A incremental clustering method for preference similar users

Shengnan Hao, Haibin Liu

College of computer science and technology, Donghua University, Shanghai, 201620, China
School of Business Administration, Hebei Normal University of Science &Technology, Qinhuangdao 066004, China
e-mail: ghliu@dhu.edu.cn

Abstract. With the development of social networks, services and applications based on social networks have become more and more abundant. While providing convenient services and applications for users, the providers have also accumulated many user information. How to use information to provide users with accurate and high-quality services is an issue that needs to be solved. The division of user groups is the basis for solving this problem. User preference is the important information for division, and clustering is the means of division. In order to solve the problem of user group partition efficiency in big data environment, this paper presents a new user preference similarity calculation method. On this basis, initial clustering based on Fast Unfolding algorithm and how to determine the attribution classes of update nodes based on the initial clustering results, the clustering method of incrementally preferring similar users is proposed. Experimental results show that the proposed algorithm greatly improves the division efficiency of user groups.

1. Introduction
In recent years, the rapid development of the Internet has led to a sharp increase in network users, and a large number of social network based services and applications represented by douban and micro blog in China have sprung up rapidly. So how to mine valuable information from these massive user information has become the focus of current research. Because the implementation of user clustering will have great significance in many fields [1]. For social applications, clustering user groups with similar preferences can not only realize fast positioning of information sources [2-3], but also facilitate application promotion and accurate advertising [4]. For online e-commerce applications, through the clustering of users with similar preferences, accurate marketing can be achieved through personalized content recommendation [5], giving users a good use experience, etc.

Due to the importance of user clustering, many clustering algorithms have been proposed[6-7], such as the Fast Unfolding algorithm proposed by Blondel et al [8].These algorithms have achieved very good clustering results. However, the actual network structure tends to change dynamically with time [9]. In order to make the clustering result always as accurate as possible, it is necessary to perform the clustering algorithm frequently. In this case, the traditional clustering algorithm can’t meet the actual needs. Taking the Fast Unfolding algorithm as an example, the main disadvantages of the traditional static clustering algorithm are as follows:

(1) Due to the instability of the algorithm, the division results after each algorithm execution inconsistent. However, the change of graph structure in the current real-time scene tends to be more
localized and fragmented [10-11]. Therefore the algorithm will destroy the original community when adding or deleting users.

(2). Secondly, even the clustering algorithm with linear time complexity can hardly guarantee the real-time performance of partitioning. Executing algorithms on large-scale data multiple times is very expensive. For example, in social networks, the increase of friends in local circles often happens.

In order to overcome the shortcomings of the current algorithm, we propose to use Fast Unfolding algorithm to initialize the clustering of users, and use the clustering results as the basis for incremental partitioning to propose an incremental partitioning algorithm for user clustering.

2. Related work

2.1 Existing user clustering algorithm

In complex big data networks, the user clustering algorithm clusters users in the network according to the similarity of users, so that users with similar characteristics are classified into one category. In general, the time complexity of finding the exact solution of this kind of partitioning method is a problem that the non-deterministic turing machine can solve in exponential time. At present, the algorithms available for clustering belong to optimization algorithms or heuristic algorithms, such as the classic K-means algorithm [12], which solves the clustering center by calculating the distance between the sample point and the randomly selected centroid point and by continuous iteration. The process of clustering users until the centroid no longer changes. The disadvantage of this method is that the user's initial clustering center is selected by a random method, so that the clustering result converges to the local minimum [13]. In order to solve the problem of local convergence, In [14], it is proposed to apply simulated annealing genetic algorithm to user clustering as a heuristic search technique, and select the initial clustering center to obtain the global optimal solution. The algorithm obviously improves the quality of clustering, but the time complexity is high, making it difficult for the algorithm to adapt to the current big data background.

2.2 Community Detection Method Based on Fast Unfolding Algorithm

Fast Unfolding algorithm is a classical community detection algorithm. It is an optimization algorithm based on modularity. The algorithm is implemented in two stages. In the first stage, each node is considered as a community, so the number of nodes is the same as the number of communities in the initialization stage. For each node $I$ in the network diagram, find the neighbor node $J$ of $I$, and then calculate the increment of module degree after the $I$ node merges from its original community to the community where the $J$ node is located. Then select the community where the $J$ node produces the largest increment of modularity, and merge the community where the $I$ node and the $J$ node are located, but only if the increment is positive. Only then will $I$ community merge, when the incremental negative value, $I$ node will remain in the original community. This process is repeatedly applied to all nodes until the first stage is completed when the movement of any node does not cause the module degree increment to be positive. Module degree increment calculation formula is as follows:

$$\delta Q = \left[ \frac{1}{2m} \sum_{i} \left( \frac{2k_{i,in}}{2m} + \left( \frac{P_{tot} + k_{i}}{2m} \right)^2 \right) \right] + \left[ \frac{1}{2m} \sum_{i} \left( \frac{P_{tot}^2}{2m} \right) - \left( \frac{k_{i}}{2m} \right)^2 \right]$$

