as well as the group. The system model designed here can have wide applications in making recommendations for restaurants, movies or events for a group of people in real time for varying attributes like users tastes and preferences. On the other hand the system tries to address a classic human psychology problem called $Abileneparadox$ [2].**

Keywords*—Group recommendation systems; Data Mining; Knowledge Discovery; Content based filtering; Collaborative filtering; Hybrid models; Learning approaches; Machine Learning.

I. Introduction

Today many applications are backed by intelligent recommendation systems. Netflix does movie recommendations where as Amazon does product recommendation. Music recommendation systems like Pandora uses very complex algorithms to predict that next song that you might like. In this paper when we say a product it means a use cases like a movie, restaurant, weekend idea etc. The domains in which recommendation systems can be applied is varying day by day from online dating to financial analysis. This shows that the research in developing intelligent recommendation systems is gaining popularity these days. But one thing we can notice in majority of these recommendation systems is they make predictions based on static data contents. ie. they all have a database which hold user profiles and product features and run a prediction algorithm to fetch those top ranked list for a given user. However for applications which involve dynamic user activity like say user preference for dinner tonight or music matching your mood etc are very random and its hard to predict. This means we need to make our recommendation systems very elastic and when I say elastic it means a generic recommendation system which can work as plug and play and can also handle dynamic data input. This recommendation systems is designed to handle multiple products at a same time for varying attributes.

II. Related works in Group recommendation system

Many researchers have come up with different models of group recommendation system but there applications in real commercial products are still limited. To get a quick intro into group recommendation systems we can refer PolyLens [9]. PolyLens is a collaborative recommendation system making recommendation for a group of people rather than an individual. We can see that they use static data contents for making predictions. Even today most recommendation systems have focused on making recommendation for an individual and it mostly depends on the applications. In PolyLens they have explored the architecture on how the system would work.In another work a TV news recommendation system is designed to work for group of people[5]. The approach taken gives an intro into elastic nature of its systems but still lacks the usage of real time dynamic input. The system needs a common user taste profile to work which means you as an individual who have many groups need to make many group profiles to make this work and this shows as a user we can feel that system needs to more elastic. In another work a user modeling strategy is introduced where the system averages the preferences of the individual and the preferences of the remaining members of the group and the actual recommendation of for individual cases are performed using the incremental critiquing method[**?**]. This incremental critiquing method looks interesting as it considers that the fact that the user preferences are subjected to change. Systems like Pandora recommendation uses a very high dimensional feature space to make predictions[10].There are many works related to this domain varying to different applications and one thing we can notice is majority of these systems use hybrid versions of collaborative filtering systems. The work presented in the paper is also one such hybrid version of collaborative and content filtering systems.

III. Problem space

Elastic group recommendation systems can be applied to wide variety of problems and perhaps one of the most common encountered problem is a group event planning. Let us consider a hypothetical situation where three friends decide to hangout. Now for the host if he/she wants to pick a restaurant then he/she has to consider various parameters like location, their availability, their choices, Mode of transportation etc and in a group discussion like this everyone has opinions and preferences and host holds the responsibility for entire event. In these situation the classical Abilene paradox says that a group of people collectively decide on a course of action that

978-1-5090-0806-3/16/$31.00 copyright 2016 IEEE
ICIS 2016, June 26-29, 2016, Okayama, Japan

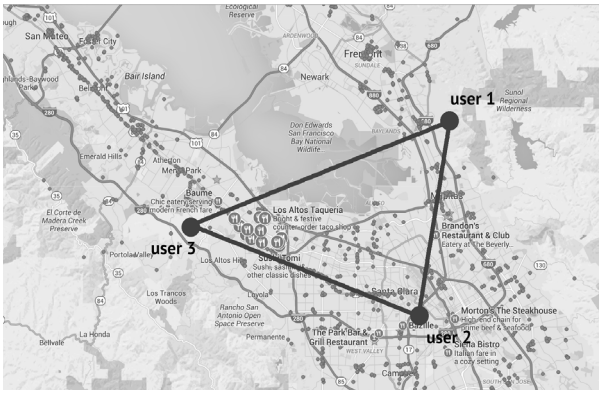

Fig. 1. A Group recommendation scenario.

