GHIC: A Hierarchical Pattern Based Clustering Algorithm for Grouping Web Transactions

Yinghui Yang
Graduate School of Management
University of California, Davis
yiyang@ucdavis.edu

Balaji Padmanabhan
The Wharton School
University of Pennsylvania
balaji@wharton.upenn.edu

Abstract

Grouping customer transactions into categories helps understand customers better. The marketing literature has concentrated on identifying important segmentation variables (e.g., customer loyalty) and on using cluster analysis and mixture models for segmentation. The data mining literature has provided various clustering algorithms for segmentation without focusing specifically on clustering customer transactions. Building on the notion that observable customer transactions are generated by latent behavioral traits, in this paper we investigate using a pattern-based clustering approach to grouping customer transactions. We define an objective function that we maximize in order to achieve a good clustering of customer transactions and present a new algorithm, GHIC, that groups customer transactions such that itemsets generated from each cluster, while similar to each other, are different from ones generated from others. We present experimental results from user-centric Web usage data that demonstrates that GHIC generates a highly effective clustering of transactions.

1. Introduction

Distance-based clustering techniques, such as *k*-means, and parametric mixture models, such as gaussian mixture models, are two main approaches used in customer segmentation in the data mining literature. While distance-based clustering is convenient, it is not obvious why it is the appropriate method for grouping customers. For mixture models, using changing model parameters to represent the difference between segments can often oversimplify the differences and can ignore variables and patterns that are not captured by the parametric models. In this paper we study a new approach to segmenting customer transactions: one that is based on the idea that there may exist natural behavioral patterns in different groups of transactions. For example, a set of three behavioral patterns in a group of wireless subscribers may be: (1) their call duration during weekday mornings is short, and these calls are within the same area, (2) they call from outside the home area on weekdays and from the home area on weekends, and (3) they have several "data" calls on weekdays. Indeed this set of patterns may be common behavioral patterns of consultants who travel frequently. More generally the example suggests that there may be natural clusters in data, characterized by a set of behavioral patterns. In such cases, "pattern-based clustering" (PBC) approaches can be effective in grouping customer transactions. The main idea is to cluster customer transactions such that patterns generated from each cluster, while similar to each other within the cluster, are very different from the patterns generated from other clusters. In the above example rules are a good representation for patterns. However in a different domain, such as time series data on stock prices, representations for patterns may be based on "shapes" in the time series. It is easy to see that traditional distance-based clustering techniques and mixture models are not well suited to learning clusters for which the fundamental characterization is a set of patterns such as the ones above.

A reason that PBC can generate natural clusters from customer transactions is that such transactions often have natural categories that are not directly observable from the data, but these categories often result in patterns that are observed. For example, Web transactions may be for work, for entertainment, shopping for gifts etc. While the categories are not usually known, transactions at work, for instance, may be quicker and more focused, while transactions for entertainment may be long and across a broader set of sites. Hence, grouping transactions such that the patterns generated from each cluster are "very different" from those generated from another cluster may be an effective method for learning the natural categorizations. In this paper we study the use of PBC for grouping Web transactions. Based on using itemsets to represent patterns in Web transactions we present *GHIC* (Greedy Hierarchical Itemset-based Clustering), a PBC algorithm for domains in which itemsets are the natural representation.

2. Pattern-Based Clustering of Web Transactions

We start with session-level Web browsing data for users provided by a market data vendor [3] and create 46 features describing the session. The features include those about time (e.g., average time spent per page), quantity (e.g., number of sites visited), and order of pages visited (e.g., first site) and therefore include both categorical and numeric types. A conjunction of atomic conditions on these attributes (an "itemset") is a good representation for common behavioral patterns in the transaction data. For example, {*starting_time* = *morning*, *average_time_page* < 2 *minutes*, *num_categories* = 3, *total_time* < 10 *minutes*} is a behavioral pattern that may capture a user's specific "morning" pattern of Web usage that involves looking at multiple sites (e.g., work email, news, finance) in a focused manner such that the total time spent is low. Another common pattern for this (same) user may be {*starting_time* = *night*, *most_visted_category* = *games*}, reflecting the user's typical behavior at the end of the day. Hence based on itemsets as the

representation for patterns below we describe an objective function for the PBC method and an algorithm, GHIC for PBC of Web transactions.

