2007

Heuristic subset clustering for consideration set analysis

Ding Yuan
University of Iowa

HEURISTIC SUBSET CLUSTERING FOR CONSIDERATION SET ANALYSIS

by

Ding Yuan

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration in the
Graduate College of The
University of Iowa

December 2007

Thesis Supervisor: Associate Professor 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. Our algorithm combines ideas from binary clustering and itemset mining, and differs from other clustering methods by reflecting the inherent structure of subset clusters. Further, we introduce two speed-up methods to make the algorithm more efficient and scalable for large datasets. Experiments on both real and simulated datasets show that our algorithm clusters effectively and efficiently even for sparse datasets. A novel evaluation method is also developed to compare clusters found by our algorithm with known ones.

Based on the clusters found by our algorithm, different classification models are built for each particular consideration set. The advantages of the two-stage model are it builds specific model for different clusters, and it helps us to capture the characteristics of each group of the data by analyzing each model.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

HEURISTIC SUBSET CLUSTERING FOR CONSIDERATION SET ANALYSIS

by

Ding Yuan

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration in the
Graduate College of The
University of Iowa

December 2007

Thesis Supervisor: Associate Professor Nick Street

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Ding Yuan

has been approved by the Examining Committee
for the thesis requirement for the Doctor of
Philosophy degree in Business Administration at
the December 2007 graduation.

Thesis Committee: _____
Nick Street, Thesis Supervisor

_____
Gary Russell

_____
Warren Boe

_____
Padmini Srinivasan

_____
Hwanjo Yu

To my husband, Di, our children, Jerry and Jack for their love and support.

ACKNOWLEDGMENTS

I wish to express sincere appreciation to Dr. Nick Street who not only served as my supervisor but also encouraged and challenged me throughout my academic program. In addition, special thanks are due to Dr. Gary Russell, whose enlightened mentoring were instrumental and inspiring. I could not say enough thanks to them for this work to be done. I learn so much from them.

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. Our algorithm combines ideas from binary clustering and itemset mining, and differs from other clustering methods by reflecting the inherent structure of subset clusters. Further, we introduce two speed-up methods to make the algorithm more efficient and scalable for large datasets. Experiments on both real and simulated datasets show that our algorithm clusters effectively and efficiently even for sparse datasets. A novel evaluation method is also developed to compare clusters found by our algorithm with known ones.

Based on the clusters found by our algorithm, different classification models are built for each particular consideration set. The advantages of the two-stage model are it builds specific model for different clusters, and it helps us to capture the characteristics of each group of the data by analyzing each model.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

With the data explosion in recent decades, more and more data are available in various fields and need to be analyzed. However, discovering the useful knowledge has been very challenging, because traditional methods can't handle the large amounts of data. Data mining, which is an integral part in Knowledge Discovery in Databases (KDD), is the process of searching for hidden patterns in a group of data [98, 31, 46]. Data mining is a technology that builds novel algorithm based on traditional methods in fields such as computer science, statistics, and pattern recognition.

The process of knowledge discovery to automatically converting raw data into useful information includes a series of steps [90] that consist of: data preprocessing, data mining and post-processing. In data preprocessing, data will be transformed into appropriate forms by performing one or combinations of following tasks: data cleaning, feature selection, dimension reduction, normalization. Data post processing part includes pattern visualization, pattern evaluation and knowledge presentation and interpretation. The most widely used data mining algorithms include classification, cluster analysis and association rule mining.

1.1 Classification

Classification is the process of building a predictive model with data that have known class labels, and then that model can be used to predict the label of new examples whose class is unknown. Classification methods are used for supervised learning, when the class label for the data points are provided. Classification finds a model that maps each attribute set to one of the predefined class labels. Thus,

the main objective of a classifier is to fit the training data well and to correctly predict the unknown class labels of the testing data. Some well-known classifiers include logistic regression [45], naive Bayes [56], support vector machines (SVMs) [91], decision trees [18, 81], and artificial neural networks [84].

The choice of classification methods depends on both the performance of the classifiers and the objective of the analysis. Classification methods can be compared and evaluated according to several criteria.

1. Predictive accuracy, which is defined as the number of correct predictions divided by the total number of predictions. It reflects the ability of the model to correctly predict the unknown class labels.

2. Lift chart, where lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model. It graphically represents the detection rate of the target variable value proportional to the number of processed cases based on the order defined by the predictive model.

3. ROC (Receiver Operating Characteristics) curve analysis, which has similar shape as lift chart, provides a way to select optimal models and discard sub-optimal ones without regard to class distribution or error cost.

4. Speed, which reflects the computation costs of building and using the model.

5. Robustness, which is the ability of the model to make correct predictions given noisy data or missing data values.

6. Scalability, which refers the ability of the model to handle large datasets.

7. Interpretability, which reflects the level of understanding and insight of the data can be provided by the model.

1.2 Clustering Analysis

Compared to classification, cluster analysis is focused on unsupervised learning, with both class label and the number of classes unknown. The objective of clustering is to partition unlabeled data into groups, such that the similarity of data points within groups is maximized, while the similarity among different groups is minimized. A cluster is a group of data objects that are similar to each other within the same cluster and are dissimilar to the objects in other clusters. Cluster analysis can be used alone to gain insight into the structure of the data, and to capture the characteristics of individual clusters. On the other hand, it can be used as a preprocessing method for other algorithms, such as classification. It is a widely-used data mining technique that can be applied to many disciplines and fields. In health care, cluster analysis can capture the spatial pattern of a certain disease by identifying dense and sparse disease occurrence regions. In text mining [88, 100], it can help us to group unlabeled documents into different categories automatically. For example, articles from science new groups can be grouped into categories such as electronics, medical, space, geography, and history. In business [78], clustering analysis has been applied in credit card portfolio management to identify the patterns of credit card defaulters.

Clustering methods [60, 31] have been studied extensively in recent decades. Classic clustering algorithms include: partitioning clustering algorithms such as K-Means [70], and CLARANS [72]; hierarchical clustering such as AGNES (AGglomerative NESting) [60], CURE [43], and ROCK [42]; density-based clustering

such as DBSCAN [33], and OPTICS [10]; model-based clustering such as AutoClass [22], COBWEB [34], SOM [62], and Expectation Maximization (EM) [63], which is a general purpose optimization algorithm that is often used to find the optimal mixture of Gaussian models that represents the dataset.

Normally, traditional clustering algorithms use distance measures to represent the dissimilarities between objects. Such algorithms tend to find clusters that are either a mixture of Gaussian distribution or well-shaped dense areas that are separated by sparse areas. Although this measurement works fine with numerical attributes, it may not be appropriate for datasets with categorical or binary features. For example, in a medicine dataset that uses categorical attributes to represent different vendors, traditional distance is not a good measurement for how different the vendors are.

1.2.1 Categorical/Binary Clustering

Recently, more researchers have developed clustering algorithms [42, 79] specifically for categorical datasets. Categorical clustering algorithm construct clusters for categorical datasets. Binary clustering, which builds clusters on binary data is a subset of categorical clustering. It has caught researchers' attention recently, because it has a wide range of applications.

For example, 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. Products that are frequently purchased together by customers forms clusters that can be used to define the marketing structure of the products. On the other hand, according to similarity in the pattern of purchase behavior, customers can be clustered together to define the market segmentation. Empirical

binary survey data sets are also frequently used in the field of market segmentation, because yes-no questions are simpler and faster to answer for respondents. Binary question format also allows questionnaire designers to pose more questions, because each single question is less tiring.

Binary clustering can be applied to other fields as well, such as binary survey data analysis , document clustering [97], and bioinformatics [32, 37]. For example, by analyzing microarray data, genes that share similar gene expression can be clustered together. It helps to better understand the structure of the relationship between genes and different experimental conditions. Also, the characteristics of poorly-known or novel genes can be inferred from the well-known genes that are clustered together with them.

1.3 Itemset/Frequent-pattern Mining

Association rule mining is another commonly used data mining algorithm. Instead of building models to predict or grouping data points to understand data structure, association rule mining [4, 85, 40] aims at finding interesting association or correlation relationships among items in a given dataset. Itemset mining, sometimes called frequent-pattern mining, is the very first step in mining associations. Itemset mining 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 $frequent$ if and only if its support is greater than or equal to some predefined threshold. According to this definition, all the subsets of a frequent itemset are also

frequent. The number of subsets is exponential in the number of items, the major challenge of itemset mining is efficiency. Many itemset mining algorithms [85, 40] are variants of Apriori [4], which needs to generate a huge number of candidate sets. FP-tree [47] avoids the problem by using an extended prefix-tree structure for storing compressed, crucial information about frequent patterns.

1.4 Consideration Set Analysis

To our knowledge, no existing data mining methods can handle the special hierarchical relationships between itemsets that is needed to analyze consideration sets. A consideration set [83, 19, 6] is the set of brands (a subset of all the brands in the product category) for which a consumer makes an explicit utility comparison before he makes his brand choice decision. For example, among all the brands of ketchup, only a few brands may be considered by a particular customer for a certain purchase. Knowledge of common consideration sets could provide valuable insight into the market structure among different brands in a category, and may help predict choice by narrowing the number of possible outcomes. The analysis of consideration sets is attracting increasing academic and managerial attention. However, the identification and analysis of consideration sets is challenging because they are not directly observable. In the past, deterministic [82] and probabilistic [8, 58] marketing models have been constructed in order to investigate the properties of consideration sets and how they affect the consumers' purchasing behavior. So far, no method has been proposed to identify the actual consideration sets for a certain group of customers. Since the number of possible consideration sets goes up exponentially with the number of elements, making the problem unrealistic to analyze with conventional methods for even moderate numbers of products. Data

mining methods could be potentially applicable to this problem.

1.5 The Need for a New Method

Given that we can only observe customers' choices, we want to infer their consideration sets, and find the group of customers that share the same consideration set. We define such a process as a clustering method to find the clusters customers' consideration sets.

If traditional clustering methods are used to build clusters by grouping those customers who share the same consideration sets, some customers may be grouped to a cluster that contains fewer products than they have actually purchased. So, such clusters poorly represent consumers' actual consideration sets. For consideration set analysis, one of the restrictions of the cluster construction is that an itemset can only be clustered to one or more of its supersets. 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 appropriate to handle datasets 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.

Therefore, in this dissertation we describe our approach of combining ideas from both binary clustering and itemset mining to build clusters in order to find the special structure of consideration sets. Based on that, we also build different

predictive models for each found clusters. This enables us to analyze the characteristics of the individual cluster. Better understanding and interpretation of the data can be achieved at the same time.

The development of the novel clustering algorithm and the empirical analysis are the first contribution of this dissertation. The second contribution is the design of a novel method to evaluate the clusters found by our algorithm compared with known ones, as generated with a data simulator. The third contribution is to use different predictive models to analyze the dataset, instead of using only one model to fit the whole dataset.

CHAPTER 2
LITERATURE REVIEW

Our clustering algorithm uses ideas from binary clustering and itemset mining to form groups of subsets in a manner appropriate for consideration set analysis. We also implement the idea of "divide and conquer" to build different predictive models for each found cluster. This section describes prior work in each of these areas.

2.1 Clustering Algorithms

Clustering algorithms [7, 49, 53, 60, 31] have been studied extensively in recent decades. Good clustering reviews can be found in [54, 16]. Current algorithms can be broadly categorized into four groups: partitioning methods, hierarchical methods, density-based methods, and model-based methods.

Partitioning clustering [60, 63] constructs k partitions of the data, given a database of n objects, where $k \leq n$. The partition should follow two requirements. (1) each group should at least contain one object, and (2) each object must belong to only one cluster. The goal of the algorithm is to minimize some measure of dissimilarity among data points within each cluster, and to maximize the dissimilarity of different clusters. K-means [70] is a typical partitioning clustering algorithm, which iteratively relocates data points to different clusters to minimize the sum of squared distance of the data points from their nearest cluster centroids as shown in Figure 2.1 (The mean of each cluster is marked by a "+").

Given n objects, k clusters that are represented by cluster centers c_i, we have:

$$Total\ Distance = \sum_{i=1}^{k} \sum_{x \in c_i} dist(c_i, x)^2,$$

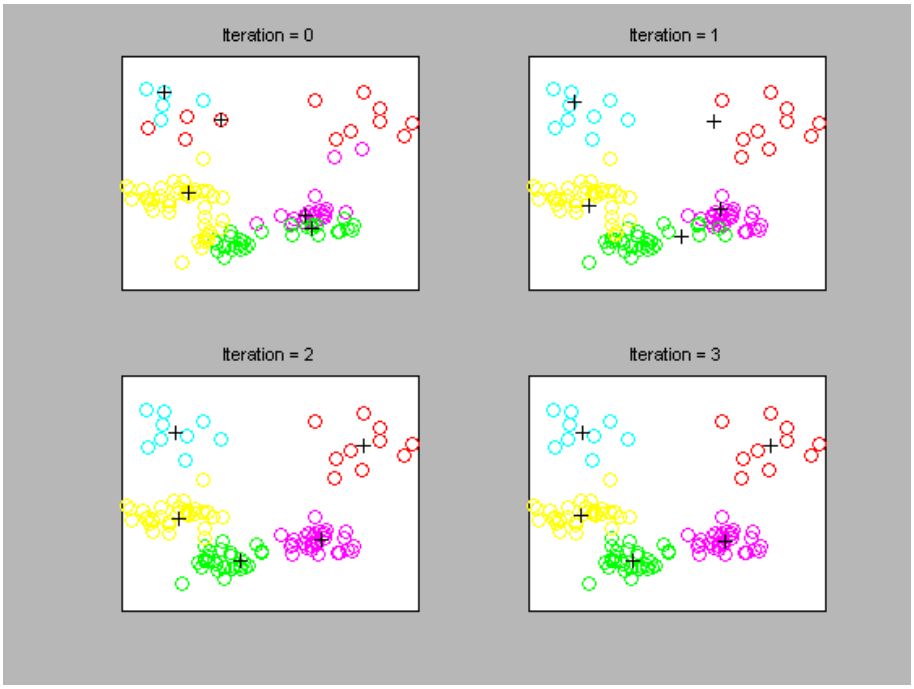

Figure 2.1: Partitioning clustering example

where *dist* is the standard Euclidean distance between two objects in Euclidean space. The centroid c_i of the i^{th} cluster is defined by:

$$c_i = \frac{1}{m_i} \sum_{x \in c_i} x.$$

K-means uses local search, so it always find a locally optimal solution. One drawback of K-means is that the number of clusters, K, must be defined by the user. The clustering result is also very sensitive to the initialization of cluster centers. There have been many improvements to K-means, such as ISODATA [55] and CLARANS [72]. When a threshold on the number of samples in each cluster is

provided, ISODATA automatically determines the number of clusters using a set of rules for splitting and combining existing clusters to obtain a final clustering. CLARANS handles the initialization problem by starting with new randomly selected data points as cluster centers to find a new local optimum. Partitioning clustering builds clusters that satisfy the optimization function. It works well to find find spherical clusters, and it need to be extended to find clusters with complex shapes.

Partitioning clustering algorithms yield satisfactory results for numeric attributes, but they are not appropriate for categorical datasets. For example, consider a market basket dataset that contains 10 products. A transaction of products {1,3,4,5} and {2,7,8,9} could be grouped into the same cluster, represented by the cluster center {3,4,5,7,8,9}, because both of the transactions have substantial items in common with the cluster center. Clusters built this way poorly represent the real structure of the dataset, because these two transactions themselves have nothing in common.

Hierarchical clustering [104, 43, 42, 59] can be implemented by either merging smaller clusters into larger ones, or by splitting larger clusters into smaller ones. The first approach is called agglomerative or bottom-up hierarchical clustering, such as in AGNES [60], CURE [43] and ROCK [42]. It successively merges the objects or groups close to each other, until all of the groups are merged into one, or until a termination condition holds. The second group is the divisive or top-down approach, such as in PDDP [17] and DIANA [60]. It starts with all objects in the same cluster, and then in each successive iteration, a cluster is split up into smaller clusters, until each object is in one cluster, or until a termination condition holds.

The clusters constructed by hierarchical clustering can be visualized by a

tree of clusters called a dendrogram as shown in Figure 2.2, which is appealing because of its ease of interpretation. One disadvantage of most such algorithms is that once a data point is assigned to a cluster, it can't be reassigned to another cluster to improve the cluster quality as in partitioning clustering. Hierarchical clustering methods can be integrated with other clustering techniques for multiple-phase clustering to improve cluster quality, such as in BIRCH [104], CURE [43], and Chameleon [59].

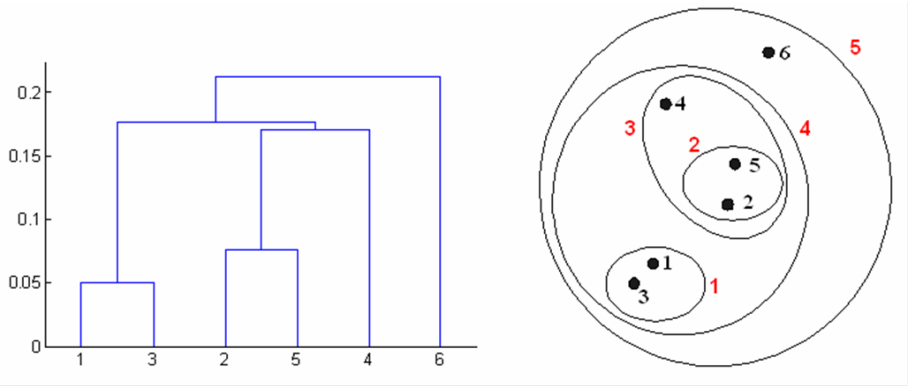

Figure 2.2: Hierarchical clustering example

Another major disadvantage is that it doesn't handle categorical datasets well. For example, consider a market basket dataset that contains four products, and four transactions: {3,4}, {2,3,4}, {1} and {4}. These four transactions can be represented using binary data to indicate if a product is included in a transaction or not. Thus, we have: {0,0,1,1}, {0,1,1,1}, {1,0,0,0}, and {0,1,0,0}. The first two transactions have the smallest Euclidean distance, and then they are merged to

generate centroid {0,0.5, 1,1}. The next smallest distance is between the third and fourth transactions. Thus, those two transactions with nothing in common will be merged together and assigned to the same cluster. The clusters generated are not appropriate.

Density-based clustering [2, 51] defines the density for each point as the number of neighbor points in a radius, then the clusters are defined as dense regions that are separated from one another by low density regions. DBSCAN [33] defines clusters as maximal sets of density-connected points. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise. The problem with DBSCAN is that it is very sensitive to user input parameters: slightly different settings may lead to very different clusterings of the data. OPTICS [10] can be considered as an extension of DBSCAN, and it orders data points so that higher density will be identified first, followed by lower density regions. Therefore, it can identify the hidden structure of clusters. DENCLUE [50] is a clustering method based on a set of density distribution functions. The basic idea of the algorithm is to model the overall point density analytically as the sum of influence functions of the data points. Clusters can then be identified by determining density-attractors and clusters of arbitrary shape can be easily described by a simple equation based on the overall density function.

Density-based clustering algorithms can discover clusters of arbitrary shapes as shown in Figure 2.3, compared to distance-based methods that find only spherical-shaped clusters. These algorithms work well for low-dimensional data of numerical attributes and they are less sensitive to outliers. Density-based clustering algorithms have trouble finding appropriate clusters when clusters have widely varying densities. Also, they don't work well for high-dimensional datasets because density

is hard to define for such datasets.

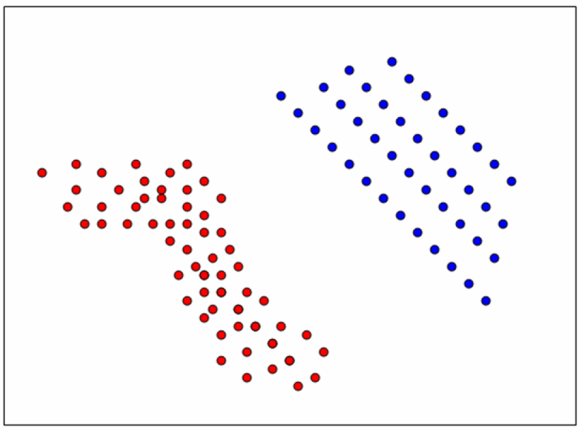

Figure 2.3: Density-based clustering example

Model-based clustering [34, 61] builds clusters by constructing a density function that reflects the spatial distribution of the data points, thus providing the full description of each cluster in terms of the probability distribution of each attribute. Expectation Maximization (EM) [63] is a general purpose optimization algorithm that is often used to find the optimal mixture of Gaussian models that represents the dataset. The first step, calculation of the cluster probabilities is "expectation"; the second, calculation of the distribution parameters, is "maximization" of the likelihood of the distributions given the data. An example of a two-dimensional point set with two clusters of different mean and variance is shown in Figure 2.4. AutoClass [22] uses a Bayesian approach to find the optimal clustering. It covers a broad variety of distributions, such as Bernoulli, Poisson, Gaussian, and log-normal

distributions. SOM (Self-Organizing Map) [62] finds a set of centroids and assigns each object in the data set to the centroid that provides the best approximation of that object. Clustering is performed by having several units compete for the current object. The unit whose weight vector is closest to the current object becomes the winning unit. SOMs assume that there is some topology or ordering among the input objects. One advantage of model-based algorithms is that they can handle data sets with different types of attributes by plugging in different distributions.

