Introduction

The explosive growth of the world-wide-web and the emergence of the e-commerce drives us to develop a recommender systems. Compute the similarities between pairs of elements allows us to determine the item that is most relevant (or similar) to a given item. This property can be applied to recommender system. For the customer-product database, compute the similarities between customers allows us to cluster then into groups with similar interest about buying products. And compute the similarities between customers and products allows us to suggest products to buy. We used three different methods, which include pseudoinverse of the Laplacian matrix(L+), matrix-forest-based algorithm (MFA) and k-nearest neighbors (kNN), to calculate the similarities between nodes.

Data

The dataset we used is the amazon customer-product dataset. We generated a bipartite network with nodes consist of customer set and product set. Each node of customer set is linked to each product set reviewed by the corresponding customer. In the data, there are 6280 customers nodes and 10,000 products nodes.

Recommendation methods

1. Direct method:

Use each scoring algorithm to compute the similarities between a given person and all the products. Products are recommended in decreasing order of the similarity values.

2. User-based indirect method:

Use each scoring algorithm to compute the similarities between a given person p0 and all other persons. Use the computed similarities to find k nearest neighbors of person p0, then for a product m0, the predicted value is $\frac{\sum_{p=1}^{k} sim(p0,p)a_{pm0}}{\sum_{p=1}^{k} sim(p0,p)}$. Products are recommended in decreasing order of the predicted values.

3. Product-based indirect method:

Use each scoring algorithm to compute the similarities between a given product m0 and all other products. Use the computed similarities to find k nearest neighbors of product m0, then for a person p0, the predicted value is $\frac{\sum_{m=1}^{k} sim(m0,m)a_{pm0}}{\sum_{m=1}^{k} sim(m0,m)}$. Products are recommended in decreasing order of the predicted values.

4. Similarity measures:

k-nearest neighbors, pseudoinverse of the Laplacian matrix and matrix-forest-based algorithm

List of algorithms

- K-nearest neighbors (kNN)
  Each person i is characterized by a binary vector vi encoding the products that that person purchased. The similarity between two persons i and j is calculated as $sim(i,j) = ||vi - vj||_2$
  Each product i is characterized by a binary vector vi encoding all persons that purchased it. The similarity between two products i and j is calculated as $sim(i,j) = ||vi - vj||_2$
- Pseudoinverse of the Laplacian matrix (L+)
  L+ is a kernel matrix where l_{ij}^{+} is the inner products of the node vectores x_i and x_j in the Euclidean space. Thus l_{ij}^{+} can be used as a similarity measure between node i and node j.
- Matrix-forest-based algorithm (MFA)
  Based on the matrix-tree theorem, the ‘relative forest accessibilities‘ between nodes can be calculated by T=(I+L)-1, where L is the Laplacian matrix and I is the identity matrix. The ‘relative forest accessibilities’ can be treated as the similarities between nodes.

Evaluation Metrics

- Degree of agreement
  Let P1 contains all products a person has purchased and P2 contains all products a person hasn’t purchased. A pair (p1,p2) where p1 $\in$ P1, p2 $\in$ P2 is ranked in correct order if p1 precedes p2 in our recommendation list. The degree of agreement is the percentage of pairs ranked in correct order with respect ot the total number of pairs.
- Percentile
  The position that the median product in the list form the test set occupies in our recommendation list.
- Recall n
  The proportion of products from the test set that appear among the top n of the recommendation list.
- 10-fold cross-validation
  To get a better estimate of the performance of our recommendation methods, we performed 10-fold cross-validation for each method. The original data set was divided into 10 subsets. In each run, one of the 10 subsets was used as the test set while the other 9 subsets together were used as the training set. The average result over the 10 runs were used as our final result and corresponding standard deviation was also calculated.

Results and Discussion

Below is the table containing degree of agreement, percentile, recall10 and recall20 values for 8 recommendation methods. (k=100 is used for indirect methods)

	kNN		MFA			L+		
	user	product	user	product	direct	user	product	direct
Agreement	0.8720	0.9592	0.9896	0.9976	0.9749	0.9918	0.9985	0.4912
std	0.0010	0.0002	0.0015	0.0010	0.0076	0.0019	0.0008	0.0009
Percentile	0.0491	0.0152	0.0057	0.0052	0.0163	0.0053	0.0052	0.5441
std	0.0010	0.0038	0.0002	0.0000	0.0039	0.0003	0.0000	0.0021
Recall 10	0.1015	0.0887	0.1062	0.1395	0.0145	0.1310	0.1403	0.0007
std	0.0008	0.0075	0.0142	0.0092	0.0460	0.0205	0.0085	0.0002
Recall 20	0.1849	0.1723	0.2290	0.2836	0.0293	0.2692	0.2852	0.0011
std	0.0007	0.0119	0.0230	0.0103	0.0927	0.0318	0.0085	0.0002

Based on the result shown above, the L+ scoring function with product-based prediction method exhibit the best performance. In general, the L+ scoring function shows better performance than that of MFA and kNN. However, L+ scoring function has very low performance when use direct prediction method. The simple kNN method has the lowest prediction compared with MFA and L+. The standard deviation for all three method are small, which demonstrate that all three methods are stable.