2.1 Developing an Objective Function

Consider a collection of transactions to be clustered $\{T_1,T_2,\ldots, T_n\}$. Each transaction T_i contains a subset of a list of candidate items $\{i_1\ i_2 ,\ldots , i_s\}$. A clustering C is a partition $\{C_1 ,C_2\ ,\ldots ,C_k\}$ of $\{T_1\ ,T_2\ ,\ldots,T_n\}$ and each C_i is a cluster. Here we only consider hard clustering where a transaction belongs to exactly one cluster. The goal is to maximize M, defined as follows:

$$M(C_1, C_2,..., C_k) = \textit{Difference}\ (C_1, C_2,..., C_k) + \sum_{i=1}^{k} \textit{Similarity}\ (C_i)$$

Below we define the difference between two clusters (which is sufficient, since we use a hierarchical clustering technique to repeatedly cluster into two groups), where $S(P_a, C_d)$ represents the *support* of pattern P_a in cluster C_d, and P_1 through P_{m} are the set of patterns based on which the difference is computed (we explain this shortly).

$$\textit{Difference}(C_i, C_j) = \sum_{a=1}^{m} \frac{\left|S(P_a, C_i) - S(P_a, C_j)\right|}{\frac{1}{2}\cdot\left[S(P_a, C_i) + S(P_a, C_j)\right]}$$

The intuition behind the definition of difference is that the support of any pattern in one cluster should be different from the support in the other cluster, and we therefore compute the relative difference between the support values and sum the relative differences across all patterns considered. The similarity measure for a cluster is harder to define and there may be several measures that can be developed, and our technique can handle any such measure. A heuristic we use is that, if transactions are more similar to each other, then they can be assumed to share more patterns. Hence, the definition below uses the number of strong patterns as a proxy for the similarity.

Similarity $S(C_i)$ = the number of frequent itemsets in cluster C_i

The definition permits any set of patterns to be used in computing the difference, but which set of patterns to use in practice? Using all patterns (frequent and infrequent) is one option, but often there are a large number of infrequent itemsets which we would like to exclude. A second option is to use only the frequent itemsets across a pair of clusters considered. This would mean running a frequent itemset discovery procedure for every pair of clusters considered, an expensive task. A third option, used in this paper, is to use only the set of patterns that are frequent in the original (unclustered) data. This is certainly a reasonable set of patterns to consider, and from a computational point of view this is efficient since it involves the generation of large itemsets only once (with perhaps a low value for the minimum support).

2.2. The Clustering Algorithm

The search for the best clustering to optimize *M* is clearly exponential in the number of transactions and we therefore present a heuristic technique below. After we obtain *FIS*, the set of frequent itemsets in the unclustered data, we generate a new dataset, BID (Binary Itemsets Dataset), where the rows represent the original transactions and the columns/(binary)features represent the presence or absence of a frequent itemset. We use $T'=\{ T'_1 , T'_2 , \ldots , T'_n \}$ to represent this new set of transactions. We convert the problem into clustering these binary vectors and present a hierarchical clustering algorithm. The transaction set is first divided into two clusters, and each cluster is further divided into two clusters until user-specified stopping conditions are met. In order to generate balanced clusters at each stage, we introduce another component to *M*. Since each division creates two clusters, the revised *M* is specified as follows:

$$M(C_1, C_2) = K_1 \cdot D(C_1, C_2) + K_2 \cdot S(C_1) + K_3 \cdot S(C_2) + f_{BALANCE}(N_1, N_2)$$

K_1 , K_2 , K_3 are user-specified weights and $N_i = |C_i|$. The weights can be used as a scaling factor to bring the difference and similarity measures to comparable values. Two approaches to defining $f_{BALANCE}(N_1, N_2)$ are:

$$f_{BALANCE}(N_1, N_2) = -K_4 \cdot |N_1 - N_2| \qquad \text{(A)}$$

$$f_{BALANCE}(N_1, N_2) = \begin{cases} 0, & K_l \le \dfrac{N_1}{N_2} \le K_h \\ -\infty, & \dfrac{N_1}{N_2} > K_h \quad or \quad \dfrac{N_1}{N_2} < K_l \end{cases} \qquad \text{(B)}$$

Inputs: C_0 = **{** T'_1 , T'_2 , … , T'_n **}**
$FIS = \{is_1 , is_2 , \dots , is_p \}$
Output: Clusters $C = \{ C_1 , C_2 , \dots , C_f \}$

Repeat {
Choose any $X = \{T_1 , T_2 , \dots , Ts \}$ from C such that the stopping condition is not satisfied for X ;
For every is_r in FIS {
Let $C_a^{(r)} = \{T_k \mid T_k \in X \text{ and } is_r \subseteq T_k \}$;
Let $C_b^{(r)} = \{T_k \mid T_k \in X \text{ and } is_r \not\subset T_k \}$;
Calculate $M_r = M(C_a^{(r)}, C_b^{(r)})$
}
$r^* = \arg\max_r (M_r)$;
$C = (C - X) \cup C_a^{(r^*)} \cup C_b^{(r^*)}$;
} While the stopping condition is not satisfied for every *X* in *C*

