HACS: Heuristic Algorithm for Clustering Subsets

Ding Yuan[*] W. Nick Street[*]

Abstract

The term *consideration set* is used in marketing to refer to the set of items a customer thought about purchasing before making a choice. While consideration sets are not directly observable, finding common ones is useful for market segmentation and choice prediction. We approach the problem of inducing common consideration sets as a clustering problem on the space of possible item subsets. Our algorithm combines ideas from binary clustering and itemset mining, and differs from other clustering methods by reflecting the inherent structure of subset clusters. Experiments on both real and simulated datasets show that our algorithm clusters effectively and efficiently even for sparse datasets. In addition, a novel evaluation method is developed to compare clusters found by our algorithm with known ones.

1 Introduction

Recent marketing literature [8, 1] has shown that a consumer may be aware of many brands but consider only a few for purchase. Consumer decision making can be described as a sequence of stages during which the number of brands decreases. Among the *universal set*, i.e., all products that could be purchased, the subset that a given consumer is aware of is called the *awareness set*. The smaller set that the consumer would consider purchasing is called the *consideration set*. Finally, the *choice set* is the subset of products that the customer is known to have purchased. Knowledge of common consideration sets could provide valuable insight into the market structure of different brands in a category, and may help predict choice by narrowing the number of possible outcomes. However, consideration sets cannot be directly observed from scanner data, since we (fortunately) record only what consumers purchase, not what they think. To our knowledge, no method has been proposed to identify the actual consideration sets of a group of customers based on scanner data. Further, since the number of possible consideration sets goes up exponentially with the number of products, the problem is unrealistic to analyze with conventional methods for even moderate numbers of products. Therefore we attempt to induce the set of common consideration sets by clustering the observed sets of choices.

The objective of cluster analysis [5] is to partition unlabeled data into groups, such that the similarity of data points within groups is maximized and the similarity among different groups is minimized. Specialized algorithms and distance measures have been developed for data with categorical [3, 7] and binary [6] features. In particular, binary clustering can be used to analyze market scanner datasets, which use binary variables to indicate whether the products have been purchased by the customers. For example, customers can be clustered together based on similarity in their purchase behaviors. Clustering algorithms typically represent clusters by some sort of center, and base similarity measurements between the data points and the cluster on that center. However, for domains with special structure such as subsets, this objective is not appropriate.

Our work is also informed by itemset mining [9], which is often used to analyze binary datasets to find frequent itemsets. We are given a set of items $I = \{i_1, i_2, ..., i_m\}$ and a database of transactions $T = \{t_1, t_2, ...t_n\}$, where each t_i is a transaction that contains a set of items from I. A set $x \subseteq I$ is called an itemset. The *support* of an itemset x is the number of transactions in which that itemset occurs as a subset. An itemset is $frequent$ iff its support is greater than or equal to some predefined threshold.

A customer's consideration set contains all products she has purchased, possibly together with some which she has not. Using traditional clustering methods to group customers with similar consideration sets may assign some customers to a cluster that contains fewer products than they have actually purchased. Further, the intuition behind what makes a consideration set 'interesting' can be specialized beyond the typical clustering objectives of 'dense' and 'well-separated.' Therefore, traditional clustering methods are not ideal for consideration set analysis. Similarly, the interesting itemsets needed for consideration set analysis may not themselves be frequent, according to the standard definition from frequent itemset mining. For example, a relatively small group of customers who buy exclusively within a set of three products might still be considered a relevant market segment. We propose HACS (Heuristic Algorithm for Clustering Subsets) to build clusters out of itemsets, in such a way that the clusters are both common and interesting. We also introduce a new method to compare the clusters found by our algorithm to known ones generated with a data simulator.