Below is the table containing recal10 value for k = 1, 10, 20, 50, 100

	kNN		MFA		L+	
	user	product	user	product	user	product
k=1	0.0488	0.0003	0.14535209	0.14063517	0.14535209	0.14535209
std	0.0048	0.0002	0.00178131	0.01342134	0.00178131	0.00156472
k=10	0.1137	0.1241	0.13300425	0.14189314	0.14467622	0.14510793
std	0.0161	0.0019	0.01635794	0.00669808	0.00104781	0.00160917
k=20	0.1205	0.1100	0.13453623	0.13982838	0.14466207	0.14549894
std	0.0175	0.0023	0.01709131	0.008.352	0.00152471	0.00103032
k=50	0.1238	0.0982	0.13078733	0.13901274	0.14456122	0.14536624
std	0.0157	0.0048	0.01961354	0.0092145	0.00150195	0.00078932
k=100	0.1015	0.0887	0.1062	0.1395	0.1310	0.1403
std	0.0008	0.0075	0.0142	0.0092	0.0205	0.0085

Below is the table containing recall 20 value for k = 1, 10, 20, 50, 100

	kNN		MFA		L+	
	user	product	user	product	user	product
k=1	0.0980	0.0006	0.2906864	0.2837968	0.29068648	0.29068648
std	0.0093	0.0003	0.0015642	0.0194657	0.00156472	0.00156472
k=10	0.1880	0.2447	0.2757820	0.2872682	0.28993454	0.29036978
std	0.0213	0.0018	0.0198409	0.0060252	0.00160917	0.00122626
k=20	0.2054	0.2165	0.277535	0.2843223	0.29055379	0.29055025
std	0.0239	0.0041	0.0229056	0.0080684	0.00103032	0.00125782
k=50	0.2186	0.1910	0.270868	0.2838110	0.28989915	0.29104565
std	0.0225	0.0080	0.0282272	0.0090802	0.00078932	0.0012809
k=100	0.1849	0.1723	0.2290	0.2836	0.2692	0.2852
std	0.0007	0.0119	0.0230	0.0103	0.0318	0.0085

For kNN-userbased recommendation method, k=50 gives the best result; For kNN-product based recommendation method, k=10 gives the best result; For MFA-userbased method, k=1 gives the best result; For MFA-userbased, L+-userbased and L+-productbased methods, k doesn't seem to have a significant effects on the performance.

Experimental running time analysis

The time used to generate training and test data is 221 seconds. The time listed below are the execution time for each method including evaluation time. The results show that all three methods doesn't scale very well for large database. Compared with MAF and L+, the kNN method is the most efficient method.

	kNN		MFA			L+		
	user	product	user	product	direct	user	product	direct
Time (seconds)	17902	18540	23709	22372	23479	19635	19609	19609

- How to run our code:

1. The folder structure

Recommender

--amazon-meta.txt: original amazon meta-data.
--dataprocess.py: a python file used to generate test and training data.
--recommender.py : a python file used to generate evaluation and prediction result.
--recommender.pbs: a pbs file used to submit job with one method on cluster.
--test.sh: a bash file used to submit job with all the methods (8 in total) on cluster.
raw_data : a folder used to store the input file, which generated by dataprocess.py.
test: a folder used to store pbs files generated by test.sh

2. How to run code on palmetto

We are using python3.4 or python3.6 installed by using anaconda3/5.1.0 on palmetto. To run the code, you need following dependencies.

bottleneck
numpy
sklearn
scipy

If you don't have python3.4 or python3.6 on palmetto, you can follow the instructions below to install the python on palmetto.

module add anaconda/5.1.0
conda create -n python3.4 python=3.4 anaconda
source activate python3.4
pip install bottleneck
pip install numpy
pip install sklearn
pip install scipy

How to use the code:

- dataprocess.py:
  python dataprocess.py
  This command will generate the training and test data for k-fold cross validation and k is 10 in our project.
- recommender.py
  python recommender.py --help
  This command will show you how to use the recommender.py.

  **python recommender.py --train='raw_data/train_all_1.npz' **
  **--test='raw_data/test_all_1.npz' **
  **--k=100 **
  **--output=evaluation.txt' **
  **--score_func="L+" **
  **--pred_M="User" **
  --pred_ID=0

This is the example of how to run the code

qsub recommender.pbs
*This command will submit the job on palmetto, which use "L+" scoring function and "User" prediction to predict. You may need change the "**source activate python3.4** " to your python env under anaconda.*

sh test.sh
This command will submit all the prediction methods (8 in total) on palmetto.

- Reference
F. Fouss, A. Pirotte, J. Renders and M. Saerens, "Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation," in IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 3, pp. 355-369, March 2007. doi: 10.1109/TKDE.2007.46