Figure 1 *Algorithm GHIC*

Algorithm *GHIC* is presented in Figure 1. The main heuristic used in the hierarchical algorithm is as follows. Rather than considering the "best" single partition at each node in the tree – a procedure that will involve considering $2^{|D|}$ pssible partitions (where D is the data to be divided at a specific level of the tree) – we use a greedy heuristic that considers $|FIS|$ possible partitions based on partitioning the data according to the presence/absence of each frequent itemset. At each stage among these partitions we choose the one for which the global objective M is

maximized. This procedure is motivated by a reasonable assumption that while there are many different patterns that may differentiate two clusters, there is often one dominating pattern that is present mostly in one of the clusters. As we show in the experiments below, based on the quality of the discovered clusters this assumption appears to be a reasonable one.

3. Experiments: Clustering user-centric Web transactions

For evaluating clustering algorithms two measures of cluster "goodness" or quality are often used [7]. One type of measure is an *internal quality measure* that can be used to compare different approaches. The internal quality measure that we use is the average *M* measure (defined above) across all pairs of clusters generated by GHIC. The second type of measure is an *external quality measure* that compares clusters with known classes. One popular external measure is entropy, which we use as follows. To test our method we constructed 80 sub-datasets from the set of all user transactions, where each sub-dataset contains transactions corresponding to a fixed number of users. Specifically among these 80 datasets, 10 datasets contain sessions from 2, 3, 4, 5, 10, 20, 50, 100 users respectively. Since in any sub-dataset we know exactly which transactions belong to which user, we construct the entropy measure as follows. Let $n_h^{(f)}$ denote the number of transactions belonging to customer f in cluster h, n_h denote the number of transactions in cluster h, N denote the number of clusters generated, and C_U denote the number of users (classes) in the sub-dataset. The entropy measurement (external measure) used to measure the quality of the clustering is as follows:

$$E = -\frac{1}{N} \cdot \sum_{h=1}^{N} \sum_{f=1}^{C_U} \frac{n_h^{(f)}}{n_h} \cdot \log_{C_U} \left(\frac{n_h^{(f)}}{n_h} \right)$$

In order to compare the performance of *GHIC*, we implemented three methods: (1) Hierarchical *k*-means on data represented in items (the "*k*-means-items" approach), (2) Hierarchical *k*-means

on data represented in itemsets (the "*k*-means-itemsets" approach), and (3) *GHIC* on data represented in itemsets. As described in Section 2, we create 46 features to describe the user's behavior for each session. We create an item for each value of the categorical features, categorize the continuous variables according to their value distribution (uniformly binned), and create an item for each bin. After this itemization, we create a session (transaction)-by-item matrix with binary values for each sub-dataset on which "*k*-means-items" is run. From this matrix, we use *Apriori* to discover frequent itemsets and create a session (transaction)-by-itemsets matrix (as described in Section 2) on which "*k*-means-itemsets" and GHIC are applied. The comparison between *k*-means-items and *k*-means-itemsets evaluates whether the itemset representation is good, since the two approaches share the same clustering algorithm but are applied on essentially what is different data formats. The comparison between *k*-means-itemsets and *GHIC* evaluates the specific heuristic used in GHIC since the two use the same data format but different algorithms. For *GHIC* we set the values of K_1 , K_2 and K_3 to bring the first three components of *M* to relatively compatible scales. If the size of a smaller cluster is at least 10% of the larger cluster, the value of $f_{BALANCE}(N_1, N_2)$ is 0, and otherwise $-\infty$. The minimum support we set for *Apriori* to discover frequent itemsets is 5% since the purpose is to learn all itemsets that may be useful in computing differences between clusters, and having a high value for this could result in not learning some small, but important set of transactions, that constitute a behavioral cluster. The support threshold that we used to define large itemsets for calculating similarity within a cluster is set to be 20%, a higher value since here we are interested in how similar transactions are within a single cluster and having many patterns with a high support value intuitively captures the idea of transactions being similar to each other. Finally, when generating itemsets we consider only itemsets that contain fewer than 4 items for computational

as well as practical reasons (for segmentation purposes it is hard to make sense of long conjuncts in users' behavioral patterns), and the stopping criterion is that the size of the cluster is smaller than 5% of the size of the dataset to be clustered.

