Conceptual Clustering Categorical Data with Uncertainty

Yuni Xia
Indiana University – Purdue University Indianapolis
Indianapolis, IN 46202, USA
yxia@cs.iupui.edu

Bowei Xi
Purdue University
West Lafayette, IN 47906, USA
xbw@stat.purdue.edu

Abstract

Many real datasets have uncertain categorical attribute values that are only approximately measured or imputed. Uncertainty in categorical data is commonplace in many applications, including biological annotation, medial diagnosis and automatic error detection. In such domains, the exact value of an attribute is often unknown, but may be estimated from a number of reasonable alternatives. Current conceptual clustering algorithms do not provide a convenient means for handling this type of uncertainty. In this paper we extend traditional conceptual clustering algorithm to explicitly handle uncertainty in data values. In this paper we propose new total utility (TU) index for measuring the quality of the clustering. And we develop improved algorithms for efficiently clustering uncertain categorical data, based on the COBWEB conceptual clustering algorithm. Experimental results using real datasets demonstrate how these algorithms and new TU measure can effectively improve the performance of clustering through the use of internal probabilistic information.

1. Introduction

In many applications, data contains inherent uncertainty. A number of factors will contribute to the uncertainty, such as the random nature of the physical data generation and collection process, measurement and decision errors, and data staling.

One example is protein database. Along with other information about various proteins, it is important to understand whether the protein is ordered or not – the existence of secondary structure. This type of information is typically obtained by literature mining – examining the experiments, or the features of similar or closely related protein datasets. However the literature mining results will introduce uncertainty to whether a protein may be marked as either ordered or not. Notice this is a categorical attribute with two levels.

In the mean time, data uncertainty often arises in automatic data integration. For example deep web data in the form of dynamic HTML pages can be used to generate related datasets. This is a challenging problem. Often the mapping from information in a web page to a set of attributes is unclear. It may be known that a page contains prices for several items and a set of numeric values. It is difficult for a program to determine which numerical value is the price for a given item with accuracy. Instead, existing algorithms will generate multiple candidates for the value of an attribute, each with a likelihood or probability of being the correct one. Similar issues arise in the domain of integrating unstructured text information with structured databases, such as automatic annotation of customer relationship management (CRM) databases, and email search databases [SMPSH07].

Uncertainty is prevalent in many application domains. Although much research effort has been directed towards the management of data with uncertainty in databases, few have addressed the issue of mining data with uncertainty. It is well known fact that data mining results will respond to the subtle errors or uncertainty in the data. The problem of inaccurate data has continuously been a challenge for many data mining applications.

We suggest incorporating information of uncertainties, such as the probability distribution of attributes, into existing data mining methods. In this paper we will study how such information can be incorporated in data mining by using clustering as a motivating example. In particular, we will study one of the most popular conceptual clustering algorithms – COBWEB.

This paper will focus on the problem of clustering categorical data with uncertainty, where candidate values for a certain attribute will be assigned probabilities. We propose new solutions as well as a new measure, Probability Utility (PU), for evaluating the quality of clustering. The new techniques are

1082-3409/07 $25.00 © 2007 IEEE
DOI 10.1109/ICTAI.2007.135

shown to provide efficient clustering results through experimental validation with real life data.

This paper is organized as follows. In the next section, we will discuss related work on data clustering and mining data with uncertainty. Section 3 discusses the conventional COBWEB conceptual clustering techniques. Section 4 will show the model for categorical data with uncertainty. In section 5, we will discuss how to extend the conventional COBWEB algorithm for clustering data with uncertainty. We will show the experimental results in section 6. Section 7 summarizes the paper.

2. Related Work

Clustering is one of the most studied areas in data mining research. Many clustering algorithms have been proposed in the literature, such as hierarchical clustering, partitioning clustering, density-based clustering, grid-based clustering, and conceptual clustering. Hierarchical clustering algorithms are either agglomerative ("bottom-up") or divisive ("top-down"). They find successive clusters using previously established ones. Whereas partitioning algorithms, such as K-Means and K-Medoids, determine all clusters at once. Density-based clustering, such as DBSCAN and OPTICS, typically regards clusters as dense regions of objects in the data space that are separated by regions with low data density. Grid-based methods, such as STING, quantize the space into finite number of cells to form a grid structure, on which all of the operations for clustering are performed. Conceptual clustering produces a classification scheme over the objects, and it goes one step further than other clustering algorithms by finding characteristic descriptions of each group. Hence each group represents a concept or a class.