Where $k_{i,in}$ is the number of edges inside the community, $k_{1;in}$ is the sum of the weights of the edges from node $I$ to the nodes inside the community $C$, $k_{tot}$ is the sum of the weights of all the edges connected to the nodes inside the community $C$, $k_{i}$ is the weighted sum of edges connected to the $I$ node, $m$ is the weighted sum of all edges in the network. Some nodes may be considered multiple times during the calculation process.

In the second phase, the user class obtained by combining the first phase is regarded as a node, and the module degree increment is calculated again. Therefore, the running time of the algorithm is mainly in the first stage. It then shows a geometric drop as the hierarchy accumulates. The time
complexity of the algorithm is $O(kdn)$, $n$ is the number of nodes, $d$ is the number of edges in the network, and $k$ is the hierarchy of graphs.

The Fast Unfolding algorithm implements community detection according to the user's network topology and the weights of edges. In this paper, we use this algorithm for the initial division of similar groups preferred by users.

3. Incremental clustering method for users with similar preferences

According to the analysis of social networks and e-commerce user networks in real life, there are mainly two kinds of dynamic change data that affect the clustering results. One is the dynamic change of user interest. Second, real-time changes in the number of users. The change of user interest affects the change of the inner edge of the community, and the change of the number of users affects the change of nodes in the community. The incremental expansion algorithm proposed in this paper consists of two parts. It is the incremental algorithm when the node changes and the incremental algorithm when the edge changes.

### 3.1 Incremental algorithm when nodes change

When some nodes are added to the network, there may be two situations, one kind of nodes have similar preferences with the nodes in the initial community set $C$, and the other kind don’t exist. Therefore, batch incremental nodes cannot be divided incrementally as a single incremental node. Firstly, there may not be a similar preference to the node preferences in the initial community $C$. Secondly, a single node calculates modularity increments in turn with preference classes in $C$. If the increments are negative, many single nodes will be in a single community. Based on this, we can first perform Fast Unfolding algorithm on the newly added batch nodes to preliminarily classify user preference similar classes, and then calculate module increment with the user preference classes in the initial community set $C$. If the modularity increment after merging with a preference class in $C$ is positive, the preference class with the largest increment is selected to merge with it, otherwise it is separately classified as a preference class.

---

**Algorithm 1** Incremental Expansion Algorithm for Batch Node Addition

**Input:** $C$ community set, NewNodes added node’s set, Edges added edge’s set with weight value  
**Output:** NEWC new community set

1: function Userscluster($C$, NewNodes, Edges)
2: PartC ← Fast Unfolding(NewNodes, Edges)
3: /* Perform Fast Unfolding algorithm on the newly added node’s set to divide the community, and PartC is the newly divided community set*/
4: for PartC1→PartCp do
5: for C1→Cn do
6: $\Delta Q_i$←$merge$(PartCi, Cn)
7: $\Delta maxQ$←$max$($\Delta Q_i$)
8: if $\Delta maxQ$≥0 then
9:     merge(PartCi, Cj)
/* Merge the new community with each original community, calculate the module increment of each new community after merging, and select the original community that has the largest module increment with the new community*/
10: else
11:     Cn+1←PartCi
12: end if
13: end for
14: end function
3.2 Incremental algorithm for edge change

The incremental algorithm when the edge changes is still based on the initial division result of Fast Unfolding algorithm. When the edge between communities decreases and the edge within communities increases, the modularity within communities increases, that is, the original community structure becomes closer and does not affect the division of communities. Just recalculate the modularity of the community. The increase of the edge between the communities and the reduction of the edge between the communities will make the original community structure more loose, so this paper proposes an algorithm to redefine the belonging community of the relevant nodes.