Tables 1 and 2 below presents the results on the 80 datasets.

Table 1 Clustering Result - Entropy values

	2-user datasets	3-user datasets	4-user datasets	5-user datasets	10-user datasets	20-user datasets	50-user datasets	100-user datasets
k-means-item	0.28	0.49	0.57	0.59	0.84	0.88	0.90	0.93
k-means-itemset	0.20	0.45	0.53	0.57	0.77	0.88	0.89	0.91
GHIC	0.17	0.28	0.34	0.41	0.62	0.73	0.80	0.87

Table 2 Clustering Result - M values

	2-user datasets	3-user datasets	4-user datasets	5-user datasets	10-user datasets	20-user datasets	50-user datasets	100-user datasets
k-means-item	1	1	1	1	1	1	1	1
k-means-itemset	1.63	1.38	1.49	1.12	1.22	1.58	1.32	1.09
GHIC	1.75	2.67	2.01	1.87	1.50	1.79	1.84	1.54

Note: In the table above because the absolute value of M is by itself not very meaningful and so we report the relative values by setting the M values of *k*-mean-item to be the baseline 1.

Based on a paired *t*-test across all the sub-datasets, *GHIC* significantly outperformed *k*-means-itemsets ($p = 0.0004$), which in turn outperformed *k*-means-items ($p = 0.004$). The comparison between *k*-means-items and *k*-means-itemsets demonstrates that the itemset pattern representation is indeed good. The comparison between *k*-means-itemsets and *GHIC* shows that our algorithm is more effective in PBC than hierarchical *k*-means on itemsets. On average, *GHIC* outperformed *k*-means-itemsets by 21% and *k*-means-items by 27%. The magnitudes are quite significant and suggest that PBC techniques may be a natural approach to cluster customer transactions such as Web transactions considered in this paper. In addition to the quantitative

results, there are several examples of interesting clusters discovered in the data. For example two most significant patterns in a cluster were {*starting_day = Saturday, most_visited_category = sports*} and {*starting_day = Sunday, most_visited_category = services such as chat rooms*}, perhaps indicating weekend behavioral patterns for some users. For another cluster two most significant patterns were {*total_time = long, most_visited_category = entertainment, time = evening*} and {*average_time_per_site = long, most_visited_category = games, time = evenin*g}, perhaps indicating entertainment patterns for some users. There were indeed more complex clusters. For instance two significant itemsets in one of the clusters were: (1) {*portal_category = yes, first_site = iwon.com*} with support 100% - in contrast, the support for this itemset over the entire dataset was just 9.5%; (2) {*last_category = portal, average_site_per_category = 1* } with support 66.1% - again, in contrast the support over the entire dataset was just 5.7%. In general, the itemsets in some of the clusters suggest highly explainable behavior patterns (as in the first two examples above), while in other cases it is more difficult to place a single behavior type (the third example) based on the itemsets in the cluster. This may be in part due to the fact that consumer behavior in the real world is highly complex. In such cases, the clusters generated can still be useful by providing a starting point for deeper understanding of consumer behavior.

Rather than just describing each cluster based on the patterns, we can also compare two clusters and identify the differences between them. In order to do this systematically we picked every partition generated at some level in the tree during clustering and identified the top ten patterns based on support within each cluster and also the top ten patterns that contribute largest to the difference metric. Such analyses can provide additional insights as Table 3 shows. For instance cluster 1 in Table 3 mainly involves entertainment-related activities while cluster 2 has mixed activities involving multiple categories. Majority of the patterns that differentiate these

two clusters the most are the strong patterns from cluster 1, although there are also strong patterns from cluster 2 (e.g. Sports = yes) that contribute to the differences.