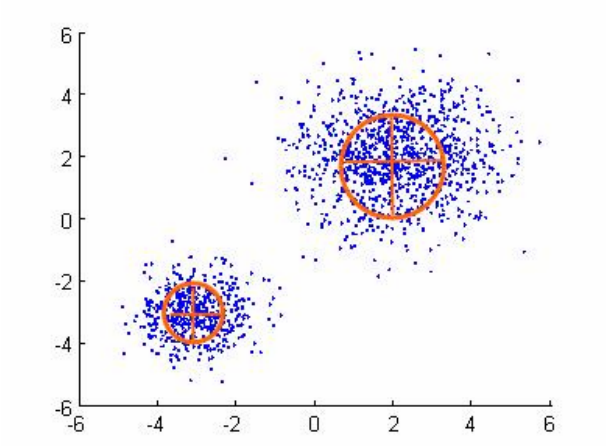

Figure 2.4: Model-based clustering example

2.1.1 Categorical Clustering Methods

Many of the clustering algorithms described above are designed to handle numerical data. Some research [42] shows that traditional clustering algorithms

that use distances between points for clustering are not appropriate for categorical datasets, because the similarity between categorical data is not sufficient for clustering such data. Recently, more researchers developed clustering algorithms [35, 38, 25, 105, 52, 43] specifically for categorical data sets using different methods. ROCK [42] is an agglomerative hierarchical clustering algorithm that uses the number of *links* (shared common neighbors) between two records as the similarity. For a dataset with n data points, the time complexity of the algorithm is $O(|n|^3)$, which makes it unsuitable for large datasets. COOLCAT [11] is an entropy-based algorithm, which explores the connection between clustering and entropy: clusters of similar points have lower entropy than those of dissimilar ones. LIMBO [9] is a scalable hierarchical categorical clustering algorithm that builds on the $Information\ Bottleneck$ (IB) framework for quantifying the relevant information preserved when clustering. They use the IB framework to define distance measures of both categorical tuples and categorical attribute values. STIRR [38] is a non-linear dynamical system that transforms the dataset into a weighted graph and uses an iterative method for assigning and propagating weights to get optimal clusters. The central idea in CACTUS [35] is that a summary of the entire dataset is sufficient to compute a set of *candidate* clusters that can then be validated to determine the actual set of clusters. They generalize the density-based rectangular regions in the numeric domain to categorical domain as the cross product of sets of attribute values. They define clusters to be interval regions that consist of a significantly larger number of tuples than the number expected if all attributes were independent. In addition, they extend a cluster to as large a region as possible. CLICK [79] finds clusters based on a search method for K-partite maximal cliques.

CLICK models a categorical dataset as a K-partite graph where the vertex set (attribute values) is partitioned into K disjoint sets (one per attribute) and edges exist only between vertexes in different partitions.

2.1.2 Binary Clustering Methods

There are some algorithms developed specifically for binary clustering, which is a special case of categorical clustering. For example, K-means was used in [30] as the base cluster method to analyze market segmentation. Based on that, an ensemble clustering method was employed to combine the results, because K-means is an unstable method that may construct different clusters depending on initializations and small changes in the training set. Ordonez et al. [74] introduce a fast clustering algorithm for sparse high-dimensional binary data based on the well-known EM clustering algorithm. They use it to fit a mixture of normal distributions to a sparse binary data set. In other work, Ordonez [73] introduces several improvements to K-means to cluster binary data streams. Both of these methods use distance as the measurement of similarity. Li [64] developed a general model for clustering binary data based on a matrix approximation that reduces the clustering problem to the trace matrix maximization problem which is solved by eigenvalue decomposition. In the same paper, it is proved that in terms of handling binary data, K-means, information-theoretic clustering framework [28], and the block diagonal clustering model [66] can all be represented as different variations of this general model. A unified view of binary data clustering based on the examination of the connections among various clustering criteria is provided in [65]. In particular, they show the relationships among the entropy criterion, dissimilarity coefficients, mixture models, matrix decomposition, and minimum description length.

In general, the key component of these algorithms is a specialized distance measure with which to compare points to each other, and to a cluster center. However, these algorithms build clusters without regard to any special structure of the data being clustered. Specifically in binary clustering, a data point could be assigned to a cluster that is represented by one of its subsets, which would be inappropriate for our application. In terms of consideration sets, a customer might be grouped to a cluster that contains fewer items he has purchased. In cases like this, the above algorithms are insufficient to build and represent the desired clusters. Further, the intuition behind what makes a consideration set "interesting" can be specialized beyond the typical clustering objectives of "dense" and "well-separated". For example, in a dataset that contains some middle-size number of products, say 30, there will be 2^{30} possible combinations of products that could be purchased by a customer. Even if we have thousands of customers, the real choices made by customers will be very sparse in such big data space. Therefore, traditional clustering methods that use "density" is not suitable in this case.

2.2 Itemset/Frequent-pattern Mining

Association rule mining is another commonly used data mining algorithm. First proposed by Agrawal et al. [3], it searches for interesting associations or correlations among a large set of data items. Itemset mining, sometimes called frequent-pattern mining, is the very first step in mining associations.

Itemset mining is often used to analyze binary datasets to find frequent itemsets, and the rules that can be generated from them. 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 $frequent$ (FI) if and only if its support is greater than or equal to some predefined threshold. According to this definition, all the subsets of a frequent itemset are also frequent. Since the number of subsets is exponential in the number of items, the first challenge of itemset mining is efficiency.

Many of the existing itemset mining algorithms [86, 75, 85, 40] are variants of Apriori [4], which employs a bottom-up, breadth-first search that iteratively generates the candidate set of length $(k+1)$ from the set of frequent sets of length k. It is costly to generate a huge number of candidate sets. FP-tree [47] avoids the problem by using an extended prefix-tree structure for storing compressed, crucial information about frequent patterns.

Given the explosive number of frequent itemsets, the second challenge of frequent-pattern mining is how to summarize and interpret the mining results. The introduction of maximal frequent itemsets (MFI) [13, 39, 20] and closed frequent itemsets (CFI) [76, 103] partially handles the problem. A frequent pattern is *maximal* if and only if it does not have a frequent superset. In other words, if an itemset x is frequent and no superset of x is frequent, we say that x is a maximally frequent itemset, and we denote the set of all maximally frequent itemsets by MFI. For example, in Figure 2.5 the itemsets in the lattice are divided into two groups: those that are frequent are located above the border, and those that are infrequent are located below the border. Among the frequent itemsets, {AD}, {ACE}, and {BCEDE} are maximal frequent itemsets, because their immediate supersets are infrequent.

MFI forms the smallest set of itemsets from which all frequent itemsets can be derived. The problem with MFI is while we know the support s for the MFI

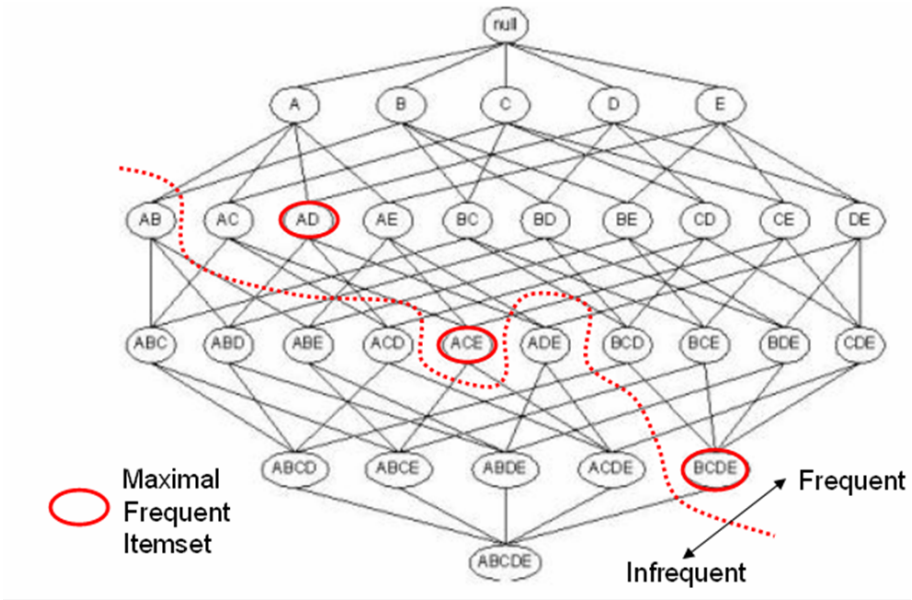

Figure 2.5: Maximal Frequent Itemsets [90]

itself, we know that the support of all its subsets is at least s, but we don't know the exact value. An additional pass over the data set is needed to determine the support counts of the non-maximal frequent itemsets. Therefore, a *closed frequent itemset* (CFI) is proposed to preserve the support information. An itemset x is *closed* if none of its immediate supersets has the same support as the itemset. Put another way, an itemset x is not closed if at least one of its immediate supersets has the same support count as x. An itemset is a closed frequent itemset if it is closed and its support is greater than or equal to minimum threshold. It is defined to be a frequent itemset without a frequent superset with the same support. For example, in Figure 2.6, each itemset is marked with a list of its corresponding transaction IDs.

Itemsets in gray are CFIs. {C} is a CFI, because none of its immediate supersets has a support of four, which is its support. However, {C} is not a MFI, because its supersets, {AC}, {BC}, and {CE} are all FIs. We note that all maximal frequent itemsets are closed because none of the maximal frequent itemsets can have the same support count as their immediate supersets. It is straightforward to see that the following relationship holds: MFI $\subseteq$ CFI $\subseteq$ FI.

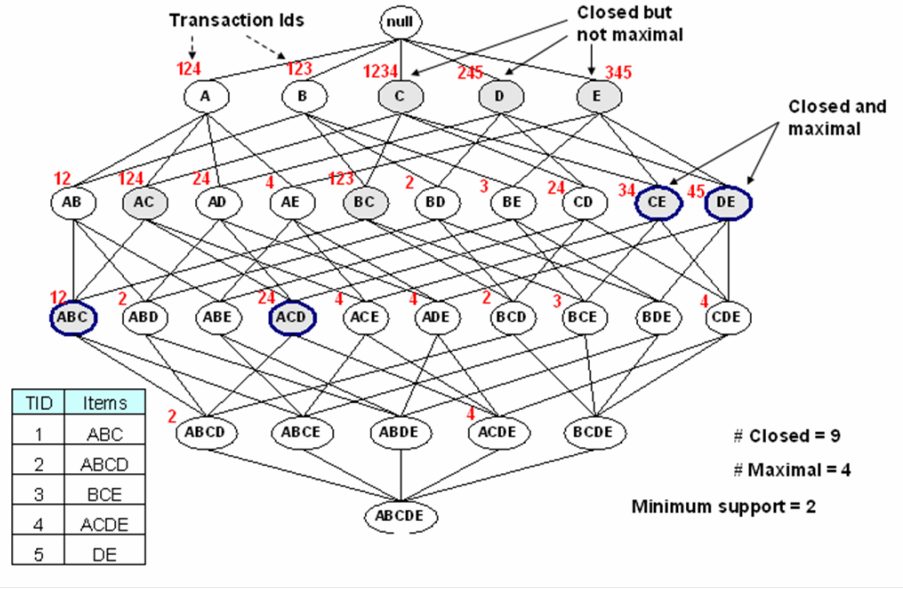

Figure 2.6: Maximal Frequent Itemsets vs. Closed Frequent Itemsets [90]

Other methods to summarize frequent itemsets include: maximal patterns [44], top-k patterns [48], condensed pattern bases [77], approximation-k sets [1], and δ-cluster [99].

Our proposed work is to construct clusters out of the itemsets. 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 that is worthy of a specific marketing strategy, even if their number is relatively small. Also, we note that all the frequent itemsets contain few items, and are pretty much located at the upper levels of the lattice structure, especially when the total number of items increases. For the datasets we analyze, interesting itemsets are spread across all levels. We are interested in finding consideration sets of different sizes that are located in evenly distributed levels, not only the small-sized frequent ones.

2.3 Consideration Sets

In marketing, ample evidence [83, 19, 6, 82, 8] has shown that a consumer may be aware of a large number of brands, but only a few will be considered for purchase on a given occasion. Consumer decision making can be described as a sequence of stages during which the number of brands decreases as shown in Figure 2.7. Among the *universal set*, which refers to all alternatives that could be obtained or purchased by any consumer, the subset that a given consumer is aware of is called the *awareness set*. The awareness set is divided further into brands the consumer would consider purchasing, the subset that is called the *consideration set*, and those that are not considered. Finally, the *choice set* is the subset of products that the customer is known to have actually purchased. For example, one of our datasets records ketchup purchases. A particular customer might consider Heinz, Hunts, and a less expensive regional brand as possible choices (their consideration set),

but in observing the customer over a specific period of time, we might only observe purchases of Heinz and Hunts (the choice set).

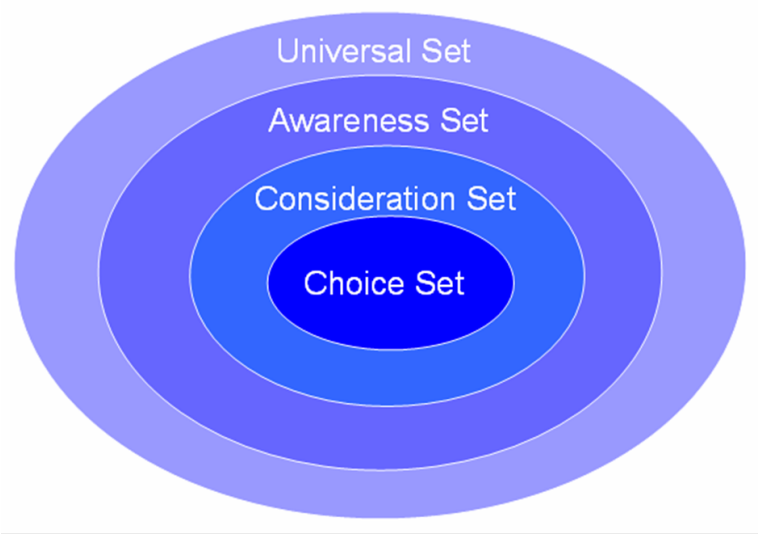

Figure 2.7: Consumer decision making process

The goal of learning and analyzing consideration sets is to better understand and predict consumers' behaviors. Different groups of consumers typically exhibit significantly different purchase behaviors, while people within each group could share similar characteristics. If we are able to correctly identify and recognize the heterogeneity and homogeneity among the consumers, we should be able to better predict their purchasing behaviors, better define market segmentation by grouping customers, and better understand market structure by analyzing the products that are often bought together by similar customers.

The analysis of consideration sets is attracting increasing academic and managerial attention. However, the identification and analysis of consideration sets is challenging because they are not directly observable. Both consumer surveys [82, 15] and the actual purchase history [24, 6] could be used to study customers consideration sets. Consumer survey is relatively fast, up to date and direct. But sometimes it may not be very reliable, because customers might behave differently depending on whether they are filling out survey or doing the actual shopping. Scanner data that shows customers purchase history could be used. They are relatively accurate since the purchased products must have been seriously considered by the customers, but they lack any information on products considered but not bought.

In marketing, there are two ways to model consideration effect in empirical discrete choice models:

1. Deterministic methods [82], which allocate brands into considered and unconsidered sets on the basis of direct consumer reports or the values of key consideration variable. For example, Roberts and Lattin [82] defined a consideration set to consist of only those brands for which the expected utility is greater than some threshold.

2. Probabilistic methods [8, 58, 6, 19, 15], which involve a more generalized two-stage modeling process: a consideration set formation stage and a brand selection stage. A customer's final brand choice probability of product j, P_j, is conditional on that brand j being a member of his/her consideration set C. Thus, $P_j = \sum_C P(C)P(j|C)$. For a total of n products, the enumeration of $2^n - 1$ choice set utilities makes it inappropriate for even moderate-sized problems.

In order to avoid this problem, individual-level consideration sets can be estimated using a Bayesian updating procedure in conjunction with the multinomial logit model as shown in [87]. In their model, a brand j will be included in one's consideration set if the utility of considering j is greater than a threshold.

Another method is to use a latent class approach [58, 19] to catch the heterogeneity of consideration probability formation. Their choice model partitions the market into consumer segments differing in both brand preference and price sensitivity.

These marketing models have been constructed in order to investigate the properties of consideration sets and how they affect the consumers' purchasing behavior. So far, no method has been proposed to identify the actual consideration sets for a certain group of customers. The number of possible consideration sets goes up exponentially with the number of elements, making the problem unrealistic to analyze with conventional methods for a large numbers of products. Data mining methods could be potentially applicable to this problem. For example, Vroomen et al. [92] developed a two-stage parametric econometric model. Their model is actually an artificial neural network, and the consideration set corresponds with the hidden layer. The disadvantage of models using neural networks is that it is difficult to interpret the relationships between the consideration sets (hidden layer) and the outputs.

2.4 Divide and Conquer

As described in Chapter 1, clustering can be used to partition unlabeled data into groups to find the unknown structure of the dataset. On the other hand, for a dataset with known labels, clustering can be used as a preprocessing method for

other algorithms, such as classification. In machine learning, the idea of building specific predictive models for different clusters is an example of a "divide and conquer" algorithm. There are two advantages of having the two stages, namely, clustering and then classification. First of all, the model built for each cluster is more specific to each cluster than a uniform model built for the whole dataset. Secondly, better understanding of the different groups can be achieved. It's natural to analyze found clusters separately and compare how differently they respond to predictive variables. This idea has extensive real world applications. For example, marketing segmentation is used to identify groups or subpopulations of customers within a data set that share unique properties relevant to their future choices. Better understanding and interpretation of the data can be achieved by analyzing the unique decision model of each particular subpopulation. This is the second step of our work after we have successfully found the clusters in the datasets. I will introduce related work in the fields of both data mining and marketing in the following subsections.

2.4.1 Data Mining Algorithms

Decision tree induction [80] is the most well-known divide-and-conquer machine learning algorithm. It can be regarded as one approach for realizing feature space localization in supervised learning. At each node, the data points are split according to one (or more) selected feature(s). Then for each part, different features might be selected to further divide the data set. Thus the subtrees can be considered different predictive models for different groups of examples. Decision trees attempt to find pure subtrees consisting of mostly or all one class.

Radial Basis Function (RBF) neural networks as proposed by Moody and

Darken [71], are another approach. The model consists of one hidden layer of unsupervised or competitive learning and one supervised output layer. Each hidden unit has its own *receptive field* in the input space defined by Gaussian functions. The activations are determined by the distance from the data point to the Gaussian centers. The prediction is then determined by a weighted combination of the activations of the hidden units. The hidden units (clusters) found by the algorithm are usually mostly positives or negatives, then the cluster membership contributes a single weight for the class prediction. Only one model is built for the whole data set, instead of particular model for each part of data set.

More complex models have also been proposed for recognizing localized responses. For example, Maclin [69] proposed RegionBoost, which realized feature space localization by making use of an ensemble of classifiers. In standard boosting, the predictions are combined by weighting the predictions by a term related to the accuracy of the classifier on the training data. It ignores the fact that later classifiers concentrate on correctly classifying examples that can't be correctly classified by previous classifiers and thus may only be good at classifying patterns similar to only that small subset. Rather than using a single measure of the accuracy on the training data, RegionBoost takes advantage of the idea that some classifiers perform well only for certain regions of the feature space by estimating the likely accuracy of the classifier for each new point. One limitation of using RegionBoost is that an ensemble of predictive models instead of a simple predictive model is built, thus it lacks interpretability.

ICC (Iterative Clustering with Classification) [101] is an integrated two-level modeling process that uses the performance of different predictive models that are built for each particular cluster to guide the cluster construction, while taking the

similarities between the data points in the original feature space into consideration. ICC combines supervised and unsupervised learning to build accurate predictive models while facilitating interpretability. The traditional EM clustering algorithm was extended by considering both the similarity of the data points in the original feature space and the predictive accuracy. The model is integrated because the performance of the classifier directs the changes in the cluster centers and variances so that the data points are more likely to belong to a cluster that correctly predicts its class. Thus, the model considers both the probability that a data point belongs to a cluster and the probability that a data point can be correctly classified in that cluster. We are trying to not only segment the data into subgroups of similar individuals, but also to segment them into more easily classified and more easily explained subgroups.

Another novel feature of this work is that different sets of attributes can be used to build the clusters and the classifiers. Let x_i^A and x_i^B represent the data point x_i projected onto two sets of features used to build clusters and classifiers respectively. x_i^A and x_i^B can be identical, disjoint, or have some overlap. x_i^A is used by the EM algorithm to build clusters, and x_i^B is used to build distribution classifiers separately for each particular cluster.