[*]Management Sciences Department, University of Iowa

2 HACS

The problem of extracting an interesting collection of consideration sets is difficult to express formally, say, as a constrained optimization problem. This, together with the complexity of the problem (there are 2^{2^n} possible combinations of subsets of n items), leads us to approach consideration set detection as a heuristic search problem. Our method is carried out in two steps. The first step is to order the candidate cluster centers based on a *quality* measurement. The second step is to build clusters that satisfy a *quantity* criterion. The resulting clusters represent the induced consideration sets. We also propose two efficiency methods to reduce the number of candidate clusters examined.

2.1 Order candidate cluster centers The challenge of identifying consideration sets is that they are not directly observable. We know only the choice set, the products purchased by a customer over a finite time window, which may be fewer than the products considered. To induce consideration sets using purchase history, we proceed from three basic assumptions. First, purchased products are included in one's consideration set. Second, one's consideration set may include some products that have not been purchased. This could be due to, say, a consumer waiting for a promotion, short purchase history, etc. Any finite data collection is inevitably an incomplete picture of the purchasing habits of at least some customers. Finally, the unpurchased products in one's consideration set can be inferred from the purchase records of other consumers with similar purchase histories. If a substantial group of customers bought products A, B and C (and rarely anything else), we consider it likely that a new customer who has purchased only items A and B may have considered C.

We define a candidate consideration set as one with the following desirable property: most people who bought items from the set, bought *only* items from the set. To find such sets, we count the number of customers who purchase exclusively within each itemset x, and the number who purchase a strict superset of x. These two counts are compared to estimates of their expectations.

We begin with a set of m items $I = \{i_1, i_2, ..., i_m\}$ and a database of transactions $T = \{t_1, t_2, ...t_l\}$, with each transaction containing one or more chosen items, as in itemset mining. The transactions for each customer are gathered together to create a database $D = \{c_1, c_2, ...c_n\}$, in which each record c_j is a choice set, containing exactly the set of items in I purchased by customer j during the sampling time. An initial scan of this dataset constructs a partial subset lattice (Figure 1) by creating a node for each unique itemset x that appears as the choice set for some customer.

We define individual support for an itemset x, is_x, as the number of customers who purchased exactly the set of items x. We further define partial support, ps_x, as the number of times x appears as a strict subset of a customer's choice set. We note that $ps_x + is_x$ is equivalent to $support_x$ in itemset mining. The number of times x is a superset of a customer's choice set is called total support, ts_x. Thus, $ps_x = \sum_{y \supset x} is_y$, and $ts_x = \sum_{y \subseteq x} is_y$. In Figure 1, is_x is shown in parentheses for each itemset x. Each node can be viewed in the shape of an hourglass, with subsets above and supersets below. For a dataset that contains four products {A, B, C, D} and 250 customers, $ps_{AB} = is_{ABC} + is_{ABD} + is_{ABCD} = 7 + 10 + 4 = 21$, and $ts_{AB} = is_A + is_B + is_{AB} = 35 + 16 + 39 = 90$.

The candidate consideration sets should be those with a total support that is larger than expected, i.e., there are more than the expected number of customers buying only from this set of products. These customers buy either all or some of the products in this set. Further, the candidates should have partial support that is smaller than expected, i.e., there are fewer than the expected number of customers buying the set of products plus others from outside the set.

We now compute expected support values. The *expected* ps_x $E(ps_x)$ and *expected* ts_x $E(ts_x)$ are estimated using the is_y, for all singleton sets $y \in x$. Given N customers, the probability that a single item i appears in a customer's purchase history is estimated as $p_i = (is_i + ps_i)/N$. Assuming that the products are independent, the support expectations are:

$$E(is_x) = N \prod_{i \in x} p_i \prod_{j \notin x} (1 - p_j),$$
$$E(ps_x) = N \prod_{i \in x} p_i - E(is_x),$$
$$E(ts_x) = N \prod_{j \notin x} (1 - p_j).$$

In our example, the expected values are calculated using $p_i, i \in \{A, B, C, D\}$ as follows: $E(is_{AB}) = N(p_A)(p_B)(1 - p_C)(1 - p_D)$, $E(ps_{AB}) = N(p_A)(p_B) - E(is_{AB})$, and $E(ts_{AB}) = N(1 - p_C)(1 - p_D)$.