In spite of the numerous clustering algorithms, how to handle missing data and data with uncertainty has remained a great challenge. One related research area is fuzzy clustering, which has been carefully studied in fuzzy logic [Ruspin69]. In fuzzy clustering, a cluster is represented as a fuzzy subset of objects. Each object has a "degree of belongingness" for each cluster. In other words, an object can belong to more than one cluster, each cluster with a different degree. The fuzzy c-means algorithm is one of the most widely used fuzzy clustering methods [Dunn73]. Different fuzzy clustering methods have been applied to regular or fuzzy data [SSJ97]. While their work focused on creating fuzzy clusters (i.e., each object can belong to more than one cluster with different degrees), our work aim at hard clustering, based on an uncertainty model of objects. The result is that each object can only belong to one cluster.

Recently there have been studies on partition-based and density-based clustering of data with uncertainty. The UK-means algorithm, [CCKN06] and [NKCCCY06], is based on the K-Means clustering algorithm. The expected distance between objects is computed using a probability distribution function. The FDBSCAN and FOPTICS algorithms, [KPKDD05] and [KPICDM05], are based on DBSCAN and OPTICS respectively. Instead of identifying regions with high data density, these algorithms identify regions with high expected density, based on the probability models of the objects. Our work differs from the previous ones in that we focus on conceptual clustering methods and our algorithms are able to handle uncertainty in categorical attributes.

3. COBWEB Conceptual Clustering

Conceptual clustering is a machine-learning paradigm for clustering. It is different than other clustering algorithms in that it generates a concept descriptor for each cluster. COBWEB [DHFisher87] is one of the mostly commonly used algorithms for conceptual clustering. In this paper, we extend the COBWEB algorithm for clustering data with uncertainty.

Whereas some iterative distance-based clustering algorithms, such as K-Means, go over the whole dataset until convergence occurs, COBWEB works incrementally, updating the clusters object by object. The clusters COBWEB creates are formed into a tree. The leaves of the tree represent every individual concept; the root node represents the whole dataset; and the branches represent the hierarchical clusters within the dataset. The total number of clusters can be up to by the size of the dataset.

The COBWEB data structure is a tree wherein each node represents a certain *concept*. Each concept is associated with a set, a multi-set, or a bag of objects. Each object is assigned binary-valued indicators on a property list. The data associated with each concept are the integer counts for the objects belonging to that concept.

Please refer to Figure 1 as an example. Let a concept C_1 contain the following three objects (repeated objects being permitted).

1. [1 0 1]
2. [0 1 1]
3. [1 1 1]

The three properties are: [is_male, has_wings, is_nocturnal]. Then what is stored at this concept node is the property count [2 2 3], indicating that 2 of the objects in the concept is male, 2 of the objects have wings, and 3 of the objects are nocturnal. The concept *descriptor* is the concept-conditional probabilities of

the properties at the node. Thus, given that an object is a member of a concept C_1, the probability that it is male is 2 / 3. Likewise, the probability that the object has wings is 2/3 and probability that the object is nocturnal is 3 / 3. The concept descriptor can therefore simply be given as [2/3, 2/3, 3/3], which corresponds to the C_1-conditional feature probability, i.e., $p(x \mid C_1)$ = (2/3, 2/3, 3/3).