TABLE I. A SAMPLE FEATURE AND PREFERENCES REPRESENTATION

	$\theta^{(1)}$	$\theta^{(2)}$	$\theta^{(3)}$	x_1	x_2	x_2
$\chi^{(1)}$	0.80	0.09	0.27	0.93	0.01	0.06
$\chi^{(2)}$	-	-	0.09	0.25	0.50	0.25
$\chi^{(3)}$	0.95	0.51	0.03	0.69	0.68	0.32
$\chi^{(4)}$	-	0.87	0.5 0	0.83	0.08	0.32

is counter to the preferences of many of the individuals in the group and this doesn't happen always but the chances are high that choice made might not please many in the group. We want our system to make recommendation such that it pleases everyone. The system should also apply learned features from different vector spaces to make good recommendation. In Figure 1 we can see that there are 3 users at different locations making a polygon (Triangle). There are multiple clusters of data point and we need to pick the best ones from them and make recommendation which works best for both group and individual members.

IV. USER PROFILES AND ITEM FEATURE VECTORS

As we know in a recommendation system we have a user preference profiles and item feature vectors. In content based filtering we use feature vectors to learn the user preferences and in collaborative filtering technique we use the user profiles to learn product/item features. before we jump into the feature and profile extraction these are some conventions used in this paper.
n_u : number of users
n_m : number of items
n_f : number of features we have
n_p : number of preferences
χ : feature vectors where $\chi^i \in R^{n_f}$
θ : user profile vector where $\theta^j \in R^{n_p}$
$G\theta$: group profile vector where $\theta^j \in R^{n_p}$
$l\theta$: learnable user profile vector where $\theta^j \in R^{n_p}$
$y^{(i,j)}$: ratings user i has given on item j
r(i, j) : 1 if user i has rated item j else its 0
$\theta^{'}$: Sub features
$X^{'}$: Sub preferences

We can represent the feature(χ) and preferences(θ) as shown in table 1. We can represent $G\theta$-χ (group preference - features table)and $l\theta$-χ (individual learnable preference - features)tables similarly.

χ represents the feature vectors and we have taken four items hence our n_m will be 4. θ represents the preference vectors and its collected from the user or leaned from $l\theta - \chi$ table. We have three members in this example hence our n_u will be 3. A sample preference vector θ^1 can be something like this.

$$\theta^{(1)} = \begin{pmatrix} \theta_0^{(1)} = 0.80 \\ \theta_1^{(1)} = 0.10 \\ \theta_2^{(1)} = 0.10 \end{pmatrix}$$

y(1, 1) is 0.80 from the table which represents the actual rating the user 1 had given to item 1 and since 0.87 is pretty close to 1 we can understand that he/she kind of like the item 1. From the table we can see that

$$\chi^{(1)} = \begin{pmatrix} x_0^{(1)} = 0.93 \\ x_1^{(1)} = 0.01 \\ x_2^{(1)} = 0.06 \end{pmatrix}$$

now

$$(\theta^{(1)})^T.\chi^{(1)} = 0.751 \tag{1}$$

and that's our predicted ratings. the y(1, 1) rating from the table is 0.80 and predicted ratings is pretty close and the error in the difference makes our learning component. We can construct a learning function based on this.

$$\gamma = ((\theta^{(j)})^T.\chi^{(i)} - y^{i,j})^2 \tag{2}$$

Table 1 computations are not expensive since a group usually don't have a big n_u and hence its computed on client side. For $G\theta$ and $l\theta$ we need to learn over large datasets and hence these tables are computed on server side.

We know that each query must have a feature vector associated with it for us to make recommendation based on it. So we need to way to represent features vectors to compute predictions and these two sections will explain for food and movie related queries.

1) Feature vectors for food related queries.: Now the question on what kind of features are we considering to make such predictions needs to be explained. Each kind of food or cuisine has some cultural influences over each other and it all goes to culinary history of the cuisine, Authors of this paper have considers the basic tastes as features in this paper[11] and in this work where they have built a food recommendation system for diabetic patients they have used food nutritions as features[8]. However in this work I have taken a different approach. Food historians are unfolding cultural influence of cuisines over each other[13]. For Instance when we say $hotpot$ it originated from Mongolia and spread to all countries so each south Asian countries now have their own version of hotpot. So using this idea instead of treating different cuisines as categorical name tag I will considering them as ordinate data. So if one queries for $hotpot$ its feature vectors would tell us what components of its feature x_i is Chinese and what

component of it is Korean and so on. For instance Mexican food can have a high degree of similarity with indian food because they both are spicy in nature.