Given the probability distribution, the likelihood function is the joint probability function of the sample. In EM, the likelihood to be maximized for data set $X = \{x_i\}_{i=1}^N$ is

$$L_{EM} = \prod_i^N \sum_{k=1}^K p_k f(x_i|\theta_k), \tag{2.1}$$

where $f(x_i|\theta_k)$ is the pdf (probability density function) for cluster k, and θ_k is the set of Gaussian parameters (μ_k, σ_k) for cluster k, with mixing proportion of p_k.

In logistic regression, for the instance x_i with J attributes, the predicted

probability that $y_i = 1$ is $P_i = \frac{1}{1+exp(-\beta_0 - \sum_{j=1}^{J} \beta_j x_{ij})}$. The contribution of the ith case to the likelihood function equals P_i if $y_i = 1$, and it equals $1 - P_i$ if $Y_i = 0$. Thus the contribution of the ith case being correctly classified by regression model h is $P(h(x_i) = y_i) = P_i^{y_i}(1 - P_i)^{1-y_i}$. Assuming that the data points are independent, the likelihood function can be expressed as:

$$L_{logistic} = \prod_{i=1}^{N} P(h(x_i) = y_i) = \prod_{i=1}^{N} P_i^{y_i}(1 - P_i)^{1-y_i}. \tag{2.2}$$

The objective of ICC is to maximize both the likelihood that the data points are formed from the mixture models that can be described by the parameters of p_k and θ_k, and the likelihood that the data points can be correctly classified in their corresponding cluster. Combining the likelihood function of both the mixture model and logistic regression, the new likelihood function to be

$$\begin{aligned} L_{ICC} &= \prod_{i}^{N} \sum_{k=1}^{K} p_k f(x_i^A|\theta_k) P(h(x_i^B|\beta_k) = y_i) \\ &= \prod_{i}^{N} \sum_{k=1}^{K} p_k f(x_i^A|\theta_k) P_i^{y_i}(1 - P_i)^{1-y_i}. \end{aligned} \tag{2.3}$$

In this way, the data points are clustered not only by their similarity, but also by the probability that they can be correctly classified by the classifier built for their assigned cluster. The predictive model can be expressed as

$$\begin{aligned} \hat{y_i} &\sim \sum_{k=1}^{K} p_k f(x_i^A|\mu_k, \sigma_k) P(h(x_i^B|\beta_k) = 1) \\ &= \sum_{k=1}^{K} p_k \frac{1}{\sqrt{2\pi}\sigma_k} exp\left[\frac{-(x_i^A - \mu_k)^2}{2\sigma_k^2}\right] \frac{1}{1 + exp(-\beta_k x_i^B)}. \end{aligned} \tag{2.4}$$

2.4.2 Marketing Methods

In the marketing segmentation literature, recent work has focused on "clustering" individuals based on how well they fit one of several predictive models. This approach is known as clusterwise regression, and was first proposed by Spath [89]. Given N J-dimensional data points x_i $(i = 1, ..., N)$ with class label y_i for each data point, the method minimizes the sum of squared error computed over all points and all clusters:

$$\min \sum_{k=1}^{K} \sum_{i \in C_k} (y_i - \sum_{j=1}^{J} \beta_{kj} x_{ij})^2, \tag{2.5}$$

where C_k is the set of data points for cluster k, and β_{kj} is the jth regression coefficient for cluster k. The method is similar to K-means in the sense that it sequentially exchanges data points to disjoint clusters to minimize the objective function (2.5). In this model, each data point belongs to only one cluster.

Desarbo and Cron [26] extend the method by allowing each data point to belong to several clusters with different probabilities. They used a mixture model to represent clusters. Given K clusters, let σ_k^2 be the variance term for the k-th cluster with proportion p_k. They assume y_i is distributed as a finite sum or mixture of conditional univariate normal densities:

$$\begin{aligned} \hat{y_i} &\sim \sum_{k=1}^{K} p_k f(y_i | x_{ij}, \sigma_k^2, \beta_{jk}) \\ &= \sum_{k=1}^{K} p_k \frac{1}{\sqrt{2\pi}\sigma_k} exp[\frac{-(y_i - \sum_{j=1}^{J} \beta_{kj} x_{ij})^2}{2\sigma_k^2}]. \end{aligned} \tag{2.6}$$

Given a sample of N independent data points, the likelihood can be expressed as

$$L_{clusterwise} = \prod_{i=1}^{N} \sum_{k=1}^{K} p_k f(y_i | x_{ij}, \sigma_k^2, \beta_{jk}). \tag{2.7}$$

Maximum likelihood is used to find the estimates of separate regression functions

for different clusters and the probability that each data point belongs to different clusters. It's worth noting that the variance (σ_k) here measures how well the data points fit the regression line. Thus the model builds clusters depending on how well the data points are along the regression line instead of how similar they are in the original feature space.

Lwin and Martin [68], De Soete and Desarbo [36] and Wedel and Desarbo [93] developed conditional mixture binomial probit and logit regression models. Kamakura and Russell [58] developed conditional mixture multinomial logit regression models in marketing context.

Wedel and Desarbo [94] developed GLIMMIX (Generalized LInear Model MIXture) to handle different types of exponential distributions of the cluster structure for different linear regression models. Wedel and Kamakura [95] contains a review of developments in mixture regression models.

Basically, the mixture regression models build clusters based on coefficient similarities. The objective is to cluster data points in a way that fits the regression function well, without considering the geometric similarities of the data points in their original feature space. In marketing, that means the marketing segmentation is determined based solely on the parameters of the response model.

CHAPTER 3
HACS

Data mining algorithms such as clustering and itemset mining have been developed extensively. The selection of an algorithm to solve a particular problem depends on both the purposes of the analysis and the types of the data. In order to analyze the special structure of consideration sets, we find that none of the existing methods are directly applicable. We also find that the problem of finding 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. We now introduce a Heuristic Algorithm for Clustering Subsets (HACS) [102] for binary data to identify and characterize cluster structures for consideration sets.

The proposed work will be carried out in two steps, shown in detail in the subsequent subsections. 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. In order to increase efficiency, we also propose top-n and quality threshold methods to reduce the number of candidate clusters to be examined.

3.1 Order Candidate Cluster Centers

The challenge of identifying consideration sets is that they are not directly observable. We know only the choice set, those products purchased by a customer over a finite time window, which may be only a subset of the products considered. To induce consideration sets using customers' purchase history, we proceed from

the following assumptions:

1. The purchased products should be included in one's consideration set. This follows directly from the definition of consideration and choice sets.

2. One's consideration set might also include some products that have not been purchased yet. This could be due to several reasons, say, consumer waiting for a promotion, purchase history not long enough, etc. Any finite data collection is inevitably an incomplete picture of the purchasing habits of at least some customers, since new customers are constantly arriving and old ones disappear.

3. The unpurchased products in one's consideration set can be inferred from the purchase records of other consumers who share similar purchase histories. This is a rewording of the basic assumption of similarity-based learning. 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 has probably also considered C.

With these, we define a candidate consideration set as a set with the following desirable property: most people who bought items from the set, bought *only* items from the set. In order to find such sets, we count the number of customers who purchase exclusively within a particular itemset x, and the number who purchase a strict superset of x. These two counts are compared to an estimate 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. We note that in consideration set analysis, the items are

assumed to be alternatives, e.g., different brands of ketchup, although this assumption does not affect the algorithm. 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 3.1) 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, and our later cluster construction is based on this first scan of data which computes is_x. We further define partial support, ps_x, as the number of times each itemset x appears as a strict subset of a customer's choice set. In other words, partial support is the number of times that itemset x appears *partially* in a customer's choice set. We note that $ps_x + is_x$ is equivalent to $support_x$ in itemset mining. The number of times each itemset x appears as a superset of a customer's choice set is called total support, ts_x. It can also be explained as the number of times that itemset x appears as the total choices of some customers, meaning that they only choose from itemset x. Thus, $ps_x = \sum_{y \supset x} is_y$, and $ts_x = \sum_{y \subseteq x} is_y$. As shown in Figure 3.1, each node can be viewed in the shape of an hourglass, with subsets above and supersets below.

As an example, Figure 3.2 shows is_x in parentheses for each itemset x. For a dataset that contains four products {A, B, C, D} and 250 customers, we have $ps_{AB} = is_{ABC} + is_{ABD} + is_{ABCD} = 7 + 10 + 4 = 21$, 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

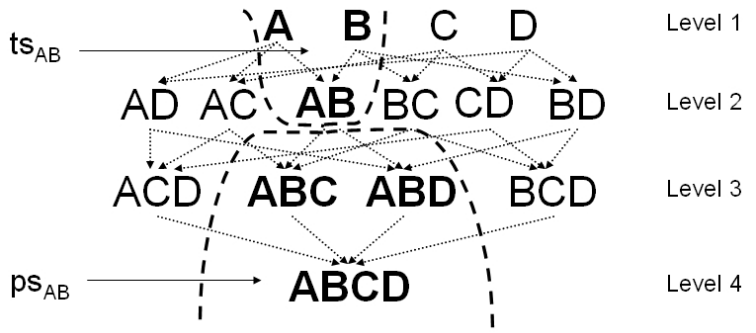

Figure 3.1: Statistics for sample subset lattice

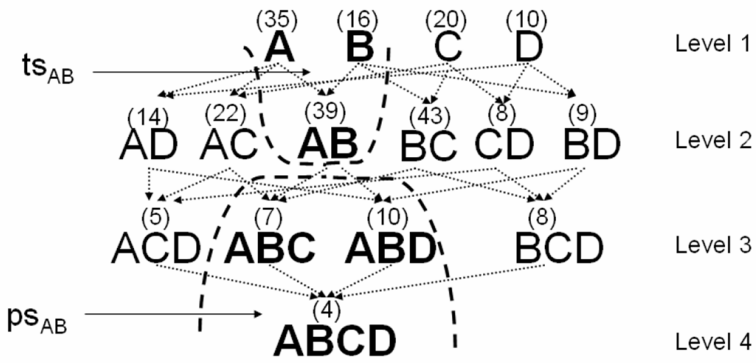

Figure 3.2: ts_x and ps_x illustration

buying only from this set of products. These customers buy either all or some of the products in this set and none from outside the 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.

Now, the question is how to calculate the expected values. In our algorithm, the *expected* ps_x $E(ps_x)$ and *expected* ts_x $E(ts_x)$ are estimated using the is_y, for all singleton sets $y \subseteq x$. Given N customers, the probability that any single item i appears in any customer's purchase history is estimated as $p_i = (is_i + ps_i)/N$. Assuming that the products are independent, the support expectations can be computed as follows:

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

$$E(ps_x) = N \prod_{i \in x} p_i - E(is_x),$$

$$E(ts_x) = N \prod_{j \notin x} (1 - p_j).$$

In our example, the expected values are calculated by individual $p_i, i \in \{A, B, C, D\}$ as follows:

$$E(is_{AB}) = N(p_A)(p_B)(1 - p_C)(1 - p_D),$$

$$E(ps_{AB}) = N(p_A)(p_B) - E(is_{AB}),$$

$$E(ts_{AB}) = N(1 - p_C)(1 - p_D).$$

Each node in the lattice is considered to be a candidate cluster center. A good candidate cluster center should follow these two criteria:

1. The set itself, plus all of its subsets, appears more often than statistical expectation. Here, we define:

$$total\ support\ ratio = \frac{ts_x}{E(ts_x)}.$$

   We want *total support ratio* to be large.

2. The set's strict supersets appear less often than statistical expectation. Here, we define:

$$partial\ support\ ratio = \frac{ps_x}{E(ps_x)}.$$

   We want the *partial support ratio* to be small.

These criteria ensure that candidate sets are shared by a large number of customers, and also represent common purchasing behavior. We introduce an overall measure of quality, $quality_x$ for a subset node,

$$quality_x = total\ support\ ratio - partial\ support\ ratio.$$

The nodes that have higher quality values are those with a larger than expected collection of subsets (the top of the hourglass), and a smaller than expected collection of supersets (the bottom of the hourglass). For example, if {A,B} ranked highest among those itemsets in the second level of the lattice, we would expect there are more than the expected number of customers who bought exclusively within product set {A,B}, and less than the expected number of customers who bought other products together with {A,B}. Therefore we would conclude that {A,B} fits the pattern of a consideration set.

The advantage of using the ratio of real over expected number of points for each node is that it is independent of where in the lattice the node appears. If the

real numbers of ts and ps were used to determine quality, the nodes at the bottom would almost always be preferred, since there are typically very few customers with large choice sets.

We note that the quality measure is not defined as

$$quality_x = \frac{total\ support\ ratio}{partial\ support\ ratio} = \frac{ts_x E(ps_x)}{E(ts_x) ps_x},$$

because as the number of items increases, customers' choice sets could be very diversified with 2^n possibilities. As an advantage of our method, we only calculate the statistics of the choice sets that appear in the real scanner dataset, which is only a subset of all the possibilities. Therefore, some itemsets may not have a strict superset appearing in our lattice structure, and the *partial support* of such an itemset will be 0. In cases like this, if we use *total support ratio / partial support ratio*, we will get an infinitely large quality measure. Thus we can't distinguish the quality of such itemsets. Our current quality calculation is design to solved this problem.

3.2 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 will be marked as a confirmed consideration set, and (some of) the support of the subsets of x will be allocated to x. Otherwise, the support of x is available for allocation to one or more of its supersets.

For the itemsets on the first level of the subset lattice, if $ts_x \geq QT$, x will be

marked as a *confirmed* consideration set, and its support won't contribute to any of its supersets. For itemsets on other levels, the candidate with the highest quality value will be checked first to see if its ts_x is bigger than QT or not. We note that the ts_x is calculated as the sum of the support of all its subsets that have not been previously allocated. The process continues to the very bottom level of the subset lattice.

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. Details are described in Section 3.8.

There are two methods of allocating an itemset to its superset(s), *hard* assignment and *soft* assignment. In hard assignment, when an itemset is allocated to a confirmed cluster, it can no longer contribute its support to the total support of other potential clusters. Conversely in soft assignment, an itemset can contribute to the support of multiple clusters with different weights. After an itemset x is allocated to cluster y, the support remaining for x to contribute to other clusters is $(1 - p(y|x))(is_x)$, where the conditional probability of a customer who bought set x having considered set y is estimated as $p(y|x) = is_y/ps_x$.

Using the example in Figure 3.1, in hard assignment, if {A,B} is declared to be a consideration set, itemsets {A}, {B}, and {A,B} contribute all their support {A,B} and none to the total support of other supersets such as {A,B,C}. In soft assignment, itemsets {A} and {B} can contribute some proportion of their support to other candidate clusters, while itemset {A,B} will not contribute its support to any other candidate clusters. For example, if {A,B} is the first constructed cluster at level 2, some proportion of itemset {A}'s support can be added to the total

support of itemsets such as {A,D}, {A,C}, {A,C,D}, and {A,B,D}. The proportion is $1-p(AB|A) = 1-is_{AB}/ps_A$. Again, we use is_{AB}/ps_A to estimate the percentage of all A buyers who considered exactly the set {A,B}. One minus this proportion is how much A can contribute to other candidates. As an itemset x is allocated to more than one cluster, the proportion that it has available contribute to others continues to decrease: $(1-\sum_y p(y|x))(is_x)$, where $x \subset y$ and y has been confirmed as a consideration set. A confirmed consideration set does not contribute any of its individual support to the total support of any superset in either hard or soft assignment.

3.3 Efficiency Improvements

For a dataset with a large number of items, the described algorithm is inefficient 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 set to be 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. As shown in Section 3.8, using top-n search takes less than 1% of the normal running time, with a small drop in resulting quality.

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 particularly dense datasets as described in Section 3.9, and shows around 50% runtime efficiency improvement.

Both of these two methods provide limit number of candidates to be checked

without removing the important ones. The number of clusters we are interested in analyzing are typically less than twenty, because if more clusters are constructed, they may not be big enough to be representative. The number of candidates provided using both methods are much bigger than twenty. Together with the quality ordering in each level, the candidates should be the ones we really need to check.

Figure 3.3 outlines the algorithm.