Each node in the lattice is considered a candidate cluster center. A good center should follow two primary criteria. First, the set and its subsets appear more

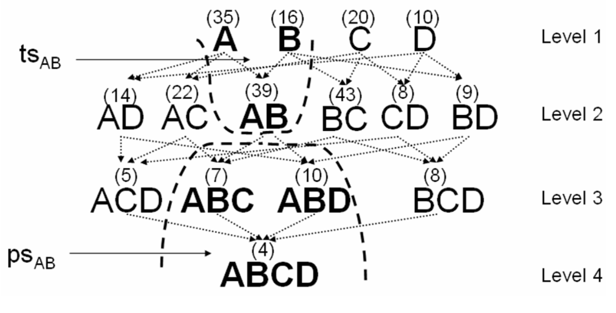

Figure 1: Statistics for sample subset lattice

often than statistical expectation, i.e., the $ts\_ratio = ts_x/E(ts_x)$ is large. Second, its strict supersets appear less often than expected, i.e., the $ps\_ratio = ps_x/E(ps_x)$ is small. The ratio of real over expected points is used because it is independent of where in the lattice the node appears. If the raw numbers of ts and ps were used, the nodes at the bottom would almost always be preferred, since relatively few customers have large choice sets. We introduce an overall measure of quality for a subset node, $quality_x = ts\_ratio - ps\_ratio$. The nodes with higher quality values are those with larger-than-expected subsets (the top of the hourglass), and smaller-than-expected supersets (the bottom). For example, if {A,B} ranked highest among itemsets in the second level of the lattice, then there are more than the expected number of customers who bought exclusively within product set {A,B}, and fewer than expected who bought other products together with {A,B}. Therefore {A,B} fits the pattern of a consideration set.

2.2 Cluster construction Once the quality measures are computed, we start from the top level of the subset lattice, and order nodes within each level based on their quality values. After ordering the candidate centers, clusters are built to identify consideration sets. A quantity threshold, QT, is used to make sure the clusters are constructed only if they contain a sufficiently large percentage of data points. For each candidate x, if $ts_x \geq QT$, x is marked as a confirmed consideration set, and the support of the subsets of x are allocated to x. Otherwise, the support of x is available for allocation to one of its supersets. Proper selection of the quantity threshold is important because it controls the size of the constructed clusters. The threshold is selected experimentally by choosing the range that gives good cluster evaluation values on simulated data.

For a dataset with many items, the described algorithm is slow because there are too many candidate cluster centers. To increase efficiency, we choose only the candidates with high quality values compared to their peers on the same level of the lattice structure. All nodes on the first level are set to be candidates. Only the top 50% of the nodes on the second level are candidates, and then top 10% of the nodes on lower levels are considered. This *top-n* candidate selection method improves efficiency significantly, especially for sparse datasets with large numbers of items. Another method of increasing efficiency is to set a quality threshold, which acts as a lower bound on the quality for sets to be considered. This is used for dense datasets and shows around 50% run-time reduction. Figure 2 outlines the algorithm.

```
Input:  Choice set database D, Quantity threshold (QT)
Output:  Clusters

//Compute individual support
for  each choice set record $c \in D$
  $is_c$ ++
end

//Compute other node statistics
for  each itemset $x$
  $ts_x = \sum_{y \subseteq x} is_y$
  $ps_x = \sum_{y \supset x} is_y$
  $E(is_x) = N \prod_{i \in x} p_i \prod_{j \notin x} (1 - p_j)$
  $E(ts_x) = N \prod_{j \notin x} (1 - p_j)$
  $E(ps_x) = N \prod_{i \in x} p_i - E(is_x)$
  $quality_x = ts_x/E(ts_x) - ps_x/E(ps_x)$
end

//Sort nodes within each level
for  $l = 1$ to $|I|$
  perform search reduction, if necessary
  sort itemsets in level l on quality, descending
end

//Build clusters
for  $l = 1$ to $|I|$
  for  each candidate $x$ at level $l$
    if  $ts_x > QT$
      confirm $x$ as a consideration set
      update $ts_y, \forall y \supset x$
    end
  end
end
```