2) Feature vectors for movie related queries.: Now for movies it should be easier for us to obtain feature vectors. We have wide range of options for movies like action, comedy, romantic etc. and also we can represent sub features for these features.

3) Sub features ($\theta^{'}$) and Sub preferences ($X^{'}$): Sub features are features embedded inside a feature. To understand sub features lets us consider an example. Say if a user want‘s to visit a restaurant for dinner. now among all the options he/she might be interested to have Chinese food. Now if he/she goes to a Chinese restaurant they have different food options. If yelp or google gives you a rating of 4.5 for that restaurant we don't know the ratings for sub features like Cantonese, Sichuan, Jiangsu, Zhejiang, Fujian, Hunan, Anhui, Shandong. Similarly india food can be broken down into more than 29 sub features based on its provinces.

In case of movies we can get feature ratings for action, comedy, romantic etc and each feature can be broken down into sub features like action can be based on disaster, super hero movies, chase movies etc.

In elastic group recommendation system we can introduce these dynamic features as we learn new features and also train them. On the other hand we can also learn user‘s new preferences vectors like over course of time we can learn that a user likes Cantonese food over schezuan and hence we can make better recommendation.

V. System flow and architecture.

The system extracts three ratings matrix which holds ranks on scale of 0 to 1. these three rank matrices are convenience ranks, group similarity ranks and system predicted ranks. convenience ranks are the ranks given to places based on how convenient these places are to the group or in other words finding a place which is least far from all members. Group similarity ranks are ranks given to the places based on the idea that a group which is has similar preferences as our group must have similar tastes in their choices. Predicted ranks are the ranks which we predict to places based on simultaneous minimization in collaborative and content filtering. These three ranking spaces form a three dimensional space consisting clusters of places. We will apply our final reduction logic to come up with a best ranking list for the group and these ranking chances when users move to different places or change when they change their preferences by interacting with the system.

A. convenience ranks : Query result extraction and convenience feature extraction algorithm.

When a host creates an event he/she will invite his/her friends and when members accept the invites the system will acquire their gps location and its in latitude and longitude form which is a x y coordinate. Now while making decision where to hangout distance plays a vital role, so we need to pick a place which is convenient for all the members. Now convenience is a deduced feature and we will use users location and the place location to deduce this. Inside the application all members can query for the restaurant or movies that they are interested in. For our example we have there members and let their queries be.
q1: sushi
q2: Indian
q3: fast food

or the members want to go for a movie then the queries can be like.
q1: The imitation game
q2: Avengers
q3: Iron Man

1) Query processing.: When each member makes a query we store those queries in our system. Since we don‘t have a database for different restaurants we will use Yelp for our data source. Yelp api usage rules says that we cannot modify their ratings but in the system we are ranking the places and not modifying their ratings and also all the tests follows their api rules. Now for movies yelp doesn't have a database so I have implemented own database for this purpose.

Yelp api will provide us all the places which come inside a geometric boundary. lets take 3 queries and in our case q1 is sushi, q2 is Indian etc. yelp [1] also provides us features of the places which includes location co ordinates, review count, categories, rating, review. The following lines show that for each query we have 10 results where pid11 shows restaurant with id 1 for query 1.
q1: sushi : [pid11, pid12, ..., pid110]
q2: indian: [pid21, pid22,..., pid210]
q3: fast food : [pid31, pid32, ..., pid310]

Here in case of movies we need to consider all theaters which are playing the queried movie so for each query there may be multiple pid‘s.

we can get the traveling distance between co ordinates (users and places) using google‘s location api but for our computation lets just use a simple euclidean distance, however it makes more sense to use haversine formula [4].