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

//Compute individual support
for  each choice set record c ∈ D
   is_c ++
end

//Compute other node statistics
for  each itemset x
   ts_x = Σ_{y⊆x} is_y
   ps_x = Σ_{y⊃x} is_y
   E(is_x) = N ∏_{i∈x} p_i ∏_{j∉x}(1 − p_j)
   E(ts_x) = N ∏_{j∉x}(1 − p_j)
   E(ps_x) = N ∏_{i∈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, ∀y ⊃ x
      end
   end
end
```

Figure 3.3: HACS algorithm outline

3.4 Algorithm Discussion

Our algorithm differs from other related algorithms in several ways. First of all, our algorithm handles the itemsets in a different way from commonly-used itemset mining. Our algorithm is inspired by analyzing the consideration sets in marketing, and our goal is not to find the frequent itemsets, nor the closed/maximal frequent itemsets. The candidate itemsets that we are interested in are not necessarily frequent ones. Rather, they are the itemsets for which a significant number of customers bought from the set, and bought only from inside the set. We introduced the ratios of real values with expected values as a consistent measure with which to compute the criteria to order the candidate clusters. The structure of the lattice efficiently stores the information, as we construct nodes only for those choice sets that actually appear in a given dataset. The memory space required is therefore linear in the number of distinct itemsets that are included in the transaction database. While this number is theoretically exponential in the number of items, in practice we find that for datasets with a large number of items, the number of actual choice sets is much smaller. For example, in a real dataset with 27 items described in the next section, only 3273 of more than 100 million possible choice sets actually appear.

Secondly, our algorithm is a special type of agglomerative hierarchical clustering algorithm, although it is different from existing ones. We merge itemsets that have subset-superset relationships compared to other hierarchical clustering algorithms in which similarity is the only criteria. The clusters constructed by other hierarchical clustering can be visualized by a tree of clusters, compared with ours that can be visualized by a lattice, which allows each node to have multiple parents.

Another important feature that distinguishes our method from other clustering

Table 3.1: Parameters of simulated data

	Simulation I		Simulation II	
parameters	default value	real value	default value	real value
number of Consideration Sets	A=7	unknown	A=10	unknown
number of products	B=8	B=8	B=27	B=27
number of customers	C=800	C=734	C=6000	C=5979
mean customer history length	D=7	D= 6.85	D=14	D= 13.16
per. buy outside of CS	F=0.01	unknown	F=0.01	unknown
mean number of items in a CS	G=3	G^a=3	G=5	G^a=4.4
std. dev. of items in a CS	H=2.0	H^a=1.5	N/A	N/A
number of infrequent large CS	I=4	unknown	I=15	unknown

[a] True values indicate choice sets, since consideration sets are not directly observable

algorithms is that the special structure of the dataset was taken into consideration when clusters are built. In our algorithm, an itemset will only be clustered to its superset. In terms of consideration set analysis, this ensures that a customer with a certain choice set won't be clustered to a cluster that contains fewer items than his/her choice set.

3.5 Evaluation and Experiments

We tested our algorithm on both simulated data and real marketing datasets. We developed a data simulator to generate data that closely resemble the real customers' purchase history. The simulated data was used to evaluate the ability of

our algorithm to identify clusters known to be in the dataset. Our approach is to generate lifelike customer records with specific consideration sets, along with some noise, and evaluate our algorithm on its ability to reconstruct these sets. 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.

3.6 The Data Simulator

The assumptions of the simulator are 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 purchases outside the consideration set with some small probability (say, because of a mother-in-law visit). First, the data simulator generates the consideration sets and their corresponding prior probabilities with respect to customers. Then the purchase history of each customer is simulated, as described in the following subsections.

The parameters of the data simulator are listed in Table 3.1. Using these, we create simulated data that are very much like the real datasets we have in hand. The third and fifth columns of Table 3.1 show the statistics of the real data sets we simulate. Differences between true parameters and the value used in the simulation, such as the distribution of purchase history length, were empirically chosen to create the right number of unique, non-empty sets. For example, the ketchup dataset contains 8 products and 145 unique customer choice sets (out of $2^8 = 256$ possible sets). Using the default values shown in the second column of Table 3.1, we generated 10 simulated datasets and observed a mean number of unique choice sets of 144.8.

3.6.1 Generate Consideration Sets

The simulation begins by generating A consideration sets. The cardinality of the sets follows different distributions as we observed for the two datasets. We note that we can only observe the choice sets instead of customers' consideration sets. Therefore, the cardinalities shown here are all for choice sets, and we assume consideration sets follow the same distribution with the same parameters as choice sets. For the ketchup dataset, it follows a normal distribution as shown in Figure 3.4. A random number is generated from Normal(G,H) and rounded to an integer to determine the cardinality. Non-positive numbers are ignored. Also, we allow no more than two consideration sets with cardinality of one. In the example shown, we generate 7 consideration sets, and the cardinality of the sets is drawn from Normal(4,3).

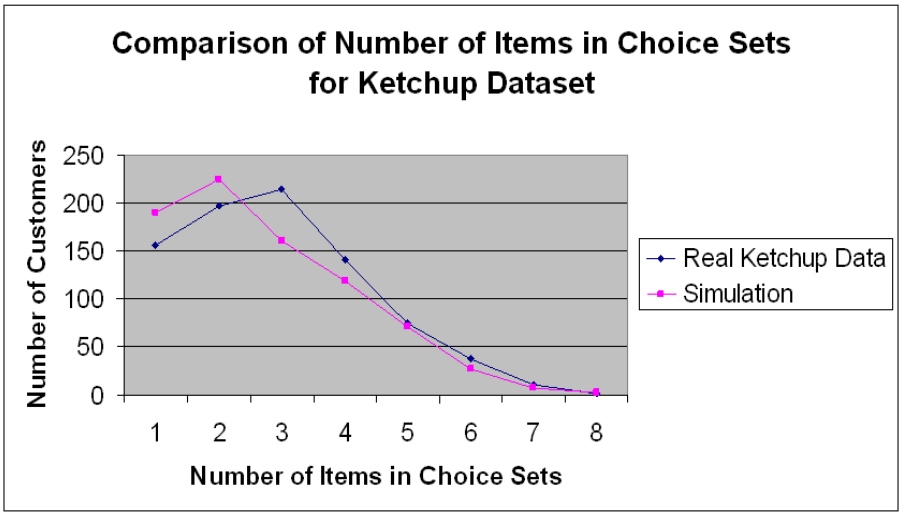

Figure 3.4: Choice set distribution comparison for ketchup

The cardinality of the choice sets for the detergent dataset follows a negative exponential distribution (G) as shown in Figure 3.5. Thus, for the detergent dataset simulation, a random number is generated from a negative exponential distribution with mean G. Again, we allow no more than two consideration sets with cardinality of one.

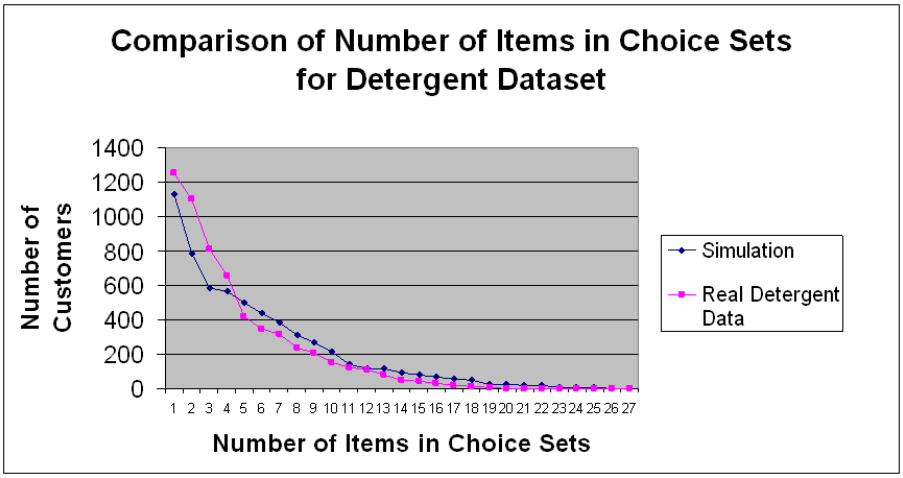

Figure 3.5: Choice set distribution comparison for detergent

The real mean and standard deviation of the number of products purchased by customers are 3 and 1.5, respectively, as shown in Table 3.1. The sizes of consideration sets should, on average, be larger than the size of choice sets, since in any finite observation sample we will not see most customers buy every item in their set.

If all consumers choose only from these consideration sets, the number of

distinct choice sets appearing would be much smaller than observed in the real data set. Thus, we used two methods to increase the variability of choice sets. One method is to generate I infrequent but large consideration sets. The number of items in the I infrequent consideration sets is uniformly distributed from five to the number of products we are simulating. This matches the somewhat surprising observation that in our real data set that contains 8 products, there are quite a number of infrequent choice sets containing 6 or more items. The second method is to add a consideration set that contains every item. This set will serve as the superset of all possible choice sets for the dataset. It will be one of the I infrequent consideration sets. All these infrequent consideration sets will be given smaller priors compared with the ones generated following either Normal or Negative Exponential distribution. In total, the expected number of consideration sets generated is A + I, and we will evaluate our method on its ability to extract all of them, both small and large.

After the cardinality of a consideration set is fixed, a random combination of distinct products is chosen according to its cardinality. For example, for a consideration set with cardinality 4, products 1, 4, 5, and 7 may be chosen.

3.6.2 Generate Priors for Consideration Sets

A uniform random number is generated for each consideration set to represent its prior probability with respect to customers. We set the range of the random number for small consideration sets to be between 0.25 and 0.75, and between 0.15 and 0.35 for large and infrequent consideration sets. These priors are normalized to sum to 1. On average, the small consideration sets will have higher probabilities than the large ones, as desired. However the range of priors is relatively moderate so

Table 3.2: Sample of simulated consideration sets

	Consideration Sets	prior
1	{6}	13.44%
2	{1, 6}	10.93%
3	{4, 5}	9.14%
4	{1, 5, 7}	9.96%
5	{0, 1, 2, 4}	12.78%
6	{0, 1, 2, 3, 4}	14.70%
7	{1, 2, 3, 4, 6}	5.82%
8	{1, 2, 3, 4, 6, 7}	4.55%
9	{0, 2, 4, 5, 6, 7}	4.61%
10	{0, 2, 3, 4, 5, 6, 7}	6.78%
11	{0, 1, 2, 3, 4, 5, 6, 7}	5.69%

that each constructed consideration set will have a reasonable number of customers assigned to it. As an example, consideration sets created in a typical run are shown in Table 3.2.

3.6.3 Simulate Each Customer's Choices

Each customer is first assigned to a consideration set according to the consideration set priors, as previously generated. We then generate priors for all products in the customer's consideration set. Similar to the priors for consideration sets, the priors for each product in a customer's set are generated uniformly between 0.25

and 0.75 and normalized to sum to 1.

The distribution of the number of purchases of ketchup made by customers follows a negative exponential distribution, as was observed in the real ketchup dataset shown in Figure 3.6. There are many customers who buy only a few times, and only a few customers with very long purchase histories. This is reasonable, as the time window on our real datasets is around 2-3 years. The simulation matches real customer purchase records well, except for the number of customers with only one purchase. This is caused by the nature of the negative exponential distribution. We accept the simulation because it only affect the cluster construction of those with cardinality of one, and we already limited such choice sets as we generate candidate consideration sets. Another reason is that although there are more than the expected number of customers with only one purchase record, their consideration sets are different, thus their single choices should also be different. So the chances that this may accidentally generate a "false" consideration set is very small.

The comparison of detergent purchase history distribution is shown in Figure 3.7. The simulation matches the real dataset well.

Finally, for each purchase, a customer makes a choice according to the generated priors on the products in his consideration set. We also allow a simulated customer to choose from outside his consideration set with a small probability F. We note that F should always be small, around 1-5%, otherwise, the product should be regarded as being within his consideration set. Although the question remains controversial in marketing, we believe it is realistic to view rare purchases as falling outside the consideration set; however, our main purpose for including such events in our simulator was to observe the effects of increased noise on our clustering performance.

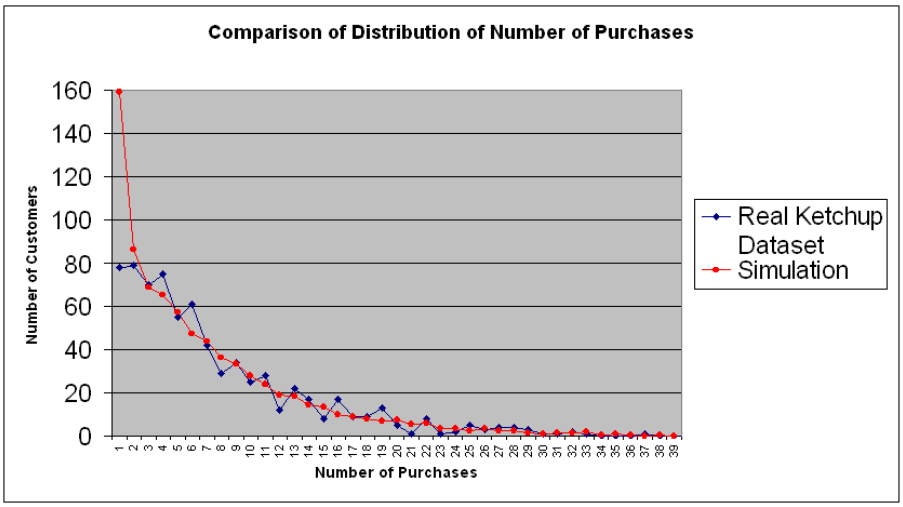

Figure 3.6: Real customer purchase history distribution for ketchup dataset

Our simulator provides very realistic datasets on which to test our approach. Emergent properties such as the total number of non-empty choice sets and the distribution of customers with choice sets of different sizes match our true data very closely. We conclude that, if our clustering method can reliably reconstruct the signal (true consideration sets) in this data, then the clusters constructed from true data will be trustworthy.

3.7 Evaluation

Clearly we would like our algorithm to find existing clusters, and not generate clusters that are not in the data. Precision and recall are two widely used measurements in information retrieval. Given a confusion matrix as shown in Table 3.3, we

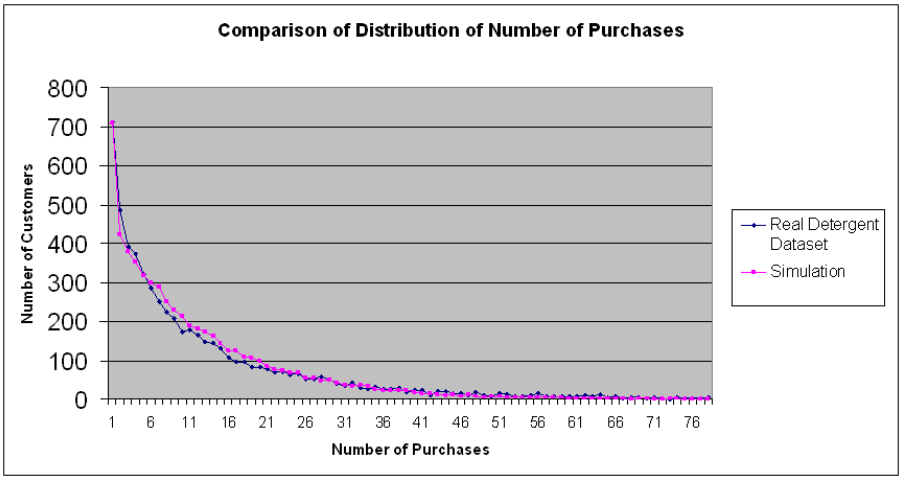

Figure 3.7: Real customer purchase history distribution for detergent dataset

have:

$$precision = \frac{TP}{TP + FP},$$

$$recall = \frac{TP}{TP + FN}.$$

Precision measures the fraction of records that actually to be positive in the group the classifier has predicted as a positive class. Recall determines the fraction of positive examples correctly predicted by the classifier. In the context of cluster analysis, precision measures the fraction of found clusters that are known clusters, and recall measures the fraction of the true clusters that are found.

Standard measurements such as precision and recall are inadequate for measuring cluster quality in our situation, since some misses are closer than others. For example, for known clusters {1234} and {4567}, both groups of found clusters

Table 3.3: Confusion matrix for a binary classification

		Predicted Class	
		+	-
Actual Class	+	TP	FN
	-	FP	TN

{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 borrow the idea of set similarity measurement, the Jaccard coefficient [53], 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. For example, given A={1234} and B={123}, $match_{AB} = \frac{3}{4} = 0.75$.

With a measure to compare subset similarity, we now need a way to compare sets of subsets. In general, we want to match up the most similar clusters between the true and the found sets, and compute match scores for the connected pairs. Given two groups of clusters that may have different sizes, we want every cluster to be matched with at least one cluster in the other group. Specifically, given two groups of clusters $\mathcal{A}$ and $\mathcal{B}$, with $|\mathcal{A}| \geq |\mathcal{B}|$, we want every cluster in the smaller group $\mathcal{B}$ to be matched to at least one in the larger group $\mathcal{A}$, and every cluster in $\mathcal{A}$ to match exactly one in $\mathcal{B}$. 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 algorithm used to find the best matches is shown in Figure 3.8. The overall matching score of two

groups of clusters will be the summation of the individual match scores divided by the total number of clusters in $\mathcal{A}$. It's a number between 0 and 1, with 1 indicating perfect match and 0 indicating no matches at all.

As shown in Figure 3.8, the first *for* loop finds 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 $n \geq 1$. For example, there are two groups of clusters: group $\mathcal{A}$ is {12}, {35}, {34}, {134}, {1234}, {1245}, and group $\mathcal{B}$ is {12}, {35}, {45}, {123}, {12345}. After the *for* loop, the matching will be like Figure 3.9(a) with a total score equal to 4.6, and match score 0.76. After the first step, there may be clusters in $\mathcal{B}$ that are unmatched.

The following *while* loop finds matches for the unmatched clusters by re-arranging the matchings. The clusters in $\mathcal{A}$ that have 1:1 matches are marked as unavailable, because they are perfect matches. Then at each iteration we find the best match for an unmatched cluster in $\mathcal{B}$ that introduces the least overall match score decrease. In the end, some unmatched cluster in $\mathcal{B}$ may not get its best match in $\mathcal{A}$. The total score change is the new match score minus the original match score. The unmatched cluster with the least score change is matched first, and so on, until all clusters are matched.

Using the same example, among the unmatched clusters {45} and {123}, we next choose to match the unmatched cluster that results in the smallest match score decrease. The best matching for {45} is {34}, which introduces -0.07 matching score. The best matching for {123} is {1234}, which introduces -0.05 matching score. Thus, {123} is matched with {1234} first. After that, {1234} becomes unavailable, and {45} can only choose from {34}, {134}, and {1245}. {34} will be

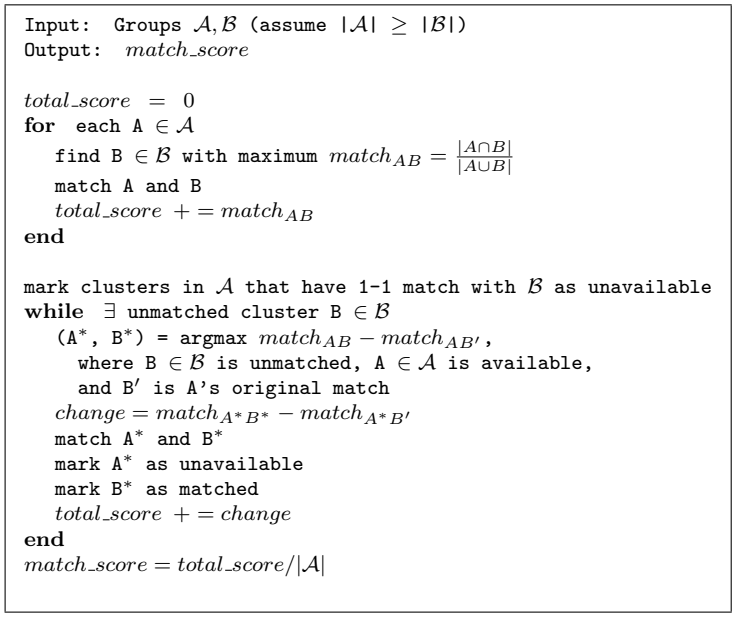

Figure 3.8: Matching algorithm outline

chosen and introduces -0.07 matching score. The final match is shown in Figure 3.9(b) with a total score equal to 4.48, and match score 0.74.

We note that we are using "match" differently than it appears in graph theory [67, 41] . A *matching* in a graph G=(V, E) is a subset M of the edges E such that no two edges in M share a common end note. A graph is *bipartite* if it has two kinds of nodes and the edges are only allowed between nodes of different kinds. A *maximum cardinality matching* is a matching with a maximum number of edges. A *maximum weighted matching* is a matching such that the sum of the weights of the edges in the matching is maximum. Thus, a *maximum weighted maximum cardinality matching* is a maximum cardinality