Table 3 The Entertainment Effect

	Top Patterns
Cluster 1	Entertainment = yes (Support: 100%) Most_visited_cat = Entertainment (Support: 100%) Most_visited_cat = Entertainment , Entertainment = yes (Support: 100%) Portals = yes (Support: 82%) Entertainment = yes , Portals = yes (Support: 82%) Most_visited_cat = Entertainment , Portals = yes (Support: 82%) Most_visited_cat = Entertainment , Entertainment = yes , Portals = yes (Support: 83%) Services = yes (Support: 64%) Entertainment = yes , Services = yes (Support: 64%) Most_visited_cat = Entertainment , Services = yes (Support: 64%) Most_visited_cat = Entertainment , Entertainment = yes , Services = yes (Support: 64%)
Cluster 2	Portals = yes (Support: 83%) Services = yes (Support: 77%) Services = yes , Portals = yes (Support: 68%) Sports = yes (Support: 60%) Sports = yes , Portals = yes (Support: 58%) Services = yes , Sports = yes (Support: 54%) Services = yes , Sports = yes , Portals = yes (Support: 53%) Corporate = yes (Support: 51%) Services = yes , Corporate = yes (Support: 45%) Portals = yes , Corporate = yes (Support: 45%)
Most different Patterns	Most_visited_cat = Entertainment , Entertainment = yes (Support Difference: 100%) Most_visited_cat = Entertainment , (Support Difference: 100%) Most_visited_cat = Entertainment , Portals = yes (Support Difference: 82%) Most_visited_cat = Entertainment , Entertainment = yes , Portals = yes (Support Difference: 82%) Entertainment = yes (Support Difference: 71%) Most_visited_cat = Entertainment , Services = yes (Support Difference: 64%) Most_visited_cat = Entertainment , Entertainment = yes , Services = yes (Support Difference: 64%) Sports = yes (Support Difference: 57%) Most_visited_cat = Entertainment , Last_cat = Entertainment (Support Difference: 57%) Most_visited_cat = Entertainment , Last_cat= Entertainment, Entertainment = yes (Support Difference: 57%) Entertainment = yes , Portals = yes (Support Difference: 57%)

Note: If there are more than 10 patterns, it means that the ones after the 10^{th} have the same support value as the 10^{th}. Services, Portals, Sports, Corporate and Entertainment are different Web site categories. Services category includes Web sites such as emails, E-cards, Coupons, Chat rooms, etc. Most_visited_cat stands for the most visited category in a Web browsing session. Last_cat stands for the last category in a Web browsing session. Services = yes means that a Web browsing session contains Web sites from the Services category.

4. Related Work

The approach in [4] considers two objects to be similar if they exhibit a coherent pattern on a subset of dimensions. The definition of a pattern is based on the correlation between attributes of objects to be clustered, while this paper uses a itemsets as the pattern representation. Hence,

within the clustering methods dealing with categorical data, those using large items/itemsets and association rules are more closely related to this paper. In [1] frequent itemsets are used to construct a weighted hypergraph and a hypergraph partitioning algorithm from [2] is used to partition the items. The result is a clustering of items (not transactions) that occur together. Suggesting that there should be many large items within a cluster and little overlapping of such items across clusters [5] introduces a clustering criterion, and notes that for transactions that come naturally in collection of items it is more meaningful to use item/rule-based methods. A similar problem is addressed in [6] and our work is most similar to LargeItem [5] and CLOPE [6] in that we do not use any pairwise distance function. We define a global goal and use this goal to guide the clustering process. Compared to LargeItem [5] and CLOPE [6] our method takes a new perspective of associating itemsets with behavior patterns and using that concept to guide the clustering process. Notably we allow a *set of itemsets* to describe a cluster instead of just a set of items, which is the focus of previous work.

5. Conclusions

As argued in this paper the existence of natural categories of customer behavior is intuitive, and these categories influence the transactions/data observed. We suggest that pattern-based clustering techniques, such as the one described in this paper, may be effective in learning such natural categories and can enable firms to understand their customers better and build more accurate customer models. In this paper we presented a new approach, *GHIC*, for pattern-based clustering of Web transactions and demonstrated that the technique performs effectively, compared to traditional techniques. A notable strength of our approach is the ability to explain the clusters and the differences between clusters and our experiments demonstrate this to be the case. There are several opportunities for future work based on these ideas. Additional

difference/similarity metrics and global objective functions could be examined and there may be opportunities to develop better heuristics for such functions. Further new representations for patterns could result in the development of further novel PBC techniques.

References

[1] E. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering based on association rule hypergraphs," Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97) (in cooperation with ACM SIGMOD'97), May 1997.

[2] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain," Proc. of the ACM/IEEE Design Automation Conference, 1997.

[3] S. Kimbrough, B. Padmanabhan, and Z. Zheng, "On Usage Metric for Determining Authoritative Sites," In Procs. of WITS 2000.

[4] H. Wang, J. Yang, W. Wang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," Proc. ACM SIGMOD Conference, pp. 394-405, June 2002.

[5] K. Wang, C. Xu, and B. Liu, "Clustering Transactions Using Large Items," Procs. of ACM CIKM'99, pp. 483-490, November, 1999.

[6] Y. Yang, X. Guan, and J. You, "CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data," In Procs. of KDD 2002, pp. 682-687, July 2002.

[7] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," In KDD Workshop on Text Mining, 2000.