In this stage we have 3 set of 10 restaurants matching query q_1, q_2, q_3 which comes inside the polygon formed by users as shown in figure 1. Now we will try to represent all these queries in form of features matrix which we will be used in later processing. Each query will return 10 places and each place has a feature vector $\chi = [x_0, x_1, ..., x_{n_f}]$.

Each place also has a place id, now will need to compute a mean feature to represent individual query. Lets build a $10 \times n_f$ matrix χq_i which holds features for 10 different places for query q_i. Let M_1 be a ones matrix of size 1×10. The equation gives a row matrix which captures mean features for query q_i

$$(\mu\vec{\chi}_{qi})_{1xn_f} = (1/n_f).M_1.\chi q_i \tag{3}$$

$(\mu\vec{\chi}_{qi})$ represents a query vector for a member. We can

TABLE II. A SAMPLE NORMALIZED ITEM USER DISTANCE MATRIX

M	$place_1$	$place_2$	$place_3$	$place_4$
usr_1	0.80	0.09	0.27	0.93
usr_2	0.45	1.0	0.09	0.25
usr_3	0.95	0.51	0.03	0.69
usr_4	0.3	0.87	0.5 0	0.83

then construct a matrix which represent the query formed by the group like this.

$$\chi_{(Q)} = \begin{pmatrix} \mu\vec{\chi}_{q_1} \\ \mu\vec{\chi}_{q_2} \\ \mu\vec{\chi}_{q_3} \end{pmatrix}$$

χ_Q is a matrix of dimension $n_{queries}$ x n_f. $\chi_{(Q)^1}$ represent feature vector for q_1 and so on.
So now we have reduced 30 places to three row feature of matrices. Next we move on to convenience ranks extraction.

2) convenience ranks extraction: we will first construct a matrix in which the columns are users and rows are places and the matrix holds distances from users to places. the table 2 shows how the values are held.

The code below explains the convenience feature extraction algorithm

```
import numpy as np

def main():
        score = dict()
        M = get_usrtoplace_distance_matrix()
        M1 = np.ones(len(M))
        sum_dis = np.dot(M1, M)
        dis = np.array(sum_dis)
        distance = dis[0]
        for x in range (0, len(distance)):
            score[get_place_id(x)] = distance[x]
        rank_list = sort(score.items(), \
                key=lambda x: x[1])
        # rank_list holds places sorted
if __name__ == "__main__":
        main()
```

M is the users to places distance matrix as show in the table 2. Let M_1 be a ones matrix of size $1 \times n_u$.
$M_1 M$ gives us the sum of all user distances to a given place and a row matrix dis will capture this. We will use a dictionary which will hold placeid as key and the net distances as value. The for loop defined there will populate the dictionary with keys as placeids. we then just need to sort it based on its values which will give us a list with which is sorted places based on distance.

B. group similarity ranks : Ranking by measuring group similarities.

In our database the system holds two tables which holds ratings a user gives to a place and ratings the group gives to that place and these ratings are stored as they are used for learning.

TABLE III. A SAMPLE INDIVIDUAL PREFERENCE AND FEATURES TABLE IN DATABASE($(\theta - \chi$ TABLE)).

TB1	$user_1(g\theta_1)$	$user_2(g\theta_2)$	...	$user_{N_u}(g\theta_{N_u})$
$place_1$	0.31	0.66	0.54	0.28
$place_2$	0.56	0.41	1.0	0.24
...	0.75	0.51	0.03	0.69
$place_{N_m}$	0.41	0.87	0.5 0	0.83

TABLE IV. A SAMPLE GROUP PREFERENCE AND FEATURES TABLE IN DATABASE ($G\theta - \chi$ TABLE).

TB1	$G_1(G\theta_1)$	$G_2(G\theta_2)$	...	$G_{N_u}(G\theta_{N_u})$
p_1	0.80	0.09	0.27	0.93
p_2	-	-	0.09	0.25
...	0.95	0.51	0.03	0.69
p_{N_m}	-	0.87	0.5 0	0.83