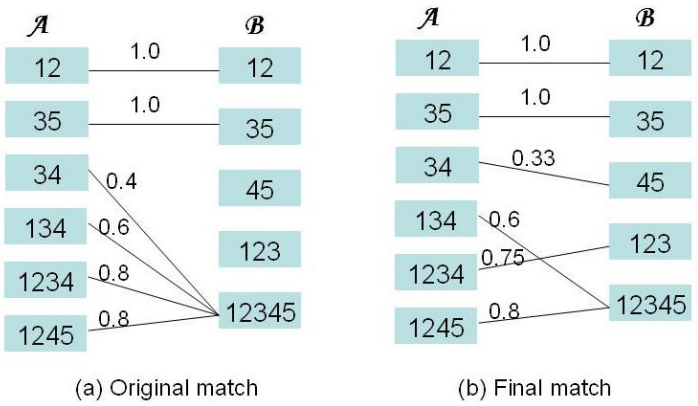

Figure 3.9: Matching example

matching with maximum weight.

The standard bipartite weighted matching problem is defined as following: given a weighted bipartite network $G = (N_1 \cup N_2, A)$, with $|N_1| = |N_2|$ and arc weights c_{ij}, find a perfect matching of maximum weight. There are two major differences between our matching algorithm and this one. First of all, we allow the cardinalities of the two kinds of nodes to be different. Secondly, nodes in the bigger set to have links with more than one node in the smaller set.

The bipartite matching problem allows the the cardinalities of the two kinds of nodes to be different. One approach is to transform the problem into a maximum flow problem [5, 14] by creating a directed version of the underlying graph G by designating all arcs as pointing from the nodes in N_1 to the nodes in N_2. Based on that, a source node s and a sink node t are added, with an arc connecting s to

each node in N_1 and an arc connecting each node in N_2 to t. The global optimal solution can be found in $O(\sqrt{n}m)$ time for an n-node and m-arc network. However, the matching result given by this algorithm enforces one-to-one matches, leaving some nodes unmatched. We are exploring the possibility that a global optimum to our problem can be found using an extension of the method in [5].

The match score measures the similarity between the known cluster structure with the one found by our algorithm. However, we note that this measurement doesn't take the size of the clusters into consideration. For example, given the known cluster {1234} with 1500 instances, if there are two found clusters {12345} with 300 instances and {12346} with 1600 instances, both of the clusters will have the same match score compared with the known cluster. Taking the size of the cluster into consideration, the one with similar size should match the known cluster better. The match score measurement could evaluate the clusters better with some improvement to consider cluster sizes.

3.8 Simulation

We simulated two real-world datasets. One is a dense dataset with 8 items, 145 distinct complete sets, and 6513 customer records. The other is a sparse dataset with 27 items, 3273 distinct complete sets and 5978 customers. Simulation parameters were set empirically to match the characteristics of these real datasets. In clustering, *top-n* search (as described in Section 3.3) is applied to the sparse data to increase efficiency dramatically with little decrease in the match score of the found clusters. Specifically, for a simulated sparse dataset, with the *top-n* search, the running time is 1439 seconds, with a match score of 0.85, compared to without *top-n* search, which takes 316230 seconds, with a match score of 0.89. The quantity threshold

was set to reflect the actual cluster size, particularly, it is between 0.05 to 0.11 for the dense data and between 0.01 to 0.11 for sparse data. All the tests are run on 1800+ MHz, 1 GB RAM Linux workstations. We note that the match score is the comparison between the found clusters with both frequent and infrequent known sets. Specifically, for dense datasets, the found clusters are compared with an average of $7 + 4 = 11$ known clusters, and for sparse datasets, the found clusters are compared with $10 + 15 = 25$ known clusters, as defined in Table 3.1.

Results are shown in Figures 3.10-3.14. Figure 3.10 shows the match score comparison averaged over 10 simulated datasets for the dense data with 1% noise. The match scores for both hard assignment and soft assignment are pretty stable around 0.9. Hard assignment is easier to implement and works better than soft assignment in most of the cases. There is a large and significant ($p < 0.0001$) difference between the clusters found by our algorithm compared to the frequent itemsets. The frequent itemsets are generated using the same quantity threshold as used to find the consideration sets as shown on the x-axis in Figure 3.10.

With the quantity threshold to be 0.08, looking into one detailed result as shown in Table 3.4, we note that soft assignment tends to find more clusters than hard assignment, because more support is allocated to some candidates during the cluster construction process. We also note that frequent itemsets hardly match the known clusters, and they tend to be itemsets that contains a small number of items. We note that the minimum support threshold for frequent itemsets is set to be the same as the quantity threshold as for hard and soft assignment. If we set the quantity threshold to be higher, we will find fewer frequent itemsets. In Table 3.4, column one shows the known clusters, column two shows the clusters found by hard assignment, column three for soft assignment, and column four for

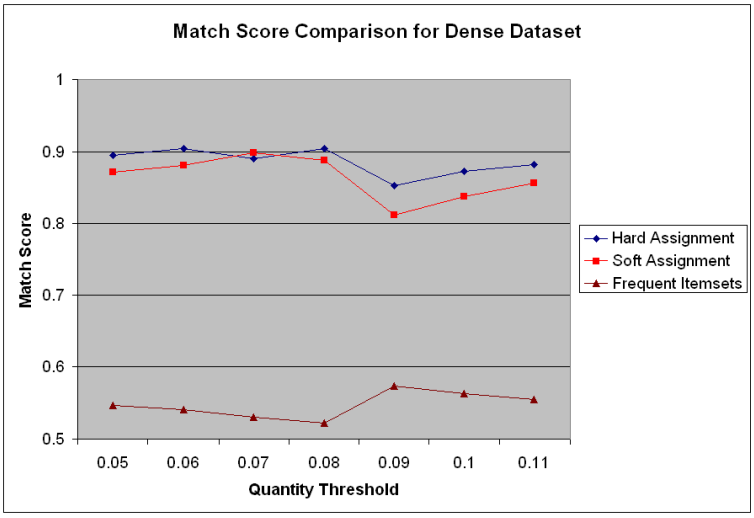

Figure 3.10: Comparison with frequent itemsets for dense dataset with 1% noise

frequent itemsets. (In order to save space, more than one cluster is shown in each row in column four.) Here, the match score for hard assignment is 0.9038, 0.8641 for soft assignment, and 0.5590 for frequent itemsets. It matches the result shown in Figure 3.10.

We simulate the real dataset that contains 145 distinct complete sets by allowing customers to buy outside their consideration sets with 1- 5% chances. The statistics of the simulated dense datasets are shown in Table 3.5. Our algorithm is robust to noise as shown in Figure 3.11. The algorithm performance decreases slightly as the noise increases. However, in general, the match score ranges from 0.8 to 0.9, compared with around 0.5 for the frequent itemsets.

Our algorithm also performs significantly ($p < 0.0001$ for both soft and hard

Table 3.4: Comparison of HACS with frequent itemsets

Known Clusters	Hard Assignment	Soft Assignment	Frequent Itemsets
{0 6}	{0 6}	{0 6}	{0} {1} {2}
{1 5 6}	{5 7}	{0 2 7}	{3} {4} {5}
{3 4 7}	{3 4}	{1 5 6}	{6} {7} {5 7}
{0 2 7}	{0 2 7}	{1 3 7}	{3 7} {2 7}
{1 3 4 5}	{1 5 6 }	{3 4 7}	{0 7} {4 7}
{0 1 2 4 5}	{3 4 7}	{1 4 5 6}	{6 7} {3 5}
{0 1 3 6 7}	{1 3 4 5}	{1 3 4 5}	{2 5} {0 5}
{0 3 4 5 6 7}	{0 1 3 6 7}	{0 2 5 6 7}	{1 5} {4 5}
{0 1 3 4 6 7}	{0 1 2 4 5}	{0 2 3 6 7}	{0 3} {1 3}
{2 3 4 5 6 7}	{0 1 3 4 6 7}	{1 3 4 5 7}	{0 2} {0 1}
{0 1 2 3 4 5 6 7}	{0 3 4 5 6 7}	{0 1 4 5 7}	{3 4} {2 4}
	{2 3 4 5 6 7}	{0 1 3 6 7}	{0 4} {1 4}
	{0 1 2 3 4 5 6 7}	{0 1 2 4 5}	{5 6} {3 6}
		{0 1 3 4 6 7}	{0 6} {1 6}
		{0 3 4 5 6 7}	{4 6} {4 5 7}
		{0 2 3 4 5 6 7}	{0 2 7} {3 4 7}
		{0 1 2 3 4 5 6 7}	{3 6 7} {0 6 7}
			{4 6 7} {3 4 5}
			{0 4 5} {1 4 5}
			{1 5 6} {0 3 6}
			{3 4 6} {0 3 6 7}

Table 3.5: Statistics of the simulated dense dataset with noise

NOISE PERCENTAGE	0%	1%	2%	3%	4%	5%	REAL DATA
MEAN OF DISTINCT CS	144.8	153.7	165.5	169.3	188.1	192.6	145
STDEV OF DISTINCT CS	8.6	9.7	12.8	12.7	11.7	17.6	N/A

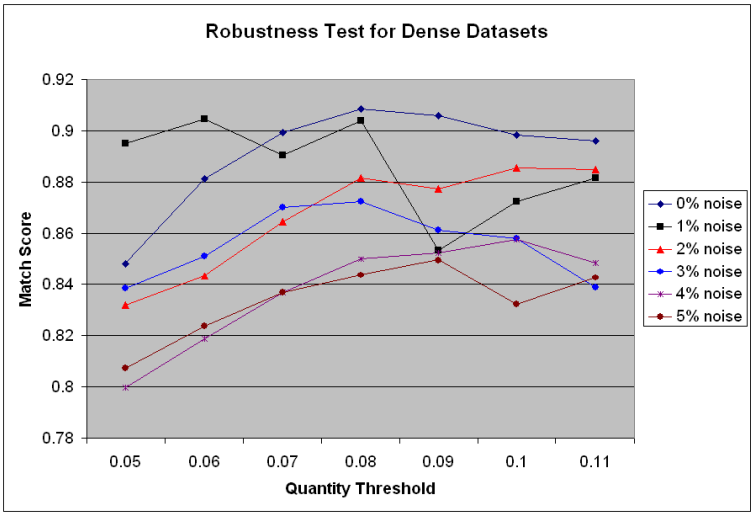

Figure 3.11: Robustness test for dense data

assignment) better than K-means as shown in Figure 3.12. The same datasets were used for this test. Since the user does not directly control the number of clusters in our method, the match scores are recorded depending on the number of found clusters, and then the averaged score is compared with K-means. For

K-means, the distance between two instances is defined as the square root of the sum of differences between each attribute. Since we are handling binary data, the difference is 1 if the two attributes share the same value, and 0 otherwise. As the number of clusters increases, the match score for our algorithm goes down slightly, especially for soft assignment. Again, we note that the number of clusters generated using soft assignment is larger than using hard assignment given the same set of parameters, although the match score is fairly consistent.

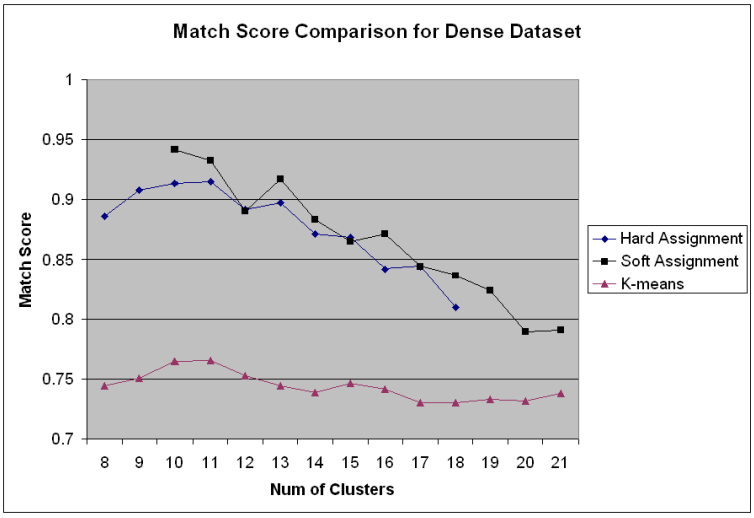

Figure 3.12: Match score comparison with K-means for dense datasets

When the number of clusters is set to be the same as known clusters, the result from K-means is shown in Table 3.6. K-means performs well to find the mid-size clusters, but fails to identify the large-sized ones, and finds some non-existing small-size clusters. The match score for this case is 0.7455, compared to approximately

Table 3.6: Performance of K-means

Known Clusters	K-means
{0 6}	{4}
{1 5 6}	{6}
{3 4 7}	{7}
{0 2 7}	{0 3}
{1 3 4 5}	{0 6}
{0 1 2 4 5}	{1 5 6}
{0 1 3 6 7}	{3 4 7}
{0 3 4 5 6 7}	{0 2 7}
{0 1 3 4 6 7}	{1 3 4 5}
{2 3 4 5 6 7}	{0 1 2 4 5}
{0 1 2 3 4 5 6 7}	{0 1 3 6 7}

Table 3.7: Statistics of the simulated sparse dataset with noise

noise percentage	0%	1%	2%	3%	4%	5%	Real data
Mean of distinct CS	3119.8	3257.8	3421.0	3537.4	3838.6	3957.8	3273
Stdev of distinct CS	174.9	150.4	272.9	275.3	236.2	159.3	N/A

0.9 for HACS.

The statistics of the simulated sparse datasets with noise are shown in Table 3.7, and the match score comparison for the sparse dataset is shown in Figure 3.13. For efficiency reasons, only hard assignment is used in this comparison. The overall match score for the sparse data is not as high as for the dense datasets, while it still shows significant difference ($p < 0.0001$) with the match score for the frequent itemsets. We also note that the highest match score is reached when the quantity threshold is much smaller than that for the dense dataset, because the simulated detergent dataset is much sparser than the simulated ketchup dataset. This allows the method to find interesting but infrequent consideration sets with relatively large numbers of items. In such a sparse environment, we see that frequent itemset analysis performs poorly.

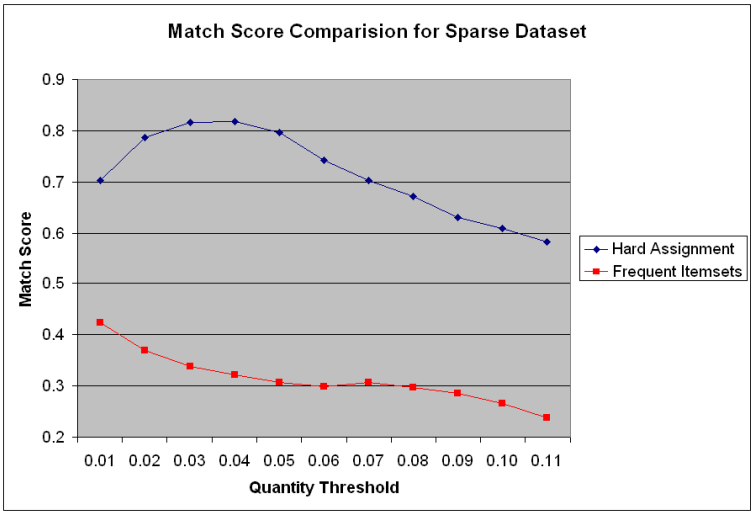

Figure 3.13: Match score comparison with frequent itemsets for sparse datasets

For sparse datasets, our algorithm performs about the same as K-means when the number of clusters is small. As the number of clusters increases, our algorithm beats K-means accordingly as shown in Figure 3.14. HACS performs significantly ($p < 0.05$) better than k-means for number 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 our algorithm are spread more evenly across the levels of the lattice structure, which makes our algorithm suitable to find structures in sparse datasets.

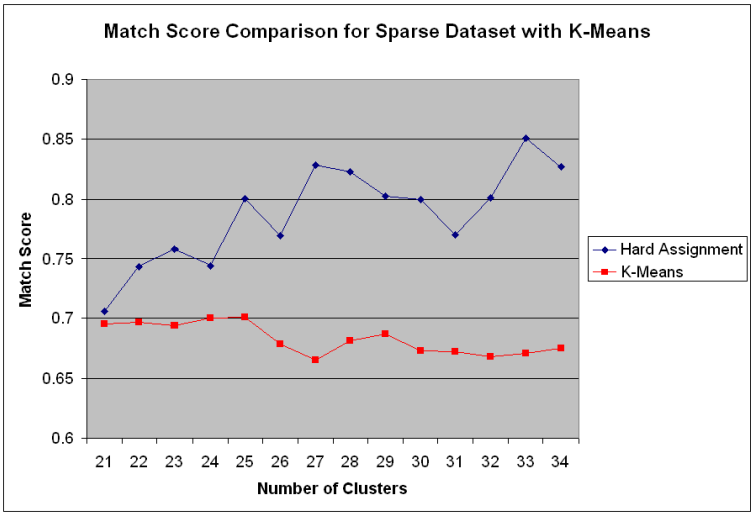

Figure 3.14: Match score comparison with K-means for sparse datasets

Our algorithm is robust to noise for sparse dataset as well, as shown in Figure 3.15. The algorithm performance decreases slightly as the noise increases, while

in general, the match score still significantly outperforms that for frequent itemsets.

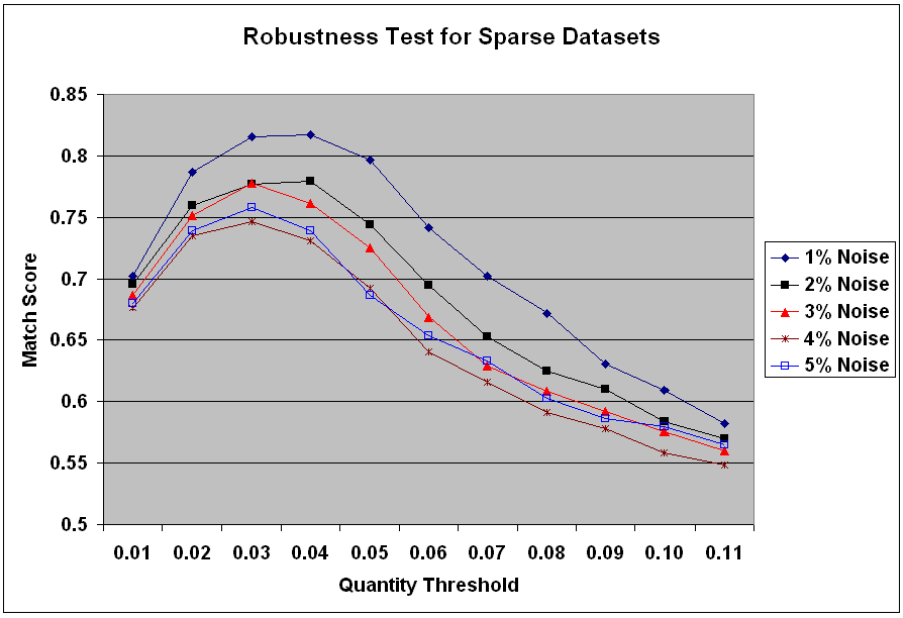

Figure 3.15: Robustness test for sparse data

In general, our algorithm performs extremely well at extracting known consideration sets from the simulated data, dominating both k-means clustering and frequent itemsets. K-means performs reasonably well at this task, but the K-means objective searches for clusters with the wrong shape and does not enforce the hierarchical relationship needed for subset clustering. We do not show comparisons with other clustering algorithms, say EM, because they also do not enforce the special hierarchical relationship 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.

3.9 Real Datasets

The proposed algorithm has been tested with two real datasets. One contains customers' purchase history from 8 brands of ketchup. Efficiency improvement methods are not necessary for this test, since the number of products is small. In general we would like to set our search parameters to the best values as determined by the simulation, however in this case, not using quality threshold will result in finding an unnecessarily large number of clusters. Experimentally, setting the quality threshold to be positive will result in finding only one cluster that contains every item, which indicates the quality threshold is too strict. By relaxing the quality threshold, the number of clusters increases as shown in Figure 3.16. Quantity threshold was set to be 7%, because this will generate the clusters of the sizes we are interested in. Together with a quality threshold at -0.5, the result is a more understandable market segmentation.

Since the number of products is small, the computational efficiency is not a big concern here. Using the quality threshold, the running time is 2.82 seconds, compared to 5.13 seconds without applying the quality threshold. As shown in Table 3.8, among 734 customers, only half of them consider at least 4 brands of the 8. From the result, we infer that Heinz is not only the leading brand in ketchup, but is also favored across all types of customers. Its middle-sized products, 28oz and 32oz, are included in everyone's consideration set. People who buy smaller Heinz products may also consider other brands, but those who include the large-size Heinz offering are typically loyal to the brand.

Table 3.8: 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+

Figure 3.16: Relationship between quality threshold and number of clusters

The algorithm has also been tested with market scanner data containing 27 different 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 3.9, s stands for small size, m stands for medium, and l stands for large. The top-n selection method constructs clusters with too many items, which is not informative for marketing analysis. Therefore, the quality threshold is used to reduce candidates. Seven clusters are constructed when the quantity threshold equals 5%, and the quality threshold equals 3 as shown in Table 3.9. We note that the quantity threshold for this dataset is smaller than for the ketchup dataset. The reason is that this is a much more sparse dataset, so if we set the quantity threshold too high, we will get only two or three clusters.

The result indicates that Tide is a dominant brand that exists in almost everyone's consideration set. Among the 5979 customers, around 10% of the customers consider only Tide. Beyond Tide, however, the extracted consideration sets are surprisingly large and varied, with few overlaps. This may be partially explained by the fact that detergent brand/sizes are not strictly substitutes. Many households may keep multiple items on hand for different laundry tasks. The market segmentation defined by our result may contain other results that are useful for marketing researchers. For example, small-sized items may in fact serve as substitutes, as most consideration sets contain nearly all medium and large sizes. Purex and Surf bands occur almost exclusively together, with consideration set 6 containing all offerings of both brands.

We note that in all cases we have a default cluster that contains all items, which is allocated all subsets that are not allocated to some other cluster. The number of customers in these 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 are representative of a large number of 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. For example, as we relax the quality threshold to be two, we will have 14 clusters, with 2425 points belonging to the last cluster; as we set the quality threshold to be one, we will have 33 clusters, with only 400 points belonging to the last cluster. Although the number of points in the last cluster drops significantly, the number of resulting clusters is large, making it uninformative to analyze each of them.

As is usually the case with unsupervised learning, the quality of the result depends largely on the desires of the user, and the "true" number of clusters (and hence, the "right" values for our thresholds) hinges on finding actionable results. In our experiments with real data, we chose to focus on finding a small number of clearly-defined consideration sets, instead of a complete partition of the space, so a fairly large number of cases fell through to the default set.

Since we don't know the exact consideration sets for the real datasets, no match scores are shown for these results. We also don't compare our results with existing marketing models, for several reasons. First of all, most of the marketing models are based on Bayesian models, which means they assume each customer has a unique consideration set, thus these models are not comparable to ours. The other stream of research in marketing segments customers (rather than consideration sets) into different groups. But in their models, each product is included in each consideration set with different weights, compared to our assumption that a product is either considered, or not. Such Bayesian models also do not consider interactions among the products, a primary strength of our method.

Table 3.9: 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

CHAPTER 4
BUILDING CLASSIFICATION MODELS BASED ON CLUSTERING RESULTS

As described in Chapter 1 and 2, clustering can be used to partition unlabeled data into groups to find the unknown structure of the dataset. On the other hand, for a dataset with known labels, clustering can be used as a preprocessing method for other algorithms, such as classification.