Using Fast Unfolding algorithm can present the complete hierarchical community structure of the network, thus obtaining communities with different resolutions. Therefore, changes in the edges will inevitably affect the community structure at each level. For example, The network generates a community set of two-tier structures after being calculated by Fast Unfolding algorithm. Community set of the first layer structure C1={ {A,B}, {C,D}, {E} } and community set of the second layer structure C2={ {A,B}, {C,D,E} }, suppose you add an edge eAC between communities and delete an edge eDE within the community. The change of the edge is eDE in the first-level community structure, which is the edge between communities. Deleting does not affect the division of the first-level community structure, but only calculates the module increment after eAC is divided into the communities where node C is located. If it is positive, the community structure at the first level will be changed to C1 = { {B}, {A,C,D}, {E} }, and the community structure at the second level will be calculated using Fast Unfolding algorithm.

Algorithm 2 Incremental Expansion of Edges Change between Communities

**Input:** C community collection at each level, AddEdges collection of new edges, DeleteEdges collection of deleted edges

**Output:** NEWC1 new first level community structure set

1. function Userscluster(C, AddEdges, DeleteEdges)
2. If (edges ⊆ AddEdges && edges ⊆ DeleteEdges)
3. delete edges from AddEdges and DeleteEdges
4. AddNewc1← changeAddEdges(C1, AddEdges) // First tier community structure after adding edges , C1 the first level communities
5. for DeleteEdges do
6. If DEij(Nodei and Nodej in same community) then // DEij ⊆ DeleteEdges
7. New C1← changeDelEdges(AddNewc1, DeleteEdges)
8. else
9. NEWC1 ← AddNewc1
10. end for
11. function changeAddEdges(C1, AddEdges)
12. for AddEdges do
13. ΔQ← AEij(Nodei add in Nodej in community)// AEij ⊆ AddEdges
14. If ΔmaxQ≥0 then
15. merge(Nodei, Nodej in community)
16. end if
17. end for
18. function changeDelEdges(C1, DeleteEdges) // C1 is the first level community structure within C
19. ΔmaxQ← merge(Nodei, neiher node in community) //Calculate the maximum modularity increment after nodei merges with the community where the neighbor node is located
4. Experimental analysis

4.1 Dataset Description
In order to verify the accuracy and efficiency of the incremental user clustering method proposed in this paper, this paper uses one-year transaction data from British online retailers(http://archive.ics.uci.edu/ml/datasets/Online+Retail) as experimental data. Online Retail DataSet is the transaction data of an online store based in the UK from December 1, 2010 to December 9, 2011. This data set has 3779 users 541909 order records in total.

4.2 Experimental process
In order to make the experimental results more accurate, we first use Fast Unfolding algorithm to cluster the data set D to get the clustering results. Then, a part of data is extracted from the data set D as incremental data ΔD, and the remaining data D-ΔD is clustered using Fast Unfolding algorithm. Then, the ΔD and D-ΔD are clustered using the incremental algorithm proposed in this paper. we compare the results of the first global partition with the results of the second first data extraction and then merging partition. So as to verify the accuracy of the algorithm. At the same time, the running time of static incremental partition and dynamic incremental partition are compared to prove the efficiency of the incremental algorithm. To be intuitive, we will use the LFR benchmark to generate the baseline network as the newly inserted nodes and edges of the Online Retail network.

4.3 Experimental results
The experimental data set was initially clustered by Fast Unfolding algorithm to form a three-tier community structure, with the maximum modularity of the three-tier community structure being 0.62.

4.3.1 Nodes Change. Tables 1 and 2 record the results obtained by static insertion and incremental insertion when adding a single node and 350 edges and adding 200 nodes and 58600 edges in the above data set. Tables 1 and 2 record the results obtained by static insertion and incremental insertion when adding a single node and 350 edges and adding 200 nodes and 58600 edges in the above data set.

4.3.2 Edges change. Table 3 and Table 4 respectively record the division results obtained by the incremental algorithm when adding edges between communities and deleting edges within communities in the Online Retail dataset. In the table, you can see the comparison with the static algorithm in terms of module degree and time spent.
| Table 1. Modularity comparison |
|-------------------------------|
| **Nodes/links**               | **Total data** | **Insert 200** | **Insert 400** |
|                               | 3779/1M        | 3579/94.7k     | 3379/93.5k     |
| **Initial data**              | 0.62           | 0.62           | 0.62           |
| **Static insert**             | -              | 0.61           | 0.608          |
| **Increment insert**          | -              | 0.59           | 0.579          |