When a member i gives rating to place j we store that rating $y^(i,j)$ in our $l\theta - \chi$ table. Now for $G\theta - \chi$ table we can just take mean of all the members and store the ratings in our group table $G\theta - \chi$ but it doesn't justify fact that the place$j$ may have a feature which the user i dislikes the most. Lets say a user dislikes spicy food and the group decides to go to Indian Andhra cuisine which is know for its spiciness. In this case no wonder the user j will give bad ratings and that doesn't mean that place is bad. we need to give weighted to the ratings given by user k who appreciates spicy food. So we need to take ratings given by individual members and compute weighted average and that can be done by these equations.
$\forall$ user i we will compute the similarity between preference θ and feature χ for item j using this equation.

$$\forall useri, itemj \qquad \beta_{i,j} = \frac{(\theta)_i^T.(\chi_j)}{||\theta_i|| * ||\chi_j||} \tag{4}$$

Now we can calculate ratings $Gy^{i,j}$ for item j given by group i using this equation.

$$\forall useri, itemj \qquad Gy^{i,j} = \sum_{i=0}^{n_u}[\frac{\beta_{(i,j)}.y^{(i,j)}}{n_u}] \tag{5}$$

In the database we also hold individual preference vectors and group preference vectors. The following tables depict how they can be held. The $table1$ holds place id's in its rows and user ids in its column and each entry into the block is ratings that $user_i$ has given to $place_j$. For each user we hold their global preferences $g\theta$ and for each place we hold place features. $table2$ is formed when a users form a group. We take individual user ids and just append to each other and compute a sha1 hash[] and this becomes an entry in our table 2. here is an example.
$user_1Id$: 345123
$user_2Id$: 787980
$user_3Id$: 586987
$Shahash(users)$: $hash(345123787980586987)$
$groupid$:
78d8505a06a95591e5f4c32a9a6a4
fa3d9f069ce1fec509b6e9a1493f11bd3ce

So now next time when a group hangs out when just need to append all the userid's compute sha hash and check if that entry is in $G\theta - \chi$ table.

If the group has not hung out before then we will not have an entry in our group table and thus we have no idea what this group might like or not so for now we will go head and create a new column in our group table and assign a new group id to it. When a user registers to the application he/she would have given some basic preferences like simple re ordering of cuisines like a user might like Asian food and its at the top in his/her list and in cases of movies action, comedy etc are ordered based on their choices. We take these preferences and use it as a kick starter. we now compute a mean preferences to represent this group.

$$\mu\theta = \begin{pmatrix} \mu\theta_1 = (\sum_{i=1}^{n_u}(\theta_1^i))/(n_u) \\ \mu\theta_2 = (\sum_{i=1}^{n_u}(\theta_2^i))/(n_u) \\ \cdots \\ \mu\theta_{n_p} = (\sum_{i=1}^{n_u}(\theta_{n_p}^i))/(n_u) \end{pmatrix}$$

our $\mu\theta_1$ is a straight mean of first preference and so on. We now need to find a group which is similar to our group i.e a group in our database which matches our $\mu\theta$. We can use $cosinemeasure$ to find similarity but as dataset grows it can be expensive so we can use technique like $KNearestNeighbor$ algorithm to find the most similar group. KNN works best for clusters like this[12]. Once we have found a nearest neighbor we now have access to all the restaurants he/she has been and thus we also have access to the features of those places.
for example if the most similar group to our group is say G_{20} and this group has tried out 15 different restaurants. Which means we have 15 feature vectors. In the section 5.1 we have computed the features matching the query (θ_q). We now compute cosine similarity to each features and pick the ones that's most similar. To retrieve all the places that group has been we can use a technique called item to item filtering which is explained in the next section.

$$\chi_{g20} = \begin{pmatrix} \vec{\chi_{n_{p1}}} \\ \vec{\chi_{n_{p2}}} \\ \cdots \\ \vec{\chi_{n_{pi}}} \end{pmatrix}$$

where $\chi_{n_{p1}}$ is a row feature vector of a restaurant 1 that group $g20$ has hung out. Now we can find out the most preferred feature vector by this equation.
$\forall i : 0- > n_{pi}$

$$fitness = min(\frac{(\mu\chi).(\chi_i)^T}{||\mu\chi|| * ||\chi_j||}) \tag{6}$$

Of course we need to a pick a group which has hung out quite often, a group which has hung out quite often and most similar to our test group yields the best matching features.

We now have all the restaurants the groups has been which matches our query preferences. If we plot them on a feature space we can see that they make clusters. we just need to find which cluster has least distance to our $G\theta$ and we will pick the top rates restaurant from that. Next we pick the next closest cluster and so on. for each group nearest neighbor we check we will compute the fitness value which measure how close the place is matching our group query. Ideally our objective here is to pick a place which yields high similarity values for group similarity and query and places similarity. We do

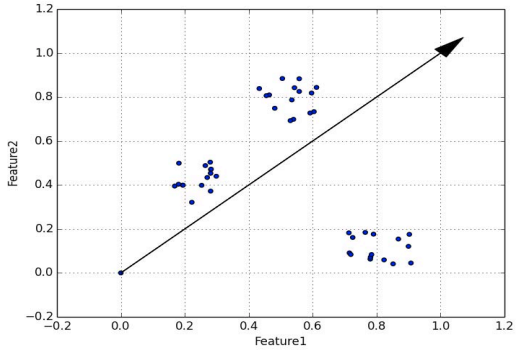

Fig. 2. places cluster over group similarity.

the above process until all the restaurants are picked and that forms our ratings2.

In figure 3 we can see many cluster belonging to a type of cuisine or movie. Each point or node represents a restaurant or a movie. This plot is shown over a two dimensional space however each point would be attributed to higher dimensional space. The vector that each node makes with origin is a feature vector for that place and the vector that's drawn in center with an arrow represents a group preference vector. A node which is close to the group preference vector is the best choice in this sample. The actual distance is the fitness value for this sample. We need to do this for next nearest neighbor and compute fitness value and so on. Finally a sample which has high fitness and high group similarity will be the best choice and ranked higher.

C. Ranking based on our Predictions.

Now this is where our core recommendation process starts. The central idea of this system is to make recommendations to the group using three rating vectors and then reduce it to a list ranked by our reduction logic. Our reduction logic will be explained in the later sections. $Ratings1$ and $Rating2$ were extracted in previous sections and now we need to extract one more ratings vectors.

1) Hybrid item to item filtering: This filtering algorithm was popularized by $Amazon$[7].The central idea here is multiple groups with similar tastes and preferences enjoys to hangout at similar places and this is how we listed places in the previous section.

If the group has hung out before then we have groups preference vectors which makes it easy for predicting a place. When the group hung out last time the application will ask each user to rate their experience. In content based filtering technique we have χ and we try to learn θ and this can be done by minimizing the cost function $J(\theta)$ $\forall$ θ_j where j runs from 0 to n_u. We can learn the θ like this

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} (((\theta^{(j)})^T.\chi^{(i)} - y^{i,j})^2)\chi_k^{(i)} + \lambda\theta_k^{(j)} \tag{7}$$

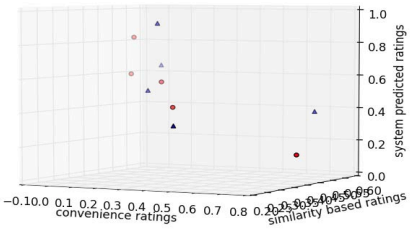

Fig. 3. Ranks reduction.

where α is the learning rate and λ is regularization component to avoid over fitting. Now this learning is for one user with learning preference vector k. For learning the preferences for all the users in our dataset we have to minimize it for each user i.e $\theta^{(1)}...\theta^{(n_u)}$.

$$min \sum_{j=1}^{n_u} [\sum_{i:r(i,j)=1} (((\theta^{(j)})^T.\chi^{(i)} - y^{i,j})^2)\chi_k^{(i)} + \lambda \sum_{k=1}^{n_p} (\theta_k^{(j)})^2] \tag{8}$$

and for collaborative filtering we go from $\chi^{(1)}...\chi^{(n_m)}$.

$$min \sum_{j=1}^{n_m} [\sum_{i:r(i,j)=1} (((\theta^{(j)})^T.\chi^{(i)} - y^{i,j})^2)\chi_k^{(i)} + \lambda \sum_{k=1}^{n_f} (\chi_k^{(i)})^2] \tag{9}$$

In real time we can simultaneously minimize the cost function $J(S_{(\chi,\theta)})$ like this.

$$\sum_{(i,j):r(i,j)=1} (\gamma^2) + \lambda \sum_{i=1}^{n_m} (\sum_{k=1}^{n_f} (\chi_k^i)^2) + \lambda \sum_{j=1}^{n_u} (\sum_{k=1}^{n_p} (\theta_k^j)^2) \tag{10}$$

where γ is from equation 2.

So the simultaneous minimization [6] helps the system to learn user preferences and item features. Once learnt we can find a most similar group to our group. Once we have a most similar group we can apply item to item filtering and matching the query as explained in previous sections with group similarity ranks. Once the system has learnt pretty good results and our evaluation yields least errors we can calculate the predicate ratings as explained in section 4. Here if the user has not hung out we will use θ and if we have $G\theta$ then we will use that for computing predictions. Finally we will apply rank reduction technique where a point whose cosine distance to (1, 1, 1) is rated higher and thus reduces to a rank list.

VI. Conclusions

In this paper an attempt to get an elastic group recommendation was made. The system is currently under use through an android app called $Metster$. To capture the entire work the system used individual preference and predicted group preferences. The system does filtering at three stages calculating convenience ratings, similarity ratings and predicted ratings. convenience was calculated based on distance measure of the user at that moment. similarity ratings were calculated by finding a most similar group to our group and applying item to item filtering on the items close to our query. We then ranked our items based on how close their features were. predicted ranking were calculated using group preference if available else we used learned individual preference. We applied simultaneous minimization on collaborative and content filtering to improve our predictions. In the last stage we reduced all the ranking into a final rank based on distance to unit vector.
This work was intended to develop a recommendation system which works best for all members in group and yet satisfying individual preferences.For a next stage of improvement we can consider these attributes mentioned in this work[3] to bring this system much closer to users.

References

[1] M. Grant, J. Krzysztof, and A. Benjamin. Weighted multi-attribute matching of user-generated points of interest. *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 440–443, November 2013.

[2] J. B. Harvey. The abilene paradox: The management of agreement. *Elsevier : Organizational Dynamics*, 1988.

[3] C. Ivn and C. Pablo. Group recommender systems: New perspectives in the social web. *Springer Berlin Heidelberg: Intelligent Systems Reference Library*, 32:139–157, January 2012.

[4] Y. H. Jovin J. Mwemezi. Optimal facility location on spherical surfaces: Algorithm and application. *New York Science Journal.*, 4(7):21–28, July 2011.

[5] K. Junzo, N. Yuji, U. Kazunori, K. Keishi, S. Shinji, and M. Hideo. A tv news recommendation system with automatic recomposition. *First International Conference, AMCP, Osaka, Japan*, 9(11):221–235, November 1998.

[6] B. Justin and H. Thomas. Unifying collaborative and content-based filtering. *ICML Proceedings of the twenty-first international conference on Machine learning*, pages 9–9, July 2004.

[7] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Computer Society*, 7(1):76 – 80, January 2003.

[8] P. Maiyaporn, P. Phathrajarin, and P. Suphakanth. Food recommendation system using clustering analysis for diabetic patients. *IEEE International Conference on Information Science and Applications (ICISA)*, pages 1 – 8, April 2010.

[9] O. Mark, C. Dan, A. K. Joseph, and R. John. Polylens: A recommender system for groups of users. *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*, 16(20):199–218, September 2001.

[10] L. Ning-Han, L. Szu-Wei, C. Chien-Yi, and H. Shu-Ju. Adaptive music recommendation based on user behavior in time slot. *IJCSNS International Journal of Computer Science and Network Security*, 9(2):219–227, February 2009.

[11] H. Ningxuan, L. Mengyuan, and Z. Fang. A chinese dishes recommendation algorithm based on personal taste. *2015 IEEE 2nd International Conference on Cybernetics(CYBCONF)*, pages 277 – 280, June 2015.

[12] Z. Qingjiu and S. Shiliang. A centroid k-nearest neighbor method. *6th International Conference, ADMA 2010, Chongqing, China*, 6440(21):278–285, 2010.

[13] R. Shepherd and M. Raats. *The Psychology of Food Choice(Frontiers in Nutritional Science)*. CABI (April 6, 2010), 2010.