Figure 2: HACS algorithm outline

2.3 Discussion HACS differs from related algorithms in several ways. It handles itemsets differently than itemset mining, as interesting candidate consideration sets are not necessarily frequent. Rather, they are the itemsets for which a significant number of customers bought from the set, and bought *only* from the set. We introduced the ratios of real values with expected values as a consistent measure with which to order the candidate clusters. The structure of the lattice efficiently stores the information, as we construct nodes only for those choice sets that appear in the dataset. The memory required is therefore linear in the number of distinct itemsets in the transaction database. While this number is theoretically exponential in the number of items, in practice we find that the number of actual choice sets is much smaller. For example, in a real dataset with 27 items described later, only 3273 of more than 10^8 possible choice sets appear. HACS is a specialized type of agglomerative hierarchical clustering that merges only itemsets that have subset-superset relationships, rather than using similarity as the only criteria. Clusters constructed by other hierarchical clustering methods can be visualized as a tree of clusters, where our lattice structure allow each node to have multiple parents.

3 Evaluation and Experiments

We tested HACS on both simulated data and real marketing datasets. We built a simulator to generate data closely resembling real customers' purchase history. We generate lifelike customer records with specific consideration sets, along with some noise, and evaluate our algorithm on its ability to reconstruct these known sets. If our clustering method can reliably reconstruct the true consideration sets, then the clusters constructed from real data will be trustworthy. This is a common approach in the evaluation of clustering methods, as there is no external, objective measure of performance on real data sets without known class labels.

Our simulation assumes that each customer has a consideration set, buys items within that set with nonzero probabilities, and is observed for some number of purchases. We add noise to the data by allowing rare purchases outside the consideration set (say, because of a mother-in-law visit). The simulator generates the consideration sets and their corresponding prior probabilities with respect to customers, and using these, creates a purchase history for each customer. Parameter values are empirically chosen based on the distributions of two true data sets to create a realistic distribution of choice sets of different sizes.

3.1 Evaluation We want our algorithm to find existing clusters, and not generate clusters that are not in the data. Standard measures such as precision and recall are inadequate for measuring cluster quality here, since some misses are closer than others. For example, for known clusters {1234} and {4567}, both groups of found clusters {123}, {456}, {567} and {13}, {45}, {47} have precision and recall equal to zero, but the first group intuitively matches the known clusters better. We thus use a set similarity measurement, the Jaccard coefficient [4], to evaluate found clusters. Given two subsets A and B, the match score $match_{AB}$ is defined as $\frac{|A \cap B|}{|A \cup B|}$. $match_{AB} = 1$ if A and B are identical, 0 if they have no items in common, and is otherwise between 0 and 1.

We developed a heuristic method to compare sets of subsets by matching the most similar clusters between the true and the found sets, and computing match scores for the connected pairs. Given two groups of clusters $\mathcal{A}$ and $\mathcal{B}$ that may have different sizes (assume $|\mathcal{A}| > |\mathcal{B}|$), we want every cluster to be matched with at least one cluster in the other group. The overall objective is to maximize $\sum match_{AB}$, where A $\in \mathcal{A}$, B $\in \mathcal{B}$, and A and B are the assigned matches. The overall matching score of two groups of clusters is $\sum match_{AB}/|\mathcal{A}|$, a measure between 0 and 1, with 1 indicating a perfect match.