| Table 2. Time comparison |
|--------------------------|
| **Nodes/links**          | **Total data** | **Insert 200** | **Insert 400** |
|                          | 3779/1M        | 3779/1M        | 3779/1M        |
| **Initial data**         | 52.63ms        | 52.63ms        | 52.63ms        |
| **Static insert**        | -              | 55.63ms        | 59.32ms        |
| **Increment insert**     | -              | 13.12ms        | 20.73ms        |

| Table 3. Increase edges test results |
|---------------------------------------|
| **AddEdges** | **IncrementT** | **StaticT** | **IncrementQ** | **StaticQ** |
| 100          | 12.93 ms       | 54.24 ms    | 0.613          | 0.62        |
| 200          | 13.35 ms       | 54.37 ms    | 0.597          | 0.62        |
| 300          | 13.79 ms       | 54.52 ms    | 0.604          | 0.62        |
| 400          | 14.26 ms       | 54.61 ms    | 0.615          | 0.62        |
| 500          | 14.57 ms       | 54.78 ms    | 0.583          | 0.62        |

Although the modularity of the running result of the incremental algorithm is slightly less than that of the static algorithm, the error can be guaranteed to be within 4% and the time is reduced by at least one time compared with the static algorithm. Therefore, the incremental algorithm in this paper can ensure that the incremental nodes can be clustered in a faster time without destroying the original community structure.

| Table 4. Delete edges test results |
|------------------------------------|
| **DeleteEdges** | **IncrementT** | **StaticT** | **IncrementQ** | **StaticQ** |
| 100          | 10.98 ms       | 52.36 ms    | 0.617          | 0.62        |
| 200          | 11.29 ms       | 52.57 ms    | 0.593          | 0.62        |
| 300          | 11.43 ms       | 52.82 ms    | 0.614          | 0.62        |
| 400          | 11.53 ms       | 53.34 ms    | 0.608          | 0.62        |
| 500          | 11.74 ms       | 53.57 ms    | 0.585          | 0.62        |

5. Conclusion
This paper mainly solves the problem of user clustering with similar preferences under the background of big data. The experiments show that the proposed method is effective. It can be used to deal with the problem of community ownership caused by frequent increase of users.
In the future, we will study how to reduce the similarity of preferences between the new users and the old users in the process of incremental learning under the premise of accurately determining the belonging community.

Acknowledgment
Foundation items: Project supported by the Hebei Social Science Foundation (No. HB18GL046).

References
[1] Abel F, Gao Q, Houben G J, et al. Analyzing User Modeling on Twitter for Personalized News Recommendations[M]//User Modeling, Adaption and Personalization. Springer Berlin Heidelberg, 2011:1-12.
[2] O’Callaghan D, Prucha N, Greene D, et al. Online social media in the Syria conflict: Encompassing the extremes and the in-betweens[C]//IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. IEEE, 2014:409-416.
[3] Sheridan G, Sheridan G, Smyth B. Aggregating content and network information to curate twitter user lists[C]//ACM Recsys Workshop on Recommender Systems and the Social Web. ACM, 2012:29-36.
[4] Sharma N K, Ghosh S, Benevenuto F, et al. Inferring who-is-who in the Twitter social network[J]. Acm Sigcomm Computer Communication Review, 2012, 42(4):533-538.
[5] Abell F, Gao Q, Houben G J, et al. Analyzing User Modeling on Twitter for Personalized News Recommendations[J]. Lecture Notes in Computer Science, 2011, 6787:1-12.
[6] Pinney J W, Westhead D R. Betweenness-based decomposition methods for social and biological networks[C]// 2006:87--90.
[7] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics, 2008, 2008(10):155-168.
[8] Huang C L, Yeh P H, Lin C W, et al. Utilizing user tag-based interests in recommender systems for social resource sharing websites[J]. Knowledge-Based Systems, 2014, 56(C):86-96.
[9] Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. Nature, 2005, 435(7043):814.
[10] Bu Z, Zhang C, Xia Z, et al. A Fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network[J]. Knowledge-Based Systems, 2013, 50(3):246-259.
[11] Duch J, Arenas A. Community Detection in Complex Networks Using Extremal Optimization[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2005, 72(2):027104.
[12] Kanungo T, Mount D M, Netanyahu N S, et al. An efficient k-means clustering algorithm: analysis and implementation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2002, 24(7):881-892.
[13] Yan Xiangbin, Li Yijun, Ye Qiang. Research on customer classification based on purchase behavior [J]. Computer integrated manufacturing system, 2005,11(12):1769-1774.