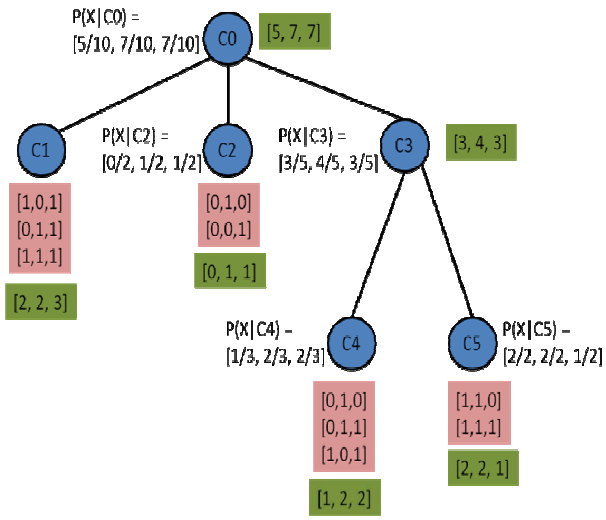

Figure 1: Classification Tree

Figure 1 shows a tree with five concepts. C_0 is the root concept, which contains all ten objects in the data set. Concepts C_1, C_2 and C_3 are the children of C_0. C_1 contains three objects, C_2 contains two objects, and C_3 contains five objects. Concept C_3 is also the parent of concepts C_4 and C_5, which contain three and two objects respectively. Note that each parent node (relative super-ordinate concept) contains all the objects of its children nodes (relative sub-ordinate concepts). For each leave node, the first box underneath it lists the actual objects, and the second box lists the attribute counts. For each internal node, the box on its right lists the attribute counts for all its children nodes. In Fisher's (1987) description of COBWEB, he mentioned that only the total attribute counts, without the conditional probabilities or the actual object lists, should be stored at the nodes. Any probabilities can be computed from the attribute counts when needed.

COBWEB starts with a tree consisting of just the root node. From there, instances are added one by one, with the tree being updated accordingly at each stage. When an instance is added, there are four possible actions. One will choose the action with the biggest *category utility*. The Category Utility (CU) is defined by the following function:

$$\frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2]}{n}$$

V_{ij} is a potential value of attribute A_i. q is the number of nodes, concepts or categories forming a partition $\{C_1, C_2, \ldots, C_q\}$ at a given level of the tree. Category Utility is the increased amount of the expected number of attribute values that can be correctly estimated from a partition. This expected number is $P(C_k)[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2$. And the expected number of correct estimates without such knowledge is the term $\sum_i \sum_j P(A_i = V_{ij})^2$. Category Utility rewards intra-class similarity and inter-class dissimilarity where:

- Intra-class similarity is the probability $P(A_i = V_{ij} \mid C_k)$. The larger this value is, the greater the proportion of class members that share this attribute-value pair will be. Hence the class members are more predictable.
- Inter-class dissimilarity is the probability $P(C_k \mid A_i = V_{ij})$. The larger this value is, the fewer the objects in contrasting classes will share this attribute-value pair. It is more likely that the pair belongs to a certain class.

4. A Model for Categorical Data with Uncertainty

In this section, we will introduce a general model for categorical data with uncertainty. Then in the next section, we will discuss how to cluster data with uncertainty based on this model.

Under the uncertainty model, a dataset can have attributes that are allowed to take uncertain values. The focus of this paper is on attributes with uncertain values that come from categorical domains. Such an attribute is called an uncertain *categorical attribute* (UCA), denoted by u.

u is an attribute in relation R which is uncertain. u takes values from the categorical domain $\mathcal{D}$ with cardinality $|\mathcal{D}| = N$. Within a regular relation with the correct value, the value of an attribute a is a single value d_k in $\mathcal{D}$, $\Pr(a = d_k)=1$. In the case of an uncertain relation, we record the information by a probability distribution over $\mathcal{D}$ instead of a single value. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, then we write T_a as the probability distribution $\Pr(u = d_i)$ for all values of i in $\{1, \ldots, N\}$. Thus, T_a can be represented by a probability vector $T_a = (p_1, p_2, \ldots, p_N)$ such that $\sum_{i=1}^{N} p_i = 1$. In many cases, the probability vector is sparse and most of the value are zeros. In such cases, we may write T_a as a set of potential value and its corresponding probability pairs,

T_a={(d,p): Pr(T_a=d)=p and p≠0}. Hereafter we write a UCA by *u* instead of T_a unless noted otherwise. Also, we write Pr(u = d_i) simply as p_i.