We first find the best matching cluster for every cluster A $\in \mathcal{A}$, so the relationship between clusters in $\mathcal{A}$ and $\mathcal{B}$ is n:1, where $n \geq 1$. Next, we find matches for any unmatched clusters in $\mathcal{B}$ by re-arranging the matchings. The clusters in $\mathcal{A}$ that have 1:1 (therefore, ideal) matches are marked as unavailable. Each unmatched cluster in $\mathcal{B}$ finds its best match among the available clusters in $\mathcal{A}$. The unmatched cluster with the least score change is matched first, and so on, until all clusters are matched.

3.2 Simulation We simulated two real-world datasets. One is a dense dataset with 8 items, 145 distinct choice sets, and 6513 customer records. The other is a sparse dataset with 27 items, 3273 choice sets and 5978 customers. Top-n search was applied to the sparse data, resulting in run-time reduction around two orders of magnitude, and a 3-4% reduction in match score. The quantity threshold was set to reflect the actual cluster size, 0.05-0.11 for dense data and 0.01-0.11 for sparse data.

Figure 3 shows the match score comparison averaged over 10 simulated datasets for dense data with 1% noise. The match scores are stable around 0.9. There is a large and significant difference between the clusters found by HACS compared to frequent itemsets, which were generated using the same quantity threshold as used to find the consideration sets.

HACS also performs significantly better than k-means (Figure 4). Since the user does not directly control the number of clusters in our method, match scores are recorded depending on the number of found clusters, and the averaged score is compared with k-means. For k-means, distance along a single dimension is 0 if the two attribute values are the same, and 1 otherwise. As the number of clusters increases, the match score for our algorithm goes down slightly.

The match score comparison for sparse datasets is shown in Figure 5. The overall match score for the sparse data is lower than for dense data, but still shows significant difference with the score for the

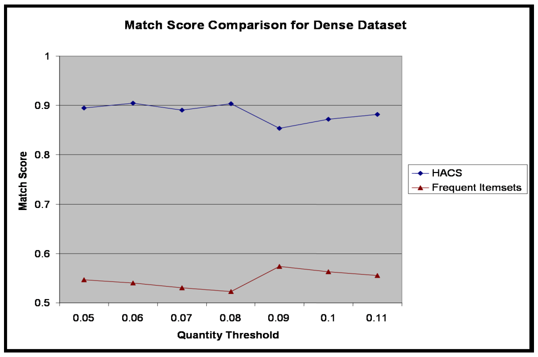

Figure 3: Dense data: HACS vs. frequent itemsets

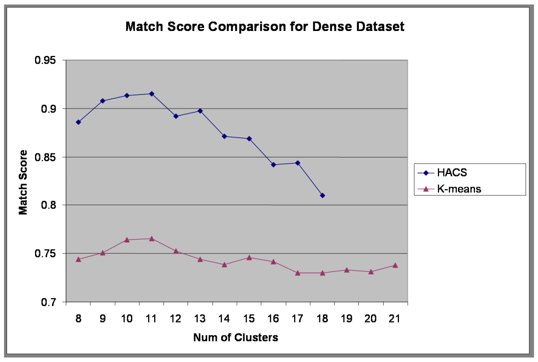

Figure 4: Dense data: HACS vs. k-means

frequent itemsets. The highest match score is reached when the quantity threshold is much smaller than that for the dense dataset. This allows the method to find interesting but infrequent consideration sets with relatively large numbers of items. In such a sparse environment, frequent itemset analysis performs poorly.

For sparse datasets, our algorithm performs about the same as k-means when the number of clusters is small (Figure 6). HACS performs significantly better than k-means for numbers of clusters above 24, which is the true number of consideration sets. In general, k-means tends to find clusters that contain small numbers of items, where the clusters are "dense" in the common definition of clusters. The clusters found by HACS are spread more evenly across the levels of the lattice structure, which makes our algorithm suitable to find structures in sparse datasets.