The datasets we used for consideration set analysis contain customers' choices, thus we are able to build classifiers to predict their future choices after we found clusters of customers that share the same consideration set. Based on the algorithm introduced in Chapter 3, we can successfully find the consideration set structures and then extend it with a post-processing method by building different predictive models for each individual cluster. We are making the assumption that customers with the same consideration set form a market segment with similar choice models. They may behave similarly to certain changes in marketing variables. For example, in a market with five products {ABCDE}, a group of customers whose consideration set is {ABC} may be sensitive to the price changes of these three products, while their purchase behavior won't change much in response to price changes of products D and E. By analyzing the specific model for each cluster, instead of a uniform model built for the whole dataset, we will better capture the characteristics of each group. Thus, better understanding and prediction of the different groups can be achieved.

There are two issues related to the predictive accuracy of this "divide-and-conquer" method. On one hand, the accuracy goes up because of the smaller number of output classes in most of the models. On the other hand, the accuracy may go down because of the smaller training sets for each cluster. Therefore, we introduce

several prediction weighting methods to analyze the relationship between customers' consideration sets and their purchasing pattern.

4.1 Different Prediction Weighting Methods

Multinomial logistic regression is chosen as the base classifier, because the predictive model is simple, and it is easy to analyze and interpret the coefficients. Since we've already done a fairly complex segmentation of the dataset in the clustering process, we don't need a sophisticated predictive model here. Another reason is that logistic regression is widely used in marketing research.

In our model, different multinomial logistic regression models are built for each of the clusters. Given a set of consideration sets found by the clustering algorithm, if a customer belongs to a confirmed consideration set that has no superset (other than the universal set) as a consideration set, his/her future choices can be predicted using the model built for his/her consideration set. A question arises when predicting choices for a customer who belongs to a consideration set with one or more superset(s) as consideration sets. For example, if both {AB} and {ABC} are consideration sets, for a customer whose current purchase history is {AB}, we are not exactly sure what consideration set this customer has, so there are different ways to predict his/her future choices. Thus we introduce several prediction weighting methods to handle cases like this. In the following subsections, we introduce a hard weighting scheme and two soft weighting schemes. Particularly, the soft weighting schemes follow the idea of ensemble methods [90, 12], which achieve better prediction by combining classifiers.

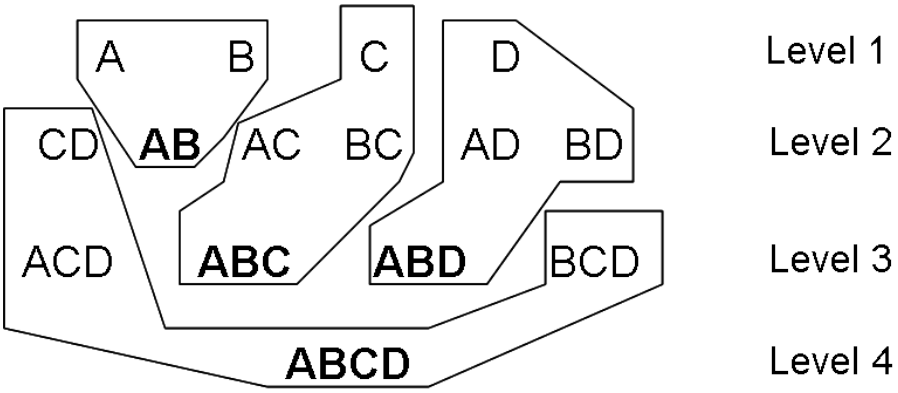

Figure 4.1: Hard weighting scheme example

4.1.1 Hard Weighting Scheme

The hard weighting scheme is to use only the classifier for the consideration set represented by the smallest-cardinality cluster the customer belongs to. In the example shown in Figure 4.1, there are four clusters constructed using our clustering methods, namely, {AB}, {ABC}, {ABD}, and {ABCD}. Using hard assignment, when we want to predict a customer whose current existing choice is {A}, only the classifier for cluster {AB} will be used. This indicates that the other possible choice for this customer is product B, although he only purchased A so far. In this case, we don't consider the possibility that he may choose from products C or D.

4.1.2 Soft Weighting Scheme (A)

Soft Weighting Scheme (A) uses a weighted prediction of the classifiers for all the consideration sets that are supersets of the data of interest. The weight of each

classifier is dependent on the total support ts_x belonging to each cluster x. Here, ts_x is calculated as $ts_x = \sum_y is_y$, where $y \in x$, and is_y hasn't contributed to any other subset x' yet. Using the same example, with four clusters: {AB}, {ABC}, {ABD}, and {ABCD} as shown in Figure 4.2, we have:

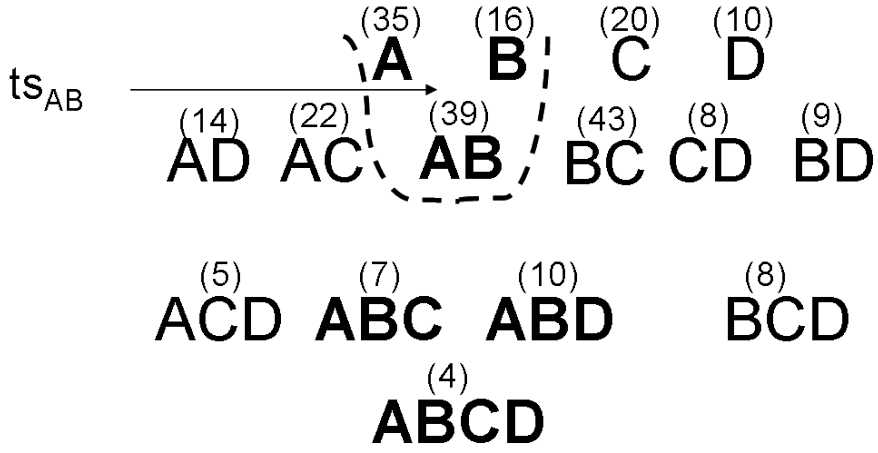

Figure 4.2: Soft weighting scheme example

$$ts_{AB} = is_A + is_B + is_{AB} = 35 + 16 + 39 = 90$$

$$ts_{ABC} = is_C + is_{AC} + is_{BC} + is_{ABC} = 20 + 22 + 43 + 7 = 92$$

$$ts_{ABD} = is_D + is_{AD} + is_{BD} + is_{ABD} = 10 + 14 + 9 + 10 = 43$$

$$ts_{ABCD} = is_{CD} + is_{BD} + is_{ACD} + is_{BCD} + is_{ABCD} = 8 + 9 + 5 + 8 + 4 = 34$$

The total weight is $90 + 92 + 43 + 34 = 259$. The weight of the each classifier w_k is then normalized by the total weight.

In multinomial logistic regression, for the instance x, the predicted probability of a class c ($c \neq C$ number of classes) being chosen is:

$$P_c(x) = \frac{exp\{\beta_c x\}}{\sum_{i=1}^{C-1} exp\{\beta_c x\} + 1}.$$

The predicted probability of a class c ($c = C$) being chosen is:

$$P_c(x) = 1 - \sum_{i=1}^{C-1} P_c.$$

Thus, given the weight of each classifier w_k, where $k = 1...K$ is the number of clusters, and the predicted probability of each class P_c^k, we have the final probability P_c of each class c as $\sum_k w_k P_c^k$. Then the class with the highest final probability will be the predicted choice.

For example, suppose the probabilities predicted for each product by the different classifiers are as shown in Table 4.1. Then, the probability of A being predicted would be:

$$P_A = \sum w_k P_A^k = w_1 P_A^1 + w_2 P_A^2 + w_3 P_A^3 + w_4 P_A^4$$

$$= \frac{90}{259}0.4 + \frac{92}{259}0.2 + \frac{43}{259}0.5 + \frac{34}{259}0.2 = 19.42\%$$

The probability of each product is calculated in the same way, and then the one with the highest probability is chosen as the final prediction.

4.1.3 Soft Weighting Scheme (B)

Soft Weighting Scheme (B) is similar to soft weighting scheme (A) in the sense that it also uses weighted prediction of the classifiers for the supersets of the data of interest. The difference is that the weight for each classifier depends on the number of items in each cluster. Again, using the same example as for soft weighting

Table 4.1: Probability of each product using different classifiers

	A	B	C	D
Classifier 1 for {AB}	40%	60%	N/A	N/A
Classifier 2 for {ABC}	20%	50%	30%	N/A
Classifier 3 for {ABD}	50%	20%	N/A	30%
Classifier 4 for {ABCD}	20%	30%	10%	40%

scheme (A), we have four clusters: {AB}, {ABC}, {ABD}, and {ABCD} as shown in Figure 4.1. We will use the classifier for each cluster to predict a customer whose current existing choice is {A}, because all these clusters are supersets of {A}. The weight for each classifier to predict a customer's choice depends on the cardinality of the customer's known choice set x and its superset y.

For example, the weight for the classifier for cluster {AB} is $|A|/|AB| = 1/2$, the weight for the classifier for cluster {ABC} is $|A|/|ABC| = 1/3$. Similarly, the weight for the other two classifiers are 1/3 and 1/4, respectively. The weights for the classifiers are then normalized to sum to 1. Thus, for a customer with a choice set x, the weight of each superset classifier w_i is:

$$w_i = \frac{|x \cap y_i|/|y_i|}{\sum_i |x \cap y_i|/|y_i|},$$

where y_i ranges over all confirmed consideration sets that are supersets of x.

In this example, we have

$$w_1 = \frac{1/2}{1/2 + 1/3 + 1/3 + 1/4} = 0.35.$$

Similarly, we have $w_2 = w_3 = 0.24$, and $w_4 = 0.17$.

Given the weight of each classifier w_k, and the predicted probability of each class P_i^k as described in Soft Weighting Scheme (A), we have the final probability P_i of each class i as $\sum w_k P_i^k$. Then the class with the highest final probability will be the predicted choice.

4.2 Dataset Description

The dataset we use includes two parts. The first part is the customers purchase history information, which is used to build clusters to find the consideration sets. The second part is the selected corresponding marketing mix variables. Marketing mix refers to the four major areas, sometimes called the Four Ps [23], of decision making in the marketing process. The Four Ps include Product, Price, Place, and Promotion, each of which consists of numerous sub-elements. Product concentrates on product functionality, quality, appearance, and customer support etc. Price is how much the intended customers are willing to pay. It includes list price, discount information, financing and leasing options. Place is the channel of distribution. It includes information about locations, logistics, channel members, channel motivation, and market coverage. Promotion is a communication process a business uses to achieve its awareness. Basic promotion tools include: advertising, direct sales, and media.

In our test, the marketing mix variables we use include: price, display, and feature information for each product when a purchase is made.

In addition, customer loyalty information is included, because a lot of research [27, 29, 57] has shown that loyalty information contributes significantly to prediction accuracy. Since we know customers purchase history, we add the loyalty

Table 4.2: Sample dataset for prediction problems

		1	2	3	...	m
Clustering Variables	purchase history	0	1	1	...	0
Classification Variables (Predictors)	prices	5.57	5.21	4.03	...	2.78
	display	0	0	0	...	1
	feature	0	1	0	...	0
	loyalty	0	12	1	...	0
Response variable	choice	5				

variables as input to the classifier as well. The loyalty information for each customer is the number of times the customer purchases a product. For example, for a total of three products, if a customer purchased product A twice, product B four times, product C nine times, then the loyalty variables are 2, 4, 9.

As shown in Table 4.2, for m products, we record each customer's purchase-related information. We have a set of binary purchase history variables shown in the second row, which use 1 to indicate if a product has been purchased by a customer, and 0 otherwise. This set of variables is used to build clusters as described in Chapter 3. We have a set of predictors, which include price variables, display variables, feature variables, and loyalty variables. Then we have the customer's final choice. This feature set is used to build classifiers. The classifiers are used to predict customers' choices using different weighting schemes as described in the previous subsections.

We use the same two real datasets as for testing the clustering algorithm. One

is the ketchup dataset, which includes 8 products. We were given the price, display and feature information for this dataset. The loyalty information is inferred from customer purchase history.

The other dataset is the detergent dataset. For the detergent dataset, we don't define different sizes of the same brand as different brands as in Chapter 3. Instead, for choice prediction, we record only the brand choice. In addition, we only choose the brands with more than 3% of market share. In this way, we reorganize the detergent dataset into 10 brands. In the new definition, each brand includes an average of 29 "products" with different UPC numbers for each product. For example, different sizes of Tide will have different UPC numbers, and they all belong to Tide in the new definition.

The data was collected from 45 stores for three years (from 1985, 35th week to 1988, 34th week). For each week we have the *price* and the total number of *units* sold for each product at different stores. Also, we have the unit *weight* for each product. In order to get the price for each brand in each week, we handle each week separately, and take the average of $price/(units * weight)$ for each product across all the stores. Thus, the final price is the unit price per ounce for each brand that the customer may see no matter which store he/she shops at. The display information is handled differently than the price information, because we don't need to consider the unit weight. A product is on display no matter what its weight is. For each week, we sum up the number of stores that have a brand on display, and then divide it by the total number of purchases of that brand in that week. Thus, the final display information indicates the probability of a customer saw a brand on display when he/she goes shopping.

For the detergent dataset, we don't have feature information. The loyalty

Table 4.3: Baseline for predictive models

	WITH LOYALTY VARIABLES	WITHOUT LOYALTY VARIABLES	ACCURACY IMPROVEMENT
KETCHUP	57.84%	46.73%	23.77%
DETERGENT	52.05%	43.59%	19.41%

information is collected the same as for ketchup dataset, which is defined as purchase frequency.

4.3 Experiments

4.3.1 Baseline

We compare our prediction weighting schemes with the standard logistic regression model, which builds one model for the whole dataset. We use 3-fold cross-validation for both our methods and logistic regression. For each customer, two-thirds of his/her records will be used for training, and the other one-third for testing. We exclude customers whose purchase history is less than three.

For the baseline, we report the accuracy result both with and without loyalty variables for both datasets as shown in Table 4.3. The results confirm that the loyalty variables are important for predictive accuracy.

4.3.2 Experiments with Our Methods

We tested different weighting schemes on both real datasets. We note that the weighting scheme we mention here is different from the data assignment methods in Chapter 3. The data assignment methods, namely, soft and hard assignment, are for cluster construction to find consideration sets. The weighting schemes, namely, hard and soft (A, B) weighting schemes are for predicting customers' choices.

The results in Chapter 3 indicate that hard assignment finds clusters that better reflect the known structures. Thus, all the prediction experiments in this chapter are based on the clusters found using hard assignment.

4.3.3 Ketchup

We tested the Ketchup dataset using both the hard and soft (A, B) weighting schemes. For the hard weighting scheme, we set a wide range for the parameters to construct clusters so that we can better analyze the relationship between clustering and classification. The quality threshold is tested in the range of [-1, 2], and the quantity threshold is tested in the range of [0.05, 0.15]. As shown in Figure 4.3, the accuracy increases as the quantity threshold increases in general. One of the reasons is that there are more training instances in each cluster with a bigger quantity threshold. The best accuracy 58.41% is reached with quality threshold of 0.1, and quantity threshold of 0.14. When we do the cross-validation, in two out of three folds, we found only two clusters, one containing products {1,2,3} and the default cluster that contains all products {0-7} as shown in Table 4.4. Results in Table 4.4 indicate that the classifiers built for clusters with small cardinalities always outperform the baseline. The quantity threshold is different from that when the highest match score is reached for cluster construction, where the quantity threshold is 0.08

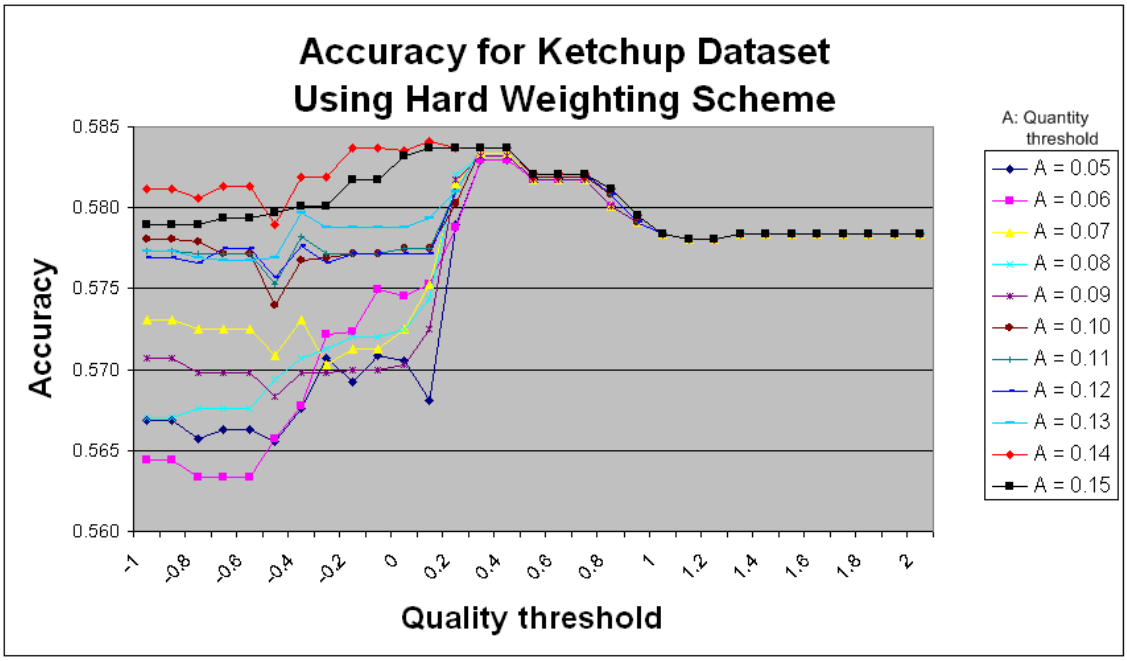

Figure 4.3: Hard weighting scheme accuracy

as shown in Chapter 3.

The overall accuracy for the hard weighting scheme is higher than that of the baseline, which is using one model for the whole dataset. Our scheme performs better for clusters with a small number of items, and performs worse for clusters with a large number of items. As shown in Table 4.4, the number of instances belonging to the clusters with a small number of items is much less than that belonging to the clusters with a large number of items. For example, in Fold 1, there are 930 instances belonging to cluster {1,2,3}, and 3741 instances in cluster {0,1,2,3,4,5,6,7}. Table 4.5 shows the average and standard deviation of accuracy

Table 4.4: Details of clusters and classifiers for ketchup

	Consideration set	# of Training Instances	% of majority class	Training Accuracy	# of Testing Instances	% of majority class	Testing Accuracy
fold 0	{1, 2, 3}	1043	57.62%	74.88%	424	54.25%	72.88%
	{1, 2, 3, 4}	1099	52.96%	74.80%	170	35.88%	60.59%
	{0, 1, 2, 5}	854	55.27%	82.90%	163	33.13%	66.26%
	{0, 1, 2, 4, 5, 6}	1708	44.09%	71.84%	240	33.75%	57.92%
	{1, 2, 3, 4, 5, 6, 7}	2971	40.09%	63.01%	661	26.32%	52.04%
	{0, 1, 2, 3, 4, 5, 6}	3048	41.21%	64.83%	167	29.34%	54.49%
	{0, 1, 2, 3, 4, 5, 6, 7}	4167	38.23%	60.69%	31	25.81%	32.23%
fold 1	{1, 2, 3}	930	58.17%	76.45%	467	51.82%	68.52%
	{0, 1, 2, 3, 4, 5, 6, 7}	3741	37.63%	61.45%	1328	31.17%	52.94%
fold 2	{1, 2, 3}	930	55.70%	75.16%	595	57.14%	69.91%
	{0, 1, 2, 3, 4, 5, 6, 7}	4228	35.10%	61.21%	1200	35.25%	53.17%

0: Heinz 14oz 1: Heinz 28oz 2: Heinz 32oz

3: Heinz 44oz 4: Hunts 5: Delmonte

6: Other 28oz- 7: Other 32oz+

Table 4.5: Accuracy for different sizes of clusters: ketchup dataset

Num. of Items in Consideration Set	Average	Standard Deviation
3	70.45%	1.75%
4	63.21%	3.64%
5	58.82%	3.35%
6	53.88%	5.88%
7	51.62%	5.26%
8	45.45%	8.13%

for tests when the quantity threshold is set to 1.4, and the quality threshold is in the range of [0.05, 0.15]. We note that both the accuracy and the variance get worse as the number of items in the consideration set gets bigger. Or, put another way, for small number of items, the model is both more accurate and more consistent. The quantity threshold is set to 1.4, because it gives the best overall accuracy for the hard weighting scheme.

Figure 4.4 compares the detailed accuracy with the baseline. Hard weighting outperforms the baseline for clusters with less than or equal to five items. The performance of the classifiers for clusters with more items is not as good as the baseline, and this affects the overall performance.

Figures 4.5 and 4.6 show the accuracy for the ketchup dataset using the soft (A, B) weighting schemes. Using both weighting schemes, the highest accuracy is reached when the quantity threshold is set to 0.06, and the quality threshold is in the range of [-6, -4]. More clusters are created with a lower quantity threshold. For

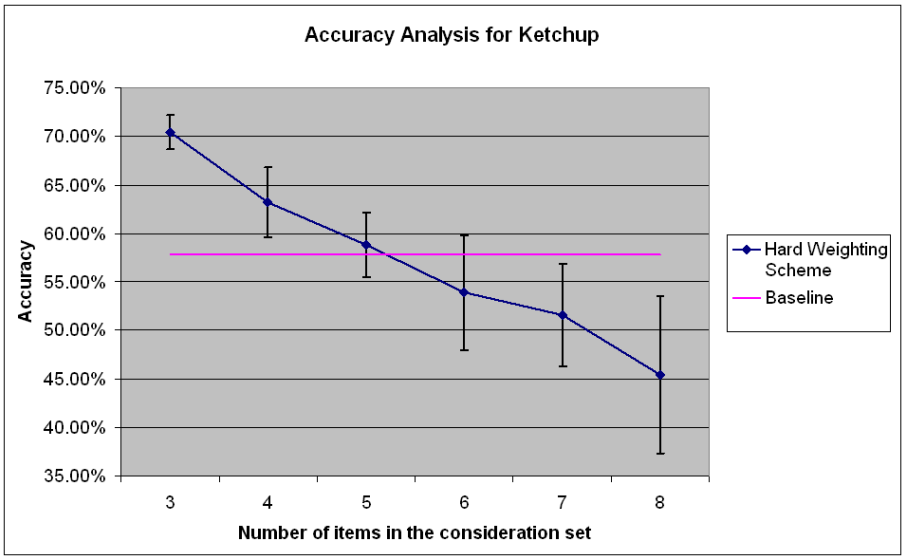

Figure 4.4: Accuracy comparison with baseline for ketchup dataset

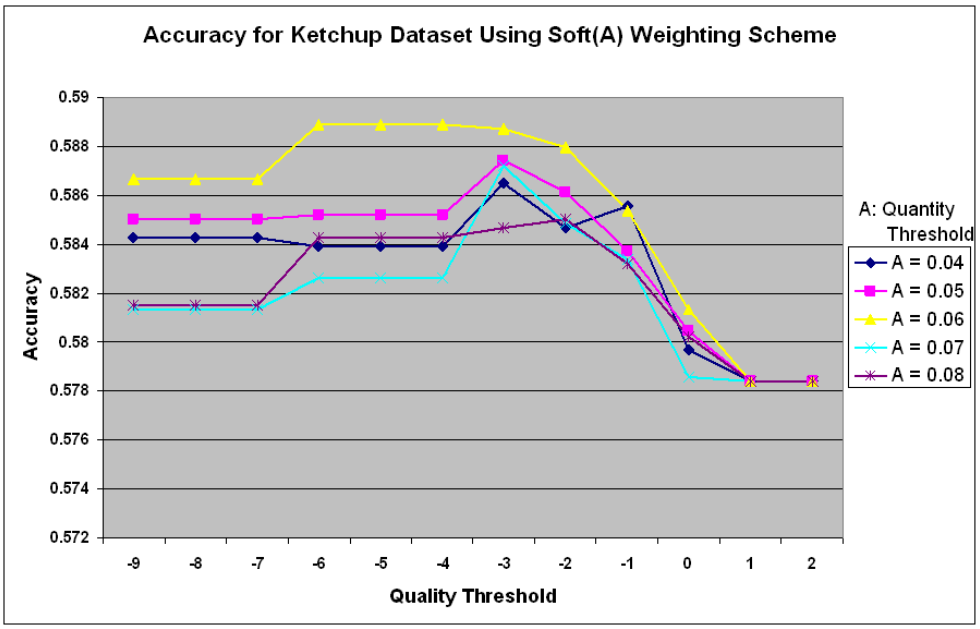

Figure 4.5: Soft (A) weighting scheme for ketchup dataset

example, using soft (A) weighting scheme, on average there are 22 clusters created when the quantity threshold is 0.04, 16 clusters when quantity threshold is 0.06, and 14 clusters when quantity threshold is 0.08. In general, as quality threshold goes up, the predictive accuracy goes down. Only one cluster that contains every item is created when the quality threshold is set to 1 or 2. In cases like this, both weighting schemes perform just like the baseline, which builds one model for the whole dataset.

Table 4.6 shows typical clusters built using different quantity thresholds. Soft

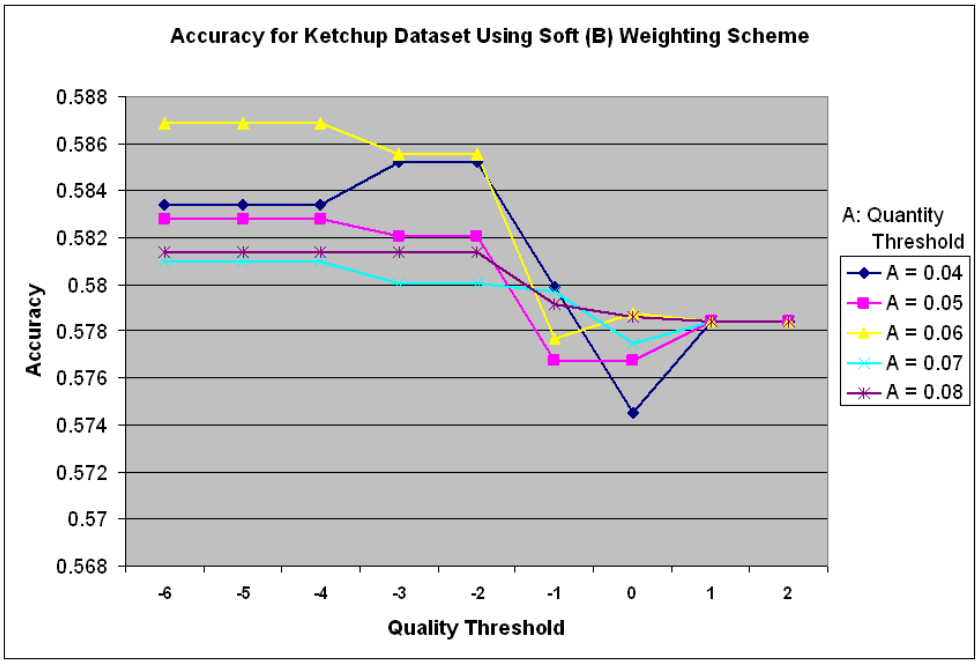

Figure 4.6: Soft (B) weighting scheme for ketchup dataset

weighting schemes work in the same way as ensemble methods by combining prediction from different classifiers. With fewer classifiers, there are more training data for each classifier. On average, there are 980 training instances when quantity threshold is 0.08, 888 instances for threshold 0.06, and 629 instances for threshold 0.04. The performance of the classifiers increases with the number of training instances. Thus the overall prediction accuracy increases. On the other hand, with too few classifiers, the ensemble effect is not obvious. As shown in Table 4.6, on average, 4 classifiers are used in the ensemble when the quantity threshold is 0.04, 5 classifiers for 0.06, and 4 classifiers for 0.08. When the quantity threshold set to 0.06, the classifiers have enough training data and the ensemble of classifiers work, therefore, the best prediction accuracy is achieved.

We note that all the weighting schemes show higher accuracy than the baseline. The best predictive accuracy of both soft weighting schemes is higher than that of the hard weighting scheme as shown in Table 4.7. With the same constructed clusters, soft weighting schemes use more than one classifiers to predict. Both soft weighting schemes improve classification accuracy by aggregating the predictions of multiple classifiers.

4.3.4 Detergent

To test the hard weighting scheme, the quantity threshold was chosen from the range [0.02, 0.09]. As shown in Figure 4.7, the accuracy decreases as the quantity and quality threshold increase. The best accuracy of 53.59% is reached when the quantity threshold is 0.03, and the quality threshold is either -9 or -8. For most of the cases, the hard weighting scheme performs better than the baseline, which is 52.05%.

Table 4.6: Quantity threshold comparison: ketchup dataset with soft (A)

	0.04			0.06			0.08		
	Cluster	training points	# superset	Cluster	training points	# superset	Cluster	training points	# superset
1	12	322	20	12	322	17	12	322	14
2	123	683	9	123	683	9	123	683	9
3	012	259	6	124	253	12	0123	570	3
4	124	253	12	0123	570	3	0124	504	3
5	245	257	11	1235	394	6	1245	641	7
6	0123	458	3	1245	527	8	12346	736	2
7	1235	394	7	01246	509	2	12345	1132	4
8	1234	485	6	12346	622	3	12347	856	3
9	1245	442	8	01247	570	2	12457	915	4
10	1247	294	6	12345	1008	3	012467	897	2
11	01246	397	4	12457	801	3	124567	1172	2
12	02457	378	7	124567	1058	3	123457	1492	2
13	12345	763	5	123457	1441	2	012345	1374	2
14	12457	644	5	012345	1281	2	01234567	2431	1
15	24567	503	4	1234567	1796	2			
16	012567	502	2	01234567	2372	1			
17	124567	837	3						
18	123567	639	3						
19	123457	1124	3						
20	012345	924	2						
21	1234567	1414	2						
22	01234567	1873	1						
Average		629	5.86		888	4.88		980	4.14

Table 4.7: Best accuracy comparison for ketchup dataset

	BASELINE	HARD WEIGHTING	SOFT(A)	SOFT(B)
KETCHUP	57.84%	58.41%	58.89%	58.69%

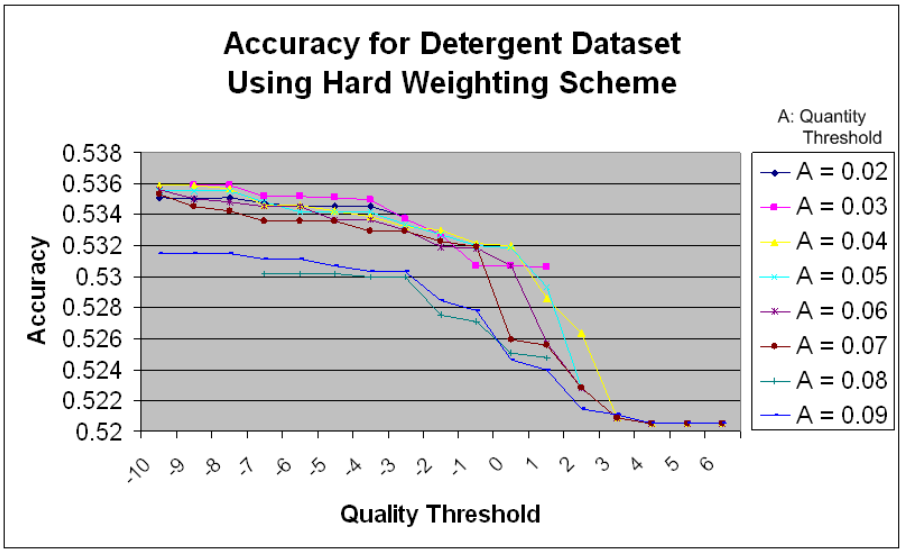

Figure 4.7: Hard weighting scheme for detergent dataset

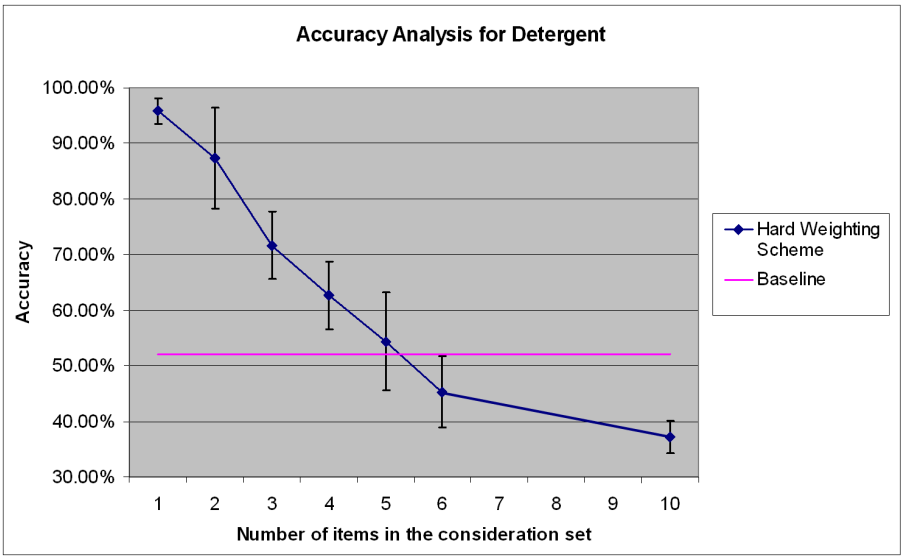

Figure 4.8: Accuracy comparison with baseline for detergent dataset

Looking into the details of the results as shown in Figure 4.8, the hard weighting scheme shows higher accuracy than the baseline for clusters with less than six items. Using the hard weighting scheme, the clusters found by the algorithm mostly have small cardinalities, with only a few clusters containing six items or more. Thus, the effect of the performance of the large clusters is not as strong as that for the ketchup dataset. The overall accuracy is higher than the baseline.

Table 4.8 shows the average and standard deviation of accuracy for clusters with different cardinalities when the quantity threshold is 0.03, and the quality threshold is -9. We note that the accuracy again goes down for larger clusters, but the increase in standard deviation is less consistent than with ketchup.

Table 4.9 shows the clusters found by our algorithm, and the training/testing

Table 4.8: Accuracy for different sizes of clusters: detergent dataset

Num of Items in Consideration Set	Average	Standard Deviation
1	95.84%	2.27%
2	87.27%	9.08%
3	71.61%	6.07%
4	62.65%	6.08%
5	54.38%	8.82%
6	45.27%	6.43%
10	37.26%	2.92%

accuracy for one fold. The clusters of small cardinality are quite different from each other with little overlap, which indicates that customers have quite different consideration sets. Thus the classifiers are different for each cluster.

Figures 4.9 and 4.10 show the accuracy for the detergent dataset using the Soft (A, B) weighting schemes. The highest accuracy of each scheme is summarized in Table 4.10. The highest accuracy of Soft (A) is achieved when the quantity threshold is 0.02. The quantity threshold is 0.01 for Soft (B) for highest accuracy. Both of the quantity thresholds are much smaller than those for the ketchup dataset. The detergent dataset has 81243 instances. Even with a quantity threshold 0.02, on average there are 1893 training instances in each cluster. The large number of training instances ensures the quality of the classifier for each cluster.

It is worth noting that as the quality threshold decreases, the prediction accuracy keeps increasing. Thus, we use $top - n$ candidate selection method introduced

Table 4.9: Model details for detergent using hard weighting scheme

Consideration sets	Training Instances	Training Accuracy	Testing Accuracy	Testing Instances
{9}	3291	100.00%	97.83%	1245
{2, 5}	1777	99.32%	98.41%	694
{6, 7}	988	100.00%	95.81%	358
{1, 9}	911	88.58%	81.82%	352
{2, 9}	1979	83.63%	72.77%	437
{0, 2, 9}	1987	89.63%	72.90%	417
{2, 6, 9}	3865	82.82%	71.19%	913
{2, 8, 9}	1954	88.84%	76.54%	537
{2, 5, 7, 9}	2957	91.72%	70.54%	404
{1, 2, 6, 9}	4747	81.34%	58.12%	685
{2, 4, 6, 9}	3934	82.56%	62.41%	399
{1, 7, 8, 9}	2909	81.20%	62.90%	345
{0, 3, 6, 9}	3210	83.74%	71.18%	340
{0, 2, 5, 6, 9}	6100	82.85%	59.86%	431
{2, 5, 6, 8, 9}	6140	82.40%	63.71%	598
{0, 2, 7, 8, 9}	4726	80.07%	63.03%	403
{1, 2, 4, 5, 9}	4542	82.21%	74.32%	366
{0, 1, 6, 8, 9}	5548	78.39%	52.48%	343
{4, 6, 7, 8, 9}	4639	79.20%	60.98%	264
{0 - 9}	45408	52.78%	39.89%	15376

0: ALL 1: B-3/F-S 2: CH 3: DASH 4: FAB

5: FRESH 6: OXYDOL 7: PUREX 8: SURF 9: TIDE

Table 4.10: Best accuracy comparison for detergent dataset

	BASELINE	HARD WEIGHTING	SOFT(A)	SOFT(B)
DETERGENT	52.05%	53.59%	53.76%	54.19%

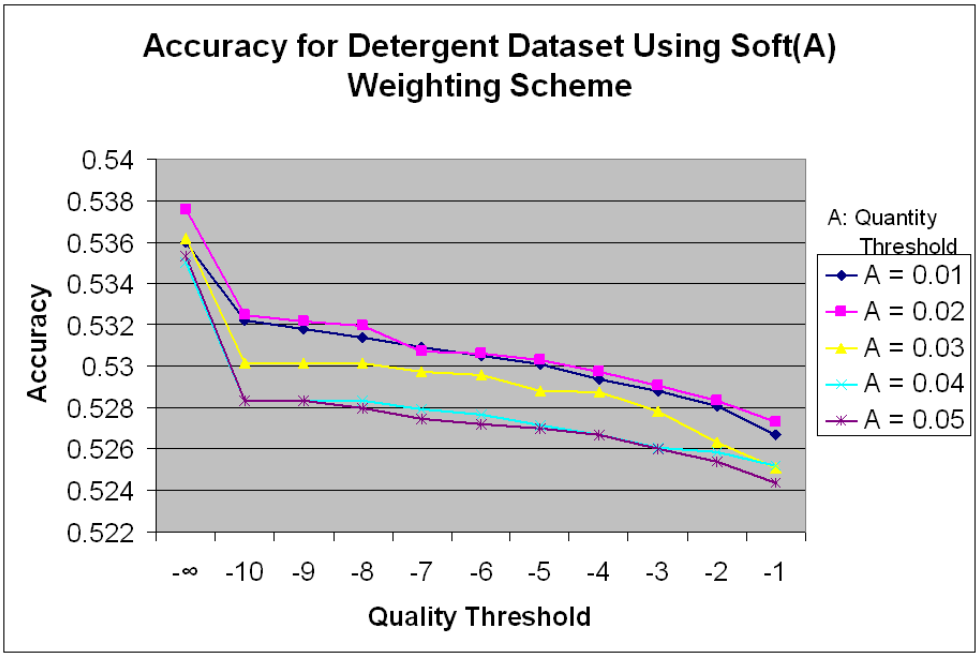

Figure 4.9: Soft (A) weighting scheme for detergent dataset

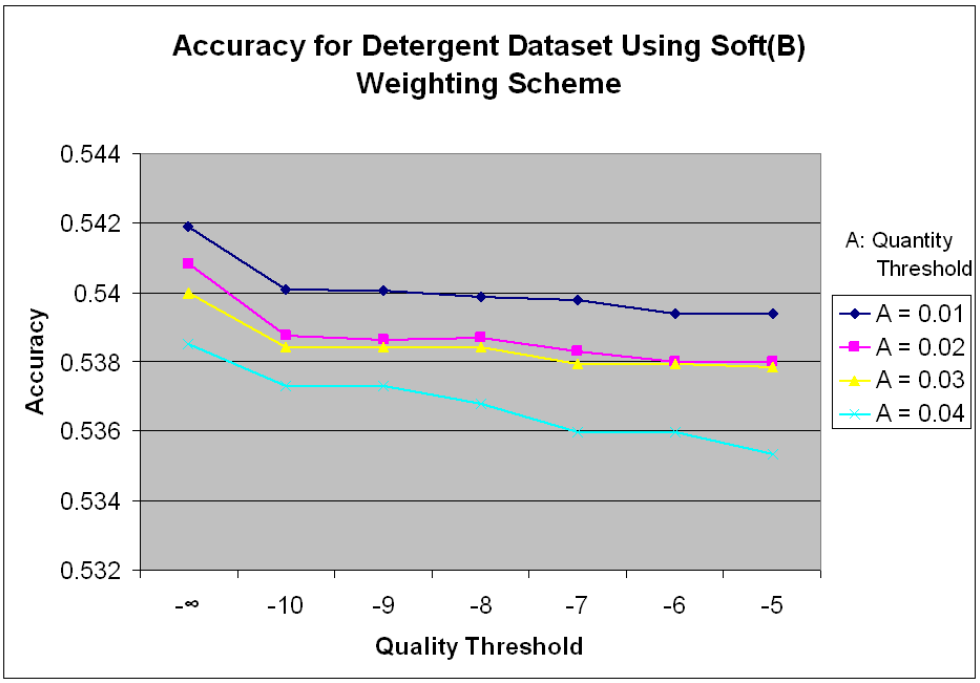

Figure 4.10: Soft (B) weighting scheme for detergent dataset

in Chapter 3 to create clusters. In this case, we don't set the quality threshold, assuming it can be as small as possible. The results in Figures 4.9 and 4.10 show that with no quality threshold, the best accuracy is achieved. The reason is that with no quality threshold, clusters containing more items are created. On the other hand, with a quality threshold, the clusters created are all the ones with small cardinality. For example, Figure 4.11 shows the average cluster cardinalities for soft (A) weighting scheme with quantity threshold 0.01. When the quality threshold is set to -5, all the clusters created are small ones with cardinality less than 5 (except the default one with every item). Therefore, the number of supersets of those clusters are smaller compared with those for the clusters created with no quality threshold. Specifically, the average number of supersets is 7 with a quality threshold -5, compared with 12 with no quality threshold. The ensemble effect is more significant with no quality threshold.

4.3.4.1 Model Interpretation for Detergent Dataset

In our model, different classifiers are built for different clusters. We can find both the similarities and differences between those classifiers by comparing them. This helps to capture the common and different behaviors of different groups of customers.

We analyzed the twenty classifiers built for the clusters shown in Table 4.9. Each classifier model with C output classes consists of a set of coefficients (β's) for $C-1$ of the classes. As expected, the probability of each product being chosen is always negatively related with its price, and positively related with its display and loyalty variables.

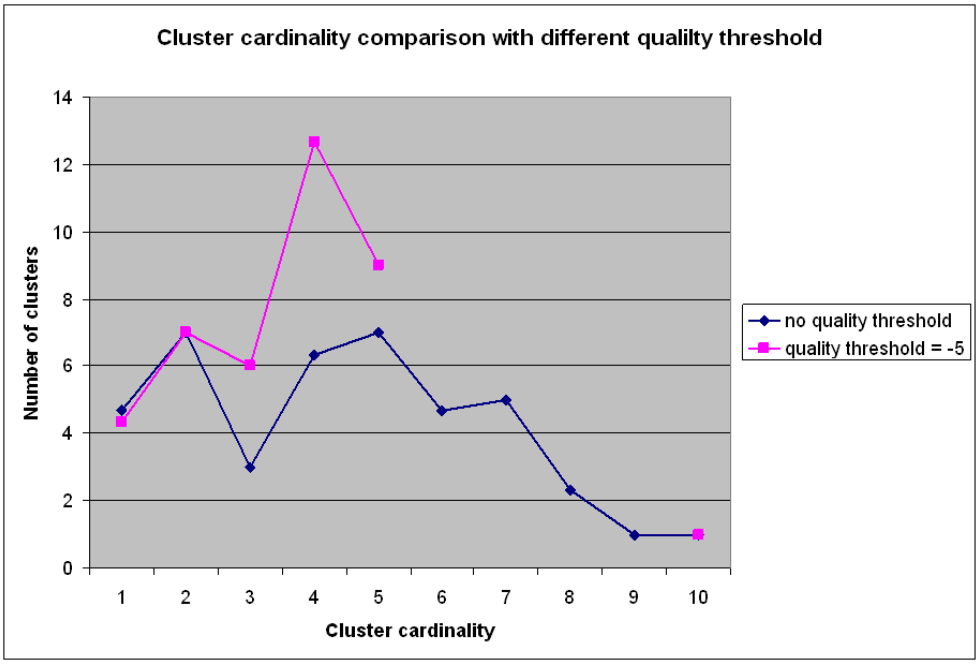

Figure 4.11: Cluster cardinality comparison with different quality threshold

We also note the differences between the classifiers for different models. Customers belong to different consideration set behave differently. Tables 4.11, 4.12, and 4.13 show the coefficients for two clusters, one for consideration set {1, 9}, and the other for consideration set {2, 9}. The *price* 2 coefficient is negative in classifier B, indicating that as the price goes up, the probability of product 2 being chosen decreases. This coefficient is positive in classifier A. The different effects of the same variable won't be noticed if only one classifier is built for the whole dataset. The same thing happens to other variables as well. For example, coefficient *display* 1 is positive in Classifier A, and negative in Classifier B. *loyalty* 2 is negative in Classifier A, and positive in Classifier B.

Customers purchasing the same product may behave differently according to the consideration set they belong to. For example, customers purchase product 2 in both consideration set {2,9} and {2, 6, 9}. Customers in consideration set {2,9} are more sensitive to price changes than customers in the other consideration set, with a *price* 2 coefficient of -1.0359 compared to -0.6539. Another finding is that customers in consideration set {2, 9} are less loyal to product 2 than customers in consideration set {2, 6, 9}, with a *loyalty* 2 coefficient 0.117 compared to 0.2218.

The probability of a product being chosen is affected by the product information of the products in the same consideration set. For example, in Classifier C, P_6 is positively related with *price* 2, with a coefficient 0.1337, meaning that as the price for product 2 increases, it is more likely that customers turn to purchase product 6. The same thing happens to product 2, whose purchase probability is positively (0.0723) related with *price* 6.

Loyalty variable has the same effect as price. P_2 is negatively (-0.0961) related with variable *loyalty* 6. P_6 is negatively (-0.03) related with variable *loyalty* 2.

Within the same consideration set, if customers are more loyal to a certain product, they are less likely to purchase other products.

The display variable has a different effect on product probabilities. Basically, it not only increases its own probability of being chosen, but also increases the probability of the products within its consideration set. For example, P_6 is positively related with both *display* 2 and *display* 6, with a bigger coefficient *display* 6, 3.3813, and a smaller coefficient *display* 2, 0.5451. The reason may be that when a product is on display, it reminds the customers to buy this type of product. For example, when a brand of detergent is on display, the customer will remember to buy some detergent. The probability of the brand on display been chosen increases. But there are also chances that the customer buys the same type of product of other brands. Another interesting finding is that P_2 is positively related with *display* 6, but slightly negatively related with *display* 2.

4.3.4.2 Alternate Model Interpretation for Detergent Dataset

We also test a classification model using only the attributes that belong to the brands in a certain consideration set. Specifically, for a cluster that contains brand 1 and 9, only variables associated with these two brands are used to build the predictive model.

The same patterns are observed using this model. As shown in Table 4.14, in general, the probability of a product being chosen is negatively related with its price, and positively related with its display and loyalty variables.

In general, the probability of a product being chosen is positively related with price increase of the other product(s) in the same consideration set. For example,

Table 4.11: Price coefficients comparison

VARIABLE	CLASSIFIER A	CLASSIFIER B	CLASSIFIER C	
	{1, 9}: P_1	{2, 9}: P_2	{2, 6, 9}: P_2	{2, 6, 9}: P_6
PRICE 0	0.4282	-0.7462	0.6815	1.1694
PRICE 1	-0.5434	-0.3381	0.0141	-0.275
PRICE 2	1.0198	-1.0359	-0.6539	0.1337
PRICE 3	-1.3578	-0.5542	-1.5854	-0.5234
PRICE 4	-0.2518	-0.0574	0.6688	0.9916
PRICE 5	-0.0532	0.0191	0.064	0.0054
PRICE 6	0.2863	-0.162	0.0723	-0.4546
PRICE 7	-1.1035	-0.3546	0.2149	-0.426
PRICE 8	-0.7172	0.109	0.2978	0.2753
PRICE 9	0.1629	0.2972	0.5028	0.6047

0: ALL 1: B-3/F-S 2: CH 3: DASH 4: FAB
5: FRESH 6: OXYDOL 7: PUREX 8: SURF 9: TIDE

Table 4.12: Display coefficients comparison

VARIABLE	CLASSIFIER A	CLASSIFIER B	CLASSIFIER C	
	{1, 9}: P_1	{2, 9}: P_2	{2, 6, 9}: P_2	{2, 6, 9}: P_6
DISPLAY 0	0.493	0.8518	1.6086	1.2483
DISPLAY 1	1.6142	-1.202	-1.1585	-1.762
DISPLAY 2	1.5107	0.1968	-0.0236	0.5451
DISPLAY 3	-2.169	-0.4898	-0.7791	-1.2796
DISPLAY 4	-3.3274	-0.1448	1.5515	1.5513
DISPLAY 5	0.6744	-0.988	0.4134	0.0476
DISPLAY 6	-1.0571	-0.3314	1.3005	3.3813
DISPLAY 7	-0.3245	-0.2276	0.0387	-0.3082
DISPLAY 8	-0.9931	0.3478	1.2062	0.5251
DISPLAY 9	-1.6642	-0.5364	-0.5023	-0.0568

0: ALL 1: B-3/F-S 2: CH 3: DASH 4: FAB
5: FRESH 6: OXYDOL 7: PUREX 8: SURF 9: TIDE

Table 4.13: Loyalty coefficients comparison

VARIABLE	CLASSIFIER A	CLASSIFIER B	CLASSIFIER C	
	{1, 9}: P_1	{2, 9}: P_2	{2, 6, 9}: P_2	{2, 6, 9}: P_6
LOYALTY 0	-0.1291	0.5209	-0.1834	0.0262
LOYALTY 1	0.3488	0.2287	-0.2616	-0.1113
LOYALTY 2	-0.134	0.117	0.2218	-0.03
LOYALTY 3	0.1431	-0.6572	-0.2801	-0.0868
LOYALTY 4	-0.1908	0.5467	-0.1459	-0.2148
LOYALTY 5	-0.0699	-0.7379	-0.0845	-0.1475
LOYALTY 6	-0.4235	-0.0671	-0.0961	0.2317
LOYALTY 7	-1.0315	0.0414	0.2021	0.095
LOYALTY 8	-0.0731	-0.0259	0.0195	-0.0105
LOYALTY 9	-0.198	-0.1878	-0.1238	-0.1414

0: ALL 1: B-3/F-S 2: CH 3: DASH 4: FAB
5: FRESH 6: OXYDOL 7: PUREX 8: SURF 9: TIDE

in Classifier B, P_2 is positively related with price 9 with a coefficient of 0.1876, and in Classifier C, both P_2 and P_6 are positively related with price 9.

For display variables, we note the different relationships between the brands' display information. For a brand that shares the same consideration set with brand 9, the probability of this product being chosen is always negatively related with brand 9's display variable. This indicates that as brand 9 is on display, the probability of other products that share the same consideration set decreases. In contrast, the probability of brand 2 or 6 being chosen is positively related with the other's display variable. This means that as brand 2 is on display, the probability of brand 6 being chosen increases, and vice versa.

As for loyalty variables, we note that the probability of a product being chosen is always negatively related with the loyalty variable of the other product(s) in the same consideration set. This is reasonable, because as customers are more loyal to other products, they are less likely to buy this product.

4.4 Conclusion

A two-stage prediction model is developed and tested on two real datasets. Specifically, three prediction weighting schemes are introduced, and compared with the baseline. In all cases, these weighting schemes outperform the baseline by segmenting the data into groups with similar characteristics. The hard weighting scheme uses only one classifier to predict, and soft weighting schemes use a combination of different classifiers to predict. The difference between Soft (A) and (B) is that the weight of each classifier is different. In Soft (A), the weight of each classifier is proportional to the size of the cluster the classifier belongs to. In Soft (B), the weight is proportional to the cardinality of the cluster. The experiments show that

Table 4.14: Coefficients comparison

VARIABLE	CLASSIFIER A	CLASSIFIER B	CLASSIFIER C	
	{1, 9}: P_1	{2, 9}: P_2	{2, 6, 9}: P_2	{2, 6, 9}: P_6
PRICE 1	-0.407	N/A	N/A	N/A
PRICE 2	N/A	-1.0132	-0.7486	-0.1718
PRICE 6	N/A	N/A	0.2912	0.2691
PRICE 9	-0.2294	0.1876	0.367	0.4659
DISPLAY 1	1.196	N/A	N/A	N/A
DISPLAY 2	N/A	0.3652	-0.0547	0.2041
DISPLAY 6	N/A	N/A	0.6141	3.1068
DISPLAY 9	-2.0935	-0.5309	-0.3713	-0.1212
LOYALTY 1	0.3188	N/A	N/A	N/A
LOYALTY 2	N/A	0.1181	0.2186	-0.0322
LOYALTY 6	N/A	N/A	-0.0901	0.231
LOYALTY 9	-0.1904	-0.1826	-0.1228	-0.1413
INTERCEPT	3.6201	4.7803	0.4215	-2.7977

0: ALL 1: B-3/F-S 2: CH 3: DASH 4: FAB
5: FRESH 6: OXYDOL 7: PUREX 8: SURF 9: TIDE

the soft weighting schemes always show higher accuracy than the hard weighting scheme.

One advantage of this two-stage model is that it helps to identify the consideration sets that are easy to predict. The other advantage is that it helps to capture the different characteristics of customers in different consideration sets.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

We propose a novel heuristic subset clustering algorithm inspired by the analysis of consideration sets. Our main contribution is that the cluster construction considers the special relationship among itemsets that can be clustered together. We also developed an evaluation method to reflect how the extracted clusters match known ones. The experiments on simulated data show that our algorithm can effectively identify the known clusters for both dense and sparse datasets.

We then use the resulting market segmentation as a preprocessing step for choice prediction [21]. A specific classifier is built for each cluster. The resulting simplified choice models identify easy-to-predict segments, and improves the overall accuracy in all cases. It also helps us to understand the characteristics of different groups of customers.

Several algorithm modifications are planned. Alternate quality measurements may help us better locate candidate cluster centers, especially for sparse datasets. Secondly, our algorithm could be generalized to use counts instead of binary variables to represent purchase history. Among people who bought products A, B and C, if people buy products A and B much more often than C, then A and B could have more weight for this consideration set. Third, other soft weighting schemes could be developed. One easy choice could be using the same weight for all the classifiers a data point belongs to.

We solved the problem of finding an optimal match between two sets of subsets with a greedy search method. This problem may correspond to an extension of the *maximum weight bipartite matching* problem [96], and as such, may have a polynomial-time algorithm for finding an exact solution. In future work, we will

determine whether such a reduction is indeed possible. In the meantime, the greedy solution serves as a lower bound on the score of the globally optimal match, and still fairly compares one algorithm to another.

Our algorithm is a general machine learning technique that is applicable to other fields. For example, in nursing, there are some standard references that describe which interventions should be given to a patient depending on his/her nursing diagnosis. In practice, a nurse may or may not give patients interventions according to the book. We will compare the interventions that are frequently given to patients together with those that appear together in the reference books, paying special attention to the discrepancies between the two.

REFERENCES

[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. *In Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'04)*, pages 12–19, 2004.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, 1998.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93)*, pages 207–216, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, 1994.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[6] 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.

[7] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[8] R. Andrews and T. Srinivasan. Studying consideration effects in empirical choice models using scanner pannel data. *Journal of Marketing Research*, 32:30–41, 1995.

[9] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. LIMBO: Scalable clustering of categorical data. *In 9th Int. Conf. on Extending DataBase Technoloy*, pages 123–146, March 2004.

[10] M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. *In Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 49–60, June 1999.

[11] D. Barbara, Y. Li, and J. Couto. Coolcat: An entropy-based algorithm for categorical clustering. *ACM Press*, pages 582–589, 2003.

[12] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, pages 36(1–2): 105–139, 1999.

[13] R. Bayardo. Efficiently mining long patterns form databases. *In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 85–93, 1998.

[14] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. Wiley-Interscience, 2004.

[15] B. Ben-Akiva and B. Boccara. Discrete choice models with latent choice sets. *International Journal of Research in Marketing*, 12:9–24, 1995.

[16] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[17] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[18] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. CA: Wadsworth International, 1984.

[19] B. J. Bronnenberg and W. Vanhonacker. Limited choice sets, local price response, and implied measures of price competition. *Journal of Marketing Research*, 33:163–173, 1996.

[20] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *In Proceedings of the 17th International Conference on Data Engineering*, 2001.

[21] 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.

[22] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AUTOCLASS: A Bayesian classification system. *In Proc. 1988 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'88)*, pages 54–64, 1988.

[23] A. Chernev. *Essential Marketing Concepts and Frameworks*. Brightstar Media, Inc., 2006.

[24] J. Chiang, C. Siddartha, and N. Chakravarthi. Markov chain monte carlo and models of consideration set and parameter heterogeneity. *Journal of Econometrics*, 89:223–48, 1999.

[25] D. Cristofor and D. Simovici. An information-theoretical approach to clustering categorical databases using genetic algorithms. In *2nd SIAM ICDM, Workshop on Clustering High Dimensional Data*, 2002.

[26] W. S. DeSarbo and W. L. Cron. A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, 5:249–282, 1988.

[27] N. K. Dhalla and W. H. Mahatoo. Expanding the scope of segmentation research. *Journal of Marketing*, pages 34–41, April 1976.

[28] I. Dhillon, S. Mallela, and S. Modha. Information-theoretic co-clustering. *In Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'03)*, pages 89–98, 2003.

[29] A. S. Dick and K. Basu. Customer loyalty: Toward an integrated conceptual framework. *Journal of the Academy of Marketing Science*, 22:99–113, 1994.

[30] S. Dolniar and F. Leisch. Behavioral market segmentation of binary guest survey data with bagged clustering. *Proc. International Conf. on Artificial Neural Networks*, pages 111–118, 2001.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley & Sons, second edition, 2001.

[32] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Nat. Acad. Sci.*, 95:14863–14868, 1998.

[33] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. *In Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Aug. 1996.

[34] D. Fisher. Improving inference through conceptual clustering. *In Proceedings of 1987 AAAI Conference*, pages 461–465, July 1987.

[35] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - Clustering categorical data using summaries. *Knowledge Discovery and Data Mining*, pages 73–83, 1999.

[36] D. S. Geert and W. S. DeSarbo. A latent class probit model for analyzing pick any/n data. *Journal of Classification*, 8:45–63, 1991.

[37] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Nat. Acad. Sci.*, 97:12079–12084, 2000.

[38] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. *VLDB Journal: Very Large Data Bases*, 8(3–4):222–236, 2000.

[39] K. Gouda and M. J. Zaki. Fast algorithms for mining association rules in large databases. *In Proc. of the IEEE Int. Conference on Data Mining*, pages 163–170, 2001.

[40] G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. *In Proc. 2000 Int. Conf. Data Engineering(ICDE'00)*, pages 512–521, 2000.

[41] J. L. Gross and J. Yellen. *Graph Theory and its Application*. International Series in Quantitative Marketing. Boston: Kluwer Academic Publishers., second edition, 2000.

[42] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[43] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 73–84, June 1998.

[44] D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. *In Proc. 1997 ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS'97)*, pages 209–216, 1997.

[45] L. C. Hamilton. *Regression with Graphics: A Second Course in Applied Statistics*. Brooks/Cole, 1992.

[46] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, 2001.

[47] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 1–12. ACM Press, May 2000.

[48] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. *In Proc. 2002 Int. Conf. on Data Mining(ICDM'02)*, pages 211–218, 2002.

[49] J. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[50] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[51] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. *In Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 58–65, Aug. 1998.

[52] Z. Huang. Extensions to the k-means algorithm for clustering large data set with categorical values. *Data Mining and Knowledge Discovery*, 2:284–304, 1998.

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

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

[55] J. R. Jensen. *Introductory digital image processing: A remote sensing perspective*. Prentice Hall, second edition, 1996.

[56] G. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. *In Proc. 11th Conf. Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[57] M. A. Jones, D. L. Mothersbaugh, and S. E. Betty. Why customers stay: measuring the underlying dimensions of services switching costs and managing their differential strategic outcomes. *Journal of Business Research*, 55:441–450, 2002.

[58] W. Kamakura and G. Russell. A probablistic choice model for market segmentation and elasticity structure. *Journal of Marketing Research*, 26:379–90, 1989.

[59] G. Karypis, E. H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *Computer*, 32:68–75, 1999.

[60] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data*. John Wiley, 1990.

[61] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, pages 43:59–69, 1982.

[62] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[63] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[64] T. Li. A general model for clustering binary data. *In Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'05)*, pages 188–197, 2005.

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

[66] T. Li and S. Zhu. On clustering binary data. *Proc. of the 2005 SIAM Int. Conf. On Data Mining(SDM'05)*, pages 526–530, 2005.

[67] L. Lovasz and M. D. Plummer. *Matching Theory*. New York, 1986.

[68] T. Lwin and P. J. Martin. Probit of mixtures. *Biometrics*, 45:721–732, 1989.

[69] R. Maclin. Boosting classifiers regionally. In *In Proceedings of 1998 AAAI Conference*, pages 700–705, 1998.

[70] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Statistics and Probability*, 1:281–298, 1967.

[71] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural computation*, 1:281–294, 1989.

[72] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. *In Proc. Int. Conf. Very Large Data Bases(VLDB'94)*, pages 144–155, 1994.

[73] C. Ordonez. Clustering binary data streams with k-means. *Proc. of the 8th ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 12–19, 2003.

[74] C. Ordonez, E. Omiecinski, and N. Ezquerra. A fast algorithm to cluster high dimensional basket data. In *Int. Conf. on Data Mining (ICDM'02)*, pages 633–636, 2001.

[75] J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *In Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95)*, pages 175–186, 1995.

[76] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *In Proc. of 7th Int. Conf. on Database Theory (ICDT'99)*, pages 398–416, 1999.

[77] J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. *In Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 378–385, 2002.

[78] Y. Peng, G. Kou, Y. Shi, and Z. Chen. Improving clustering analysis for credit card accounts classification. *In Proc. Int. Conf. on Computational Science*, 3:548–553, 2005.

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

[80] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1984.

[81] R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Manteo, CA, 1993.

[82] J. Roberts and J. Lattin. Development and testing of a model of consideration set composition. *Journal of Marketing Research*, 28:429–440, 1991.

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

[84] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. *Parallel Data Processing*, 1:318–362, 1986.

[85] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. *In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 343–354, 1998.

[86] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. *In Proc. 1995 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 432–443, 1995.

[87] S. Siddarth, R. E. Bucklin, and D. G. Morrison. Making the cut: Modeling and analyzing choice set restriction in scanner panel data. *Journal of Marketing Research*, 32:255–266, 1995.

[88] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. *In Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, pages 208–215, 2000.

[89] H. Spath. Algorithm 39: Clusterwise linear regression. *Computing*, 22:367–373, 1979.

[90] P. N. Tang, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

[91] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[92] B. Vroomen, P. H. Franses, and E. V. Nierop. Modeling consideration sets and brand choice using artificial neural networks, 2001. ERIM Report Series ERS-2001-10-MKT.

[93] M. Wedel and W. S. DeSarbo. A latent class binomial logit methodology for the analysis of paired comparison data: An application reinvestigating the determinants of perceived risk. *Decision Sciences*, 24:1157–1170, 1993.

[94] M. Wedel and W. S. DeSarbo. A mixture likelihood approach for generalized linear models. *Journal of Classification*, 12:21–55, 1995.

[95] M. Wedel and W. Kamakura. *Market Segmentation: Conceptual and Methodological Foundations*. CRC Press LLC, second edition, 1998.

[96] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[97] P. Willet. Recent trends in hierachical document clustering: A critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[98] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 1999.

[99] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. *In Proc. 31th Int. Conf. Very Large Data Bases(VLDB'05)*, pages 709–720, 2005.

[100] W. Xu and Y. Gong. Document clustering by concept factorization. *In Proc. of the 27rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*, pages 202–209, 2004.

[101] D. Yuan and W. N. Street. ICC: A new machine learning technique for identifying response-specific subpopulations. *Working Paper*, 2006.

[102] D. Yuan and W. N. Street. HACS: heuristic algorithm for clustering subsets. *Proc. of the 2007 SIAM Int. Conf. On Data Mining(SDM'07)*, 2007.

[103] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining.

[104] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. *In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, June 1996.

[105] Y. Zhang, A. Fu, C. Cai, and P. Heng. Clustering categorical data. *In Proc. of the 16th Int. Conf. on Data Engineering (ICDE'00)*, page San Deigo, 2000.