Table 1: Example of Data with uncertainty

Age	Gender	…	Tumor
10-20	F		(Benign, 0.8), (Malignant, 0.2)
60-70	M		(Benign, 0.9), (Malignant, 0.1)
70-80	M		(Benign, 0.3), (Malignant, 0.7)
40-50	M		(Benign, 0.2), (Malignant, 0.8)

Table 1 shows an example. It is for a medical diagnosis application with an UCA attribute. The table stores information for patients with tumor. The type of tumor is an UCA attribute, whose value cannot be determined exactly. It may be either benign or malignant, each associated with a probability.

Previously, [SMPSH07] defined UCA as follows:

Definition 1 *Given a categorical domain* $\mathcal{D}$ ={d_1, .., d_N}, *an uncertain discrete attribute (UCA)* u *is characterized by probability distribution over* $\mathcal{D}$. *It can be represented by the probability vector* $\boldsymbol{P}$ = (p_1, ..., p_N) *such that* Pr(u = d_i) = p_i.

Assuming *n* is the totally number of attributes and *m* is the maximal number of candidate values for the attributes, under the definition of UCA, we can record a general dataset with uncertainty using a *n*m* matrix as follows:

$$\begin{pmatrix} p_{11}, p_{12}, p_{13}, \ldots, p_{1m} \\ p_{21}, p_{22}, p_{23}, \ldots, p_{2m} \\ \ldots \\ p_{n1}, p_{n2}, p_{n3}, \ldots p_{nm} \end{pmatrix}$$

p_{ij} is the probability of A_i equal to the value V_{ij}. If an attribute A_i has only *k* candidate values, *k*<*m*, then for all *k*<*l*<*m*, p_{il} = 0.

Dataset without uncertainty can be treated as a special case of data with uncertainty. When using a matrix to represent a data record without uncertainty, there is only one element per row to be non-zero. The value of the non-zero element is one, which means that the value for each attribute is certain – the probability that the attribute equals to such a value is 100%.

5. Conceptual Clustering with Uncertainty

In this section, we will discuss how to extend the COBWEB algorithm for conceptual clustering categorical data with uncertainty. We propose three solutions, a naïve solution, an extended COBWEB solution and a Total Utility (TU) based solution. We will explain the three solutions in detail next.

5.1. Naïve Solution

For categorical data with uncertainty, a naïve solution is to pick one of the most likely candidate values for each attribute represented by probability distribution. For example, with the data shown in table 1, the tumor attribute is an uncertain one, with possible values to be either malignant or benign with certain probability. The naïve solution will set the attribute to be the type of tumor with higher probability for each data record. That is, for record 1 and 2, the tumor will be benign and for record 3 and 4, the tumor is malignant. Therefore the uncertainty within a dataset disappears, and the traditional COBWEB algorithm can be applied on the dataset without modification.

While this naïve approach is simple to implement, it will result in significant loss of information and lower quality of data clustering. Hence we need alternative approaches that can process the uncertainty of the attribute values directly.

5.2. Extended COBWEB

For traditional COBWEB, the category utility is computed as:

$$\frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2]}{n}$$

Specifically, the intra-class similarity is defined as $P(A_i = V_{ij} \mid C_k)$ and inter-class dissimilarity is defined as $P(C_k \mid A_i = V_{ij})$.

When we have the exact value of every attribute in a dataset, the intra-class similarity $P(A_i = V_{ij} \mid C_k)$ is calculated as: $|\{O : O \in C_k \,\&\, A_i = V_{ij}\}| / |C_k|$, which is the cardinality of objects equal to value V_{ij} for attribute A_i in C_k divided by the overall cardinally of C_k. As shown in Figure 2 (a), the number of objects in C_1 is 3, thus the cardinally of C_1 is 3. Out of these three objects, two of them has the first attribute A_0 equal to one, therefore, $P(A_0 = 1 \mid C_k) = 2/3$.

When data contains uncertain attributes, the way of calculating the intra-class similarity $P(A_i = V_{ij} \mid C_k)$ needs to be changed to: $\Sigma\, P_r(O : Ai = Vij) / |C_k|$ for all object *O* in C_k. Figure 2 (b) shows an example. Here the cardinality of C_2 is also 3. Assume each of them have only one attribute, which could be either 0 or 1. The first object is 1 with probability 0.9 and is 0 with probability 0.1, object 2 is 1 with probability 0.2 and 0 with probability 0.8, and object 3 is 1 with probability 0.7 and 0 with probability 0.3. Therefore, for all three objects, the sum of the probability of being 1 should be 0.9+0.2+0.7=1.8. The probability for class C_2 of being 1 is 1.8/3 = 60%. This is different from the naïve approach in section 5.1. If we use the naïve approach,

which pick one of the most likely candidate values for each uncertain attribute, then the three object in C_2 will be 1, 0, 1 and the probability for class C_2 of being 1 is 2/3.

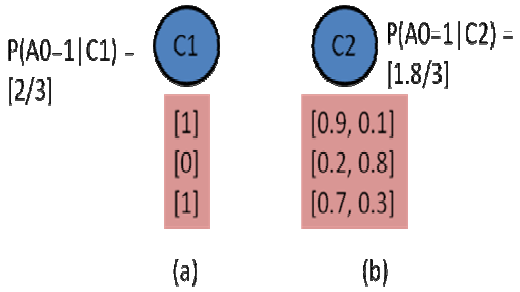

Figure 2: Data with uncertainty

5.2. Total Utility based COBWEB

5.2.1 Probability Utility

When data contains uncertain attributes, we should take uncertainty into consideration when measuring the quality of a cluster/concept. In other words, ideally data within one cluster should not only have a high similarity measure based on all attributes, but also have similar probability distributions for the uncertain attributes. We will explain the reason with an example shown in Figure 3.

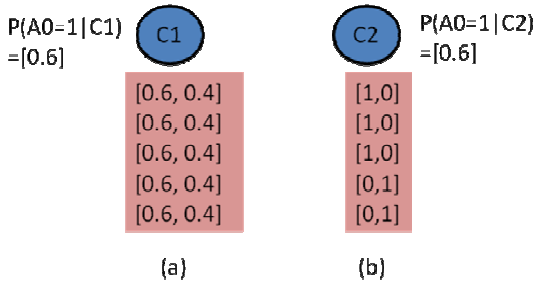

Figure 3: Data with uncertainty

Figure 3 shows two clusters, C_1 and C_2. Both C_1 and C_2 contain five records. For simplicity, assume each record has only one attribute, which describes the type of a tumor, and this attribute has two possible values, 0 or 1: 0 for benign and 1 for malignant. The five records in C_1 all has value 1 with probability 0.6, while for the five records in C_2, three of them have value 1 with probability 1 and the rest two of them has value 0 with probability 1. When using the category utility measure of the traditional COBWEB, both nodes has the same feature – the probability of being 1 is 0.6 on average. However, C_1 should be considered as a better cluster than C_2, since the data records within C_1 has higher intra-class similarity – not only all of them tend to have value 1, but also all with 60% probability. A researcher tends to find a cluster with five tumors, all of which may be malignant with the same 60% probability to be more interesting than a cluster with three malignant tumors and two benign ones.

Based on the above observation, we propose another heuristic measure, Probability Utility (PU), to guide the search and clustering. The Probability Utility is defined as:

$$\frac{-\sum_{k=1}^{n} P(C_k)\left[\sum_i \sum_j \sigma^2_{P(A_i=V_{ij}|C_k)^2} - \sum_i \sum_j \sigma^2_{P(A_i=V_{ij})^2}\right]}{n}$$

Probability Utility can be viewed as a function that rewards the similarity of the probability distributions of objects within the same class and dissimilarity of the probability distributions of objects between different classes. In particular, probability utility is a tradeoff between intra-class probability similarity and inter-class probability dissimilarity of objects. Intra-class similarity is reflected by the term: "$-(\sigma_{P(Ai = Vij|Ck)})^2$". This is a non-positive value: negative one times a variance. A large value of this term means a small variance $(\sigma_{P(Ai = Vij|Ck)})^2$. Hence a big proportion of the class members will share identical or similar probability distributions. Inter-class similarity is represented by the term $(\sigma_{P(Ai = Vij)})^2$, the variation of unconditional probabilities. When this term is large, there will be fewer objects in contrasting classes that share the same or similar probabilities. Therefore each cluster will be more predictive.

Please note that Probability Utility works best when uncertainty naturally arises instead of by errors such as measurement imprecision. The reason is that only when uncertainty is part of the nature of the data itself, it can work as a similarity measure. If it is caused by external factors such as measurement imprecision, then the similarity derived from the uncertainty distributions probably indicates the similarity of measuring techniques or surrounding environments. It will have nothing to do with the data itself.

We further define the Total Utility (TU) index for guiding the clustering. Total Utility is a balance between the Category Utility and the Probability Utility: $TU = \alpha CU + (1- \alpha)\ PU$, where $0<=\alpha<=1$. When $\alpha=1$, this is the same as regular COBWEB. As α becomes smaller, higher weight is given to the similarity of probability distributions within the same cluster.

5.2.2 TU based COBWEB Algorithm

Since Probability Utility can be used to guide the clustering process, each node should store not only the probabilities $P(A_i = V_{ij} |C_k)$, but also the variance of

the probabilities $\sigma^2_{P(Ai = Vij|Ck)}$. This is for computing the Total Utility of each node.

Same as traditional COBWEB algorithm, at each node, four possible operations – insert, create, merge and split – will be considered and the one that yields the highest Total Utility value will be selected.

Given a new object, TU based COBWEB descends the tree along an appropriate path, updating counts along the way, in search of the best node to place the object. The decision is based on temporarily adding the object in one node and computing the Total Utility of the resulting partition. The placement that results in the highest Total Utility should be a good candidate host for the object. COBWEB also computes the Total Utility of a new node that is created entirely for the object. Then we will compare the Total Utilities from inserting the object in existing nodes and the Total Utility of the new node. The object is then placed according to the higher value of the two options.

The results of two operations – insert and create – are highly sensitive to the input order, same as the conventional COBWEB. We then use two additional operators that help to make it less sensitive to the input order, by merging and splitting the nodes. When an object is processed, the two best hosts are considered for merging into a single node. Furthermore, we consider splitting the children nodes of the best host among all other existing nodes. These decisions are all based on Total Utility. The 'merge' and 'split' operators will allow COBWEB to perform a bidirectional search. For example, a merge can reverse a previous split.

All of the four operations – split, merge, create and insert – must take uncertainty gain or loss into account. A large increase in uncertainty in a node should be treated as a penalty and we must avoid it. This is automatically being taken care of by using the Total Utility measure and by adjusting the value of α in TU. The algorithm for TU based COBWEB is described as follows:

```
Cobweb(N: Node, I:Instance)
If N is a terminal node,
Then Create-new-terminals(N, I)
     UpdateNodeProbability(N,I).
Else.
   UpdateNodeProbability(N,I).
   For each child C of node N,
   Compute the utility for placing I in C.
   N1 := the node with the highest utility U1.
   N2 := the node with the second highest U2.
   UNew := the utility for creating a new node for I
   UMerge := the utility for merging N1 and N2
   USplit := the utility for splitting N1
   UMax := Max(U1, UNew, UMerge, USplit),
  If U1 == UMax  //insert
      Then Cobweb(N1, I) (place I in category N1).
   Else if UNew == UMax,  //create
         Then Nnew = new Node(I)
          Else if UMerge == UMax  //merge
                Then  NMerge :=  Merge(N1, N2, N).
                      Cobweb(NMerge, I).
               Else if NSplit == UMax  //split
                     Then Split(N1, N).
                          Cobweb(N, I).

UpdateNodeProabability(N: Node, I:Instance)
   Update the probability and the variance of probability of
category N.
   For each attribute A in instance I,
      For each value V of A,
         Update the probability of V and the variance of
probability given category N.

Merge(N1, N2, N)
  Make O a new child of N.
  Compute O's probabilities
  Compute O's variance
  Remove N1 and N2 as children of node N.
  Add N1 and N2 as children of node O.
  Return O.

Split(P, N)
  Remove the child P of node N.
  Promote the children of P to be children of N.
```

The algorithm is similar to conventional COBWEB. The major difference is in the merge operation. When merging node N_1 and N_2 into a new node O, the probabilities and the variance of the probabilities of the new node O should be computed as follows:

- The probabilities of the new node O are the weighted average of the probabilities of C_1 and C_2. As shown in Figure 4, if $P(A_0=1|C_1)$ = 1.6/2=0.8, and $P(A_0=1|C_2)$ = 1.2/2 = 0.6, by merging N_1 and N_2 into O, $P(A_0=1|O)$ should be (0.8*2+0.6*2)/(2+2) = 2.8/4 = 0.7.
- The variances of probabilities of the new node O can be computed based on the variance decomposition property or the law of total variance. According to this property, suppose the data is partitioned into subgroups. Then the variance of the whole group is equal to the mean of the variances of the subgroups plus the variance of the means of the subgroups. As shown in Figure 4, suppose that a group consists of a subgroup of C_1 and an equally large subgroup of C_2. Suppose that C_1 has a probability for $P(A_0=1|N_1)$ = 0.8 and the variance of the probabilities is 0.04, C_2 has a probability for $P(A_0=1|N_2)$ = 0.6 and the variance of the probabilities is 0.01, then the mean of the variances is (0.04 + 0.01) / 2 =

0.025; the variance of the means is the variance of 0.6, 0.8 which is 0.01. Therefore, for the merged node C_3, the variance of $P(A_0=1|C_3)$ will be 0.025 + 0.01 = 0.035. In a more general case, if the subgroups have unequal sizes, then they must be weighted proportionally to their size in the computations of the means and variances.

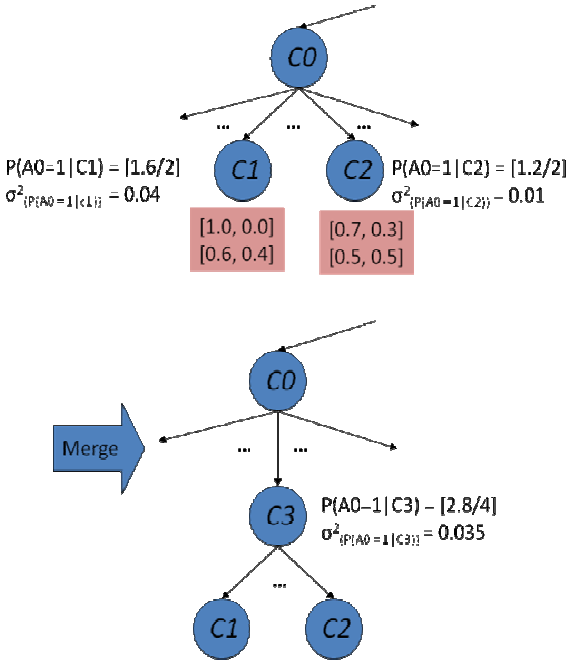

Figure 4: An example of node merge

6. Experiments

In this section, we will present the experimental results for clustering the categorical data with uncertainty. Our primary aim is to measure the effectiveness of the three methods in the presence of data uncertainty.

We used data records from a car evaluation dataset from the UCI machine-learning repository. The dataset contains six categorical attributes of cars: buying prices, maintenance price, doors, persons, luggage-boot and safety. Each attribute has a number of possible values. For example, the levels for buying price and maintenance price are 'very high', 'high', 'median' or 'low'. The possible values for safety are 'low', 'median' or 'high'. The true class label for each data object is one of the four: 'unacceptable', 'acceptable', 'good', and 'very-good'.

Due to lack of real uncertain data sets, we artificially introduce uncertainty into the dataset. For each experiment, we hide the value of an attribute for one third of the data records. For example, the exact value of the safety attribute is deliberately removed for part of the dataset. Therefore, the safety attribute is recorded with a probability distribution for some data instances. The probability distribution can be computed based on other attributes. Then the safety for a car instance may be 'low' with probability 0.8, 'median' with probability 0.1, and 'high' with probability 0.1.

We compare three clustering approaches discussed in section 5:

(1) The naïve approach
(2) The extended COBWEB
(3) The TU based COBWEB

We use the Rand Measure or Rand Index [WMRand71] to measure the quality of clustering. We compare the RIs between the sets of clusters created by the three approaches using data with uncertainty and the sets of clusters created by the original accurate data, that is, the data without uncertainty introduced. Ideally, the clustering results should be similar. A higher RI value indicates a higher degree of similarity between two sets of clusters. The Rand Index has a value between 0 and 1: 0 indicates that the two data clusters do not agree on any pair of points; 1 indicates that the data clusters are exactly the same.

Table 2. Experiment Results

	BuyPrice	MaintPrice	Safety
Naïve	0.628	0.623	0.612
Extended COBWEB	0.731	0.733	0.709
TU Based COBWEB	0.756	0.751	0.738

Table 2 shows the experiment results. The second column shows the RI index between our approaches using data with uncertainty in the buying price and the clustering results when the buying price is certain. When maintenance price is uncertain, the result is in the third column and the fourth column is the result when safety is uncertain. The extended COBWEB algorithm consistently showed a higher RI than the naïve approach. Furthermore, when the probability utility is taken as a measure for clustering, the performance is further improved. The results demonstrated that the extended COBWEB algorithm and TU based COBWEB can give a better prediction of the clusters that would be produced if the data uncertainty information is available and utilized.

7. Conclusions and Future Work

In this paper we present the extended COBWEB algorithm, which aims at improving the accuracy of clustering by considering the uncertainty associated with data. Although in this paper we only present the

COBWEB clustering algorithms for uncertain categorical data, the model can be easily generalized to uncertain numerical data using the algorithm proposed in [GLF89]. As future work, we would like to examine our approaches on more data sets, especially real uncertain data sets. We will also extend our model to other clustering approaches and other data mining techniques including outlier detection, classification and association mining.

8. Acknowledgement

We would like to thank reviewers for their valuable comments.

9. References

[GLF89] J. H. Gennari,P. Langley, D. Fisher, Models of incremental concept formation, Artificial Intelligence, vol. 40, pp. 11 – 61, 1989.

[DHFisher87] D. H. Fisher, Knowledge Acquisition via Incremental Conceptual Clustering, Machine Learning, vol 2, pp. 139-172, 1987.

[CCKN06] M. Chau, Reynold Cheng, B. Kao and J. Ng. Data with uncertainty Mining: An Example in Clustering Location Data. In the Methodologies for Knowledge Discovery and Data Mining, Pacific-Asia Conference (PAKDD 2006), Singapore, 2006.

[NKCCCY06] J. Ngai, B. Kao, C. Chui, R. Cheng, M. Chau and K. Yip. Efficient Clustering of data with uncertainty. In IEEE Intl. Conf. on Data Mining (ICDM), 2006.

[CCAggarwal07] C. C. Aggarwal. On Density Based Transforms for Data with uncertainty Mining. In IEEE Intl. Conf. on Data Engineering (ICDE), 2007.

[SMPSH07] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, S. Hambrusch, Indexing Categorical data with uncertainty In IEEE Intl. Conf. on Data Engineering (ICDE), 2007.

[WMRand71] W. M. Rand, Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66, pp846–850, 1971.

[KPKDD05] H.-P. Kriegel and M. Pfeifle. Density-based clustering of data with uncertainty. In Proc. of ACM SIGKDD Conference, 2005.

[KPICDM05] H.-P. Kriegel and M. Pfeifle. Hierarchical density-based clustering of data with uncertainty. In Proc. of IEEE ICDM Conference, 2005.

[Ruspin69] E. H. Ruspini. A new approach to clustering. Information Control, 15(1):22–32, 1969.

[Dunn73] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics, 3:32–57, 1973.

[SSJ97] M. Sato, Y. Sato, and L. Jain. Fuzzy Clustering Models and Applications. Physica-Verlag, Heidelberg, 1997.