We tested robustness to noise by allowing customers to buy outside their consideration sets with 1- 5% chances. Our algorithm degrades slowly in the presence of noise, losing around 1-1.5% of match score for every added 1% of noise. For example, in dense datasets with 5% noise, a quantity threshold of 0.08 resulted in a match score of 0.84, compared to 0.91 with no noise. HACS continues to outperform k-means and frequent

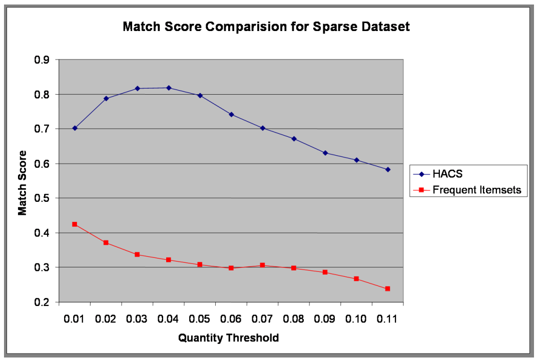

Figure 5: Sparse data: HACS vs. frequent itemsets

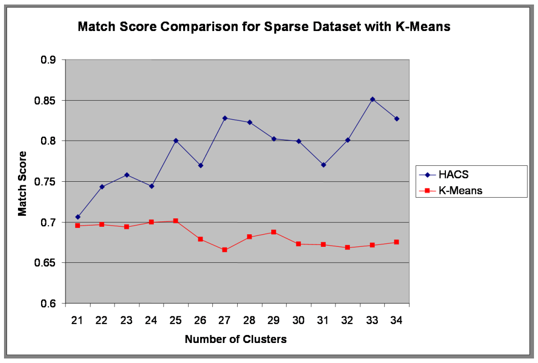

Figure 6: Sparse data: HACS vs. k-means

itemsets regardless of noise level.

In general, HACS performs extremely well at extracting known consideration sets from simulated data, dominating both k-means and frequent itemsets. K-means performs reasonably well, but its objective searches for clusters with the wrong shape and does not enforce the hierarchical relationship needed for subset clustering. Itemset mining fails because it finds only the frequent itemsets, making it likely to find redundant sets near the top of the lattice. Its support threshold (similar to our partial support measure) eliminates interesting but less frequent subsets with more items.

3.3 Real datasets Since true consideration sets are unknown for real datasets, no match scores are shown. The first dataset contains purchase history on 8 brands of ketchup. We added a quality threshold to the best search parameters found in the simulation to reduce the number of clusters, making the results more understandable. Since there are few products, computational efficiency is not a concern, as tests ran in a few seconds. Table 1 shows extracted consideration sets with a quality threshold of -0.5 and a quantity threshold of 7%. Only half of the customers consider at least 4 brands of the 8. Heinz is the leading brand, favored across all customers. Its mid-sized products, 28oz and 32oz, appear in all consideration sets. People who buy smaller Heinz products may also consider other brands, but those who consider large Heinz are typically loyal to the brand.

We also tested HACS on scanner data with 27 brand/sizes of non-liquid detergent. Here, products represent brand/size pairs, e.g., all small containers of Tide go together as one product. In Table 2, s = small size, m = medium, and l = large. Both top-n selection and a quality threshold were used to reduce the number of candidates. Seven clusters are found with a quantity threshold of 5% and a quality threshold of 3. Tide is a dominant brand that exists in most consideration sets; about 10% of customers consider only Tide. Still, the found consideration sets are surprisingly large and

Table 1: Found consideration sets example: Dense data

	Consideration Sets	Customers
1	{B, C}	100
2	{B, C, D}	93
3	{B, C, E}	54
4	{A, B, C, D}	66
5	{A, B, C, F}	52
6	{A, B, C, E, G}	42
7	{A, B, C, D, E, F, G, H}	327

