

Privacy-Preserving Data Mining through Knowledge Model Sharing*

Patