Sales and ratings based recommender systems

Ismail Bouanani and Nacer Zaim Wadghiri

***Abstract*—Whether it is for suggesting relationships in social media or highlighting items in online shopping websites, relevant recommendation systems are essential to intensify consumption from the users. The availability of huge amounts of data make it possible to assume the user's evaluation of a product. The data can be used in various ways as we are about to see. Indeed, we are comparing two methods to recommend items : One that is user-independent, based on the joint sales of items and the other ones based on the ratings users made on items, by inferring the missing ratings.**

I. Introduction

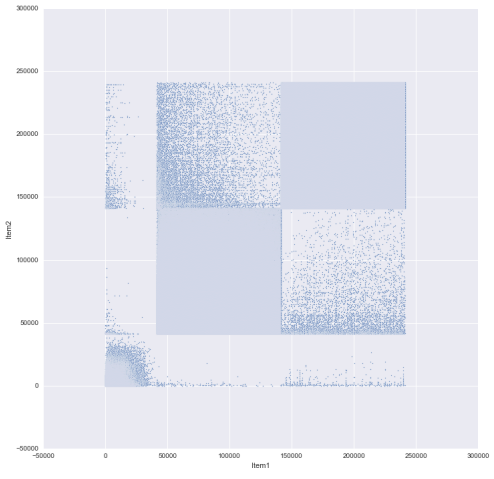
We are working on data from the Amazon dataset comparing here two methods to recommend items. The first one , uses the previous cases of joint sales. Two items that were bought by the same user are declared as similar. The second one performs on the ratings data, a sparse matrix in $\mathbb{R}^{u*i}$, where u=5226 is the number of users and i=2321 the number of items. We apply a method called Matrix factorization with Alternate Least Squares that is supposed to fill the missing entries and assume the ratings $r_{i,j}$ on a 1 to 5 stars scale. The goal is to minimize the RMSE cost function, defined as follows:

$$RMSE(\hat{R}) = \sqrt{\frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} |r_{u,m} - \hat{r}_{u,m}|^2} \qquad (1)$$

Where $r_{u,m}$ is an available rating and $\hat{r}_{u,m}$ is its version evaluated by the model. In the upcoming sections, we will briefly describe the sales based method (mathematically it does not have much details), explain the properties and pre-processing of the ratings dataset, describe the ALS algorithm and how its parameters were tuned and finally, analyze the difference of the relation graphs obtained with both methods visualized on the 50 most popular items of 3 categories (Electronics, Books and CD's).

II. Sales Based Recommendations

One of the datasets contains the metadata of Amazon's products. A sample of this dataset contains these features : the item's identifier (also called ASIN number), a description of the item,its price, its salesRank in its category and the related items in different ways("also-bought","also viewed","sold together","bought after viewing"). We first clean out the irrelevant features (price,description and relationships other than "also bought"). We are working on three categories of items : Electronics, Books and CD's but the properties mentioned in our research work can be generalized to more categories. The recommendation here is simple , if two items belong to each other's "also-bought" list, then they are recommended for each other. We decided to highlight and visualize these relationships on the top 50 (50 best salesRank in the available dataset , not necessarily the top 50 ones in the whole engine) of the three categories . We will analyze our results later and compare them with ones from the second method. Before realizing the graph, the figure below announces the main shortcoming of this assumption: Items are only related within their category, which makes sense . Indeed we notice three big squares (we indexed first the books,then the CD's and finally the Electronics) . This is not good, we need to be able to come up with the customer's needs even in categories he did not buy from, by finding inter-category similarities.



III. Ratings based recommendations

A. Data and pre-processing

The first step, is to take the ratings dataset, keeping only the top-1000 salesRank product per category (so a total 3000 products).The reason behind that is to make computations feasible (the whole dataset comprises more than 2M items !).Still, this dataset is very sparse as the ratings are made by over 72'000 different users, most of them having very few ratings. The next step is to filter out the ratings made by these. That was done in order to reduce sparsity and minimize the risk of obtaining a singular matrix (zero determinant making computations impossible) after cutting the data into train and test set. Therefore, we decided to discard the users who have less than 5 ratings. This gives us a much less sparse matrix with 46622 ratings over the 5226*2321 $\geq$ 11M . But yet,this is still too sparse to apply user or item based collaborative filtering , as we can see in Fig. 1 so a model-based matrix-factorization method was preferred here.

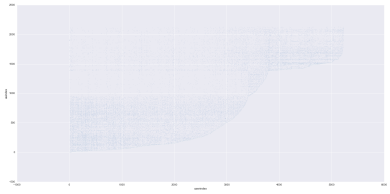


Fig. 1. pairplot representing the available (user,item) ratings

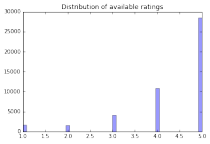


Fig. 2. distribution of the available ratings

B. ALS

Alternate Least Squares is the name of the model we will be fitting. It has the reputation to be appropriate for very sparse matrices. ALS is an iterative matrix-factorization algorithm. The idea behind this algorithm is to approximate the rating matrix R by the product of two low-rank matrices U and I (for users and items) of rank n_f (That is, our prediction matrix $\hat{R}$ is equal to the product of the matrices U^T and I) such that the RMSE in (1) is minimized. This model is based on a collaborative filtering approach which takes in account both users and items to make the predictions. The parameter n_f is called the dimension of the feature space.

We observe that with this method, the number of parameters to estimate n_f (i + u) (the total number of entries of the two matrices) can be huge if n_f is large, so ALS might lead to overfitting. To tackle this problem, we have to add regularization terms to the cost function. We learned from Equation 1 that for this kind of problem, the Tikhonov regularization described in [1] works well. Moreover, the regularization will also tackle the problem of the noise. Here, the meaning of the noise is that each user can be biased by his feelings at a given time (see Fig.2 , a wide majority of 5 stars are given). Thus, by taking this regularization, we will take the general behavior of each user and dont overfit by taking in account those user's feelings. In a sense, if the model take in account this noise, then it will take in account those periodic feelings into the new predictions. The cost function becomes :

$$\zeta(U,I) = \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} |r_{u,i} - (U^T M)_{u,i}|^2 + \tag{2}$$

$$\lambda(\sum_k n_{u_k} ||\mathbf{u}_k||^2 + \sum_j n_{i_j} ||\mathbf{i}_j||^2) \tag{3}$$

where n_{u_k} and n_{i_j} denote the number of ratings of user k and item j respectively, and $||\mathbf{x}||^2$ is simply usual the squared norm of the vector **x** (u_k and i_j are the vectors of the ratings of user k and item j respectively).
The algorithm works this way :

1) Initialize the item matrix I by assigning the average rating for that item as the first row, and small random numbers for the remaining entries
2) Fix I, solve for U by minimizing the above cost function
3) Fix U, and update I similarly
4) Repeat step2 and 3 until a stopping criterion is satisfied. (Either a maximum number of iterations is reached or the difference of RMSE between this step and the previous one is small)

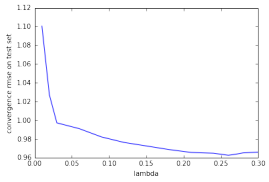


Fig. 3. Value of converging testing RMSE for every value of λ .

In step 2 , we consider i as fix, so minimizing the cost function is equivalent to solving its differentiation w.r.t to U being null,
we set :

$$\frac{d\zeta(U,I)}{du_{k,l}} = 0 \forall k,l \tag{4}$$

that is how we compute the vector u_k , as the solution that takes the value:

$$u_k = (I_{Id_l} I_{Id_l}^T + \lambda n_{u_l} Id_{n_f * n_f})^{-1} = I_{Id_l} R(l, Id_l) \forall l \tag{5}$$

Where *Id* is the identity matrix. we compute i_j in a similar manner. The solutions depend on our two main model parameters : λ and n_f, it is important to find out how to tune them.

C. Parameters tuning

First, let us tune λ as it is the most critical one. Finding its optimal value, would avoid us to overfit the training set. We fix n_f to an arbitrary value of 30 and vary λ to train with different values of it. For each training (on a training dataset representing 90 % of our available ratings), we yield the performance: RMSE evaluated on the testing set. After the algorithm described in the previous section converges (either maximum number of iterations, fixed to 50 is reached or difference between two consecutive RMSEs is small, under a fixed 0.0001). That gives us the following graph for λ taking values between 0.01 and 0.3 :

As , we can see in 3, the gloabal minimum is reached at the value of λ = 0.26. So we chose it as our optimal value. The corresponding testing RMSE is 0.9626

Now, let us tune n_f :
We fix λ to 0.26, its optimal value. Now,we vary n_f on a 10 to 100 scale, by step of 10 and yield the RMSE on test set each time . This give us the graph depicted in Fig.3 We can notice that from a value to another, the difference of performance is very small, though the global minimum is reached for n_f=40, so we decide to go for this value, with corresponding testing RMSE of 0.9620

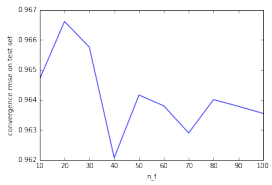


Fig. 4. convergence RMSE on test set for various n_f values

D. Bias Term

As we explained earlier, we assume a certain bias due to user emotions and motivations . That probably explains, the huge amount of 5 stars ratings. Therefore, we need to resort to a bias correction technique. We define the case-specific bias as : $b_{u,i}$ = $r_{u,i}$-$\hat{r}_{u,i}$, where $\hat{r}_{u,i}$ is the entry computed by our model matrix $\hat{R}$ and $r_{u,i}$ the actual rating, for user u and item i in R. The global average bias b is computed as the mean $b_{u,i}$ among the previously known ratings , meaning:

$$b = \frac{1}{|\Omega|} \sum_{u,i \in \Omega} b_{u,i} \tag{6}$$

Plus, we have

$$b = \frac{1}{|\Omega|} \sum_{u,i \in \Omega} r_{u,i} - \hat{r}_{u,i} \tag{7}$$

$$\leftrightarrow \sum_{u,i \in \Omega} r_{u,i} - \hat{r}_{u,i} + b = 0 \tag{8}$$

$$\leftrightarrow \sum_{u,i \in \Omega} r_{u,i} - \hat{r'}_{u,i} = 0 \tag{9}$$

where $\hat{r'}_{u,i}$ = b + $\hat{r}_{u,i}$, so we expect the average bias for the matrix $\hat{R}$ + b (adding b to every element of $\hat{R}$) to be zero. For λ = 0.26 and n_f = 40, we find a bias of 0.402. When we add it to $\hat{R}$, the testing RMSE decreases to 0.8688

E. Final model and post-processing

Here is the summary of our final chosen model :

1) $\lambda = 0.26$
2) $n_f = 40$
3) bias =0.402 testing RMSE =0.868

The out of scale values (less than 1 and more than 5) are brought to the values 1 and 5. Now, we have to compute the item to item graph as we did for the other recommendation method. We come up with the following rule to say an item is related to another one : item1 points to item2 if all of item1's are assumed to like item2 by $\hat{R}$ with a score $\geq$ 4.75. This is actually a strong link between two items as the recommendation is systematic for every user who has bought item1. We could have chosen a less strict rule, but these give us a graph that is too dense and hardly analyzable.

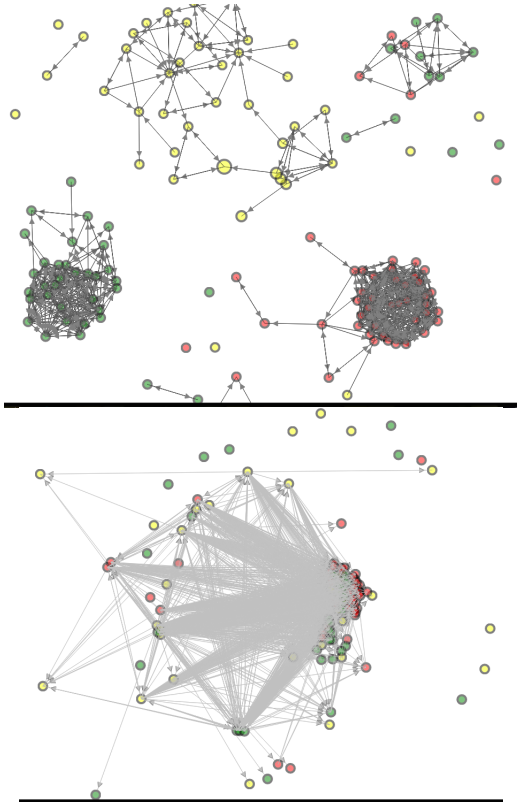


Fig. 5. Force graphs obtained with sales based (top) and ratings based(bottom) approaches

IV. Graph comparison

A. Analysis

Here is a snapshot of the graphs obtained with both methods (Animated version featured in the repository).

As we were expecting it, the first graph shows a clustered structure, where all the items in a given cluster belong to the same category. This confirms the fact that sales only reveal relationships between actually similar items. Most of the similarities are actually underlying and have to be infered.Therefore, we need collaborative filtering in order to yield reputation based similarities between items. Infering those similarities works pretty well in the second graph. As we can see, there is a giant component gathering most of the items (thus, within different categories), it looks like almost any node can be reached by any other one, through a series of recommendations, example: a customer buying a laptop can find himself buying a Pop songs CD in a few clicks time , although these items don't have anything to do with each other. The whole graph can be covered within a few hops. Minor point for the the second graph, a lot of nodes (even from the top 50) were filtered out during the pre-processing phase prior to training ALS (28 out of 150 items). Those, are not even featured in the graph and thus, do not have recommendations.

B. Metrics comparison

Let us analyze the structure of the graphs in a more concrete way in order to confirm the statements made in the previous part :

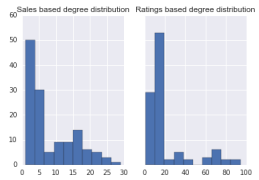


Fig. 6. Distribution of node degrees in both realized graphs

1) Node degrees distribution :
   The average number of neighbors for a node in the first graph is 10, it is twice higher in the second. This means more recommendations are made by the second system after one buys a single item. The distribution can be viewed in the histograms Fig.5 . We can notice that the range of values is much wider in the second graph. The majority has between 10 and 20 neighbors whereas in the first graph, over half the nodes have a degree of less than 5. As mentionned, the second graph features star items (with degrees between 40 and 70), the maximum node degree reached in graph 1 is 30.
2) Transitivity :
   The transitivity of a graph (or global clustering coefficient) is defined as follows :

$$T = 3X\frac{numberoftriangles}{numberofconnectedtriples} \tag{10}$$

   This characterizes how chunky and clustered a graph is. Naturally, it is twice higher in the graph 1 where T=0.63 than in the second graph where T = 0.31. This reveals a clique-rich structure for the first graph.
3) Giant component existence A giant component (subgraph connecting more than 50 % of the items) is a valuable feature in a graph. It indicates a possibility of crossing the whole graph starting from and all the way to every node (except the isolated ones). The largest subgraph of graph 1 occupies a poor proportion of 0.27. In graph 2, over 85 % of the nodes are comprised in it.

V. Conclusion

Due to the last mentioned feature, we can vote in favor of the second recommendation system. Indeed, more recommendations are made with it and every item is involved and can be sold to every user(If I buy a product, a series of recommendations can lead me to any other one). Plus, items from different categories can recommended for each other without any kind of similarity in their content. However, it involved a filtering in the pre-processing phase that made us lose quiet some nodes from the data, making us unable to infer their matching with users. Thus, one can add merge the first recommender system to the second, as a default recommendation, in case these are not available for the item. Indeed,joint sales can address the cold start issue of collaborative filtering(items not or not much rated,can be related to the ones bought by the same single/few user(s))

References

[1] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan. Large-scale Parallel Collaborative Filtering for the Netflix Prize. HP Labs, 1501 Page Mill Rd, Palo Alto, CA, 94304.