A: Heinz 14oz	B: Heinz 28oz	C: Heinz 32oz
D: Heinz 44oz	E: Hunts	F: Delmonte
G: Other 28oz-	H: Other 32oz+	

Table 2: Found consideration sets example: Sparse data

	Consideration Sets	Customers
1	{25, 26}	528
2	{2, 6, 7, 10}	313
3	{2, 10, 11, 18, 23, 25, 26}	359
4	{4, 5, 10, 17, 18, 24, 26}	285
5	{1, 7, 10, 14, 16, 18, 25, 26}	360
6	{2, 3, 19, 20, 21, 22, 23, 25, 26}	327
7	{0-26}	3806

0: ALL_m	1: ALL_l	2: ARM_l
3: $B-3_s$	4: $B-3_m$	5: $B-3_l$
6: CH_m	7: CH_l	8: $DASH_m$
9: $DASH_l$	10: $DREFT_m$	11: $F1S_s$
12: FAB_m	13: $FRSH_s$	14: $FRSH_m$
15: $GAIN_m$	16: $OXYDOL_s$	17: $OXYDOL_m$
18: $OXYDOL_l$	19: $PUREX_m$	20: $PUREX_l$
21: $SURF_s$	22: $SURF_m$	23: $SURF_l$
24: TD_s	25: TD_m	26: TD_l

varied, with few overlaps. This is because detergent brand/sizes are not substitutes; many households buy multiple items for different laundry tasks. Small-sized items may serve as substitutes, as sets 1-6 contain one small product on average. Purex and Surf bands occur almost exclusively together, with set 6 containing all offerings of both brands.

HACS includes a default cluster that contains all items and represents all subsets that were not allocated to some other cluster. The number of customers in the default clusters is somewhat surprising, as we find that many customers buy from a relatively large set of items. However, few of these large sets represent many customers, resulting in a disproportionate number of customers falling through to the default cluster. If our quality threshold were relaxed, this number would fall, and a more complete market segmentation could be achieved; however, the resulting clusters would be of relatively poor quality. As always with unsupervised learning, result quality depends largely on the desires of the user, and the "true" number of clusters (hence, the "right" values for our thresholds) hinges on finding actionable results. In our real-data experiments, we focused on finding a small number of clearly-defined consideration sets, not a complete partition of the space.

4 Conclusions and Future Work

We propose a novel heuristic subset clustering algorithm, HACS, inspired by the analysis of consideration sets. Our main contribution is that the cluster construction considers the special relationship among itemsets. We also developed an evaluation method to reflect how the extracted clusters match known ones. Experiments on simulated data show that HACS effectively identifies known clusters for both dense and sparse datasets. HACS is a general machine learning technique that is applicable to recommender systems and marketing problems such as cross-selling.

HACS could be generalized to categorical clustering by using counts instead of binary variables for purchase history. Further, our market segmentation could be used as a preprocessing step for choice prediction [2], by building a classifier for each each cluster. This could aid accuracy by eliminating some outcome classes for many customers, and help explain the characteristics of customer groups.

References

[1] G. M. Allenby and J. L. Ginter. The effects of instore displays and feature advertising on consideration sets. *International Journal of Research in Marketing*, 12:67–80, 1995.

[2] I. V. Cadez, P. Smyth, E. Ip, and H. Mannila. Predictive profiles for transaction data using finite mixture. Tech. Report No. 01-67, University of California, Irvine.

[3] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *In Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, pages 512–521, March 1999.

[4] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[6] T. Li. A unified view on clustering binary data. *Machine Learning*, 62:199–215, 2006.

[7] M. Peters and M. Zaki. CLICK: Clustering categorical data using k-partite maximal cliques. Technical Report 04-11, Computer Science Dept., RPI, 2004.

[8] J. H. Roberts and J. M. Lattin. Consideration: Review of research and prospects for future insights. *Journal of Marketing Research*, 34:406–410, 1997.

[9] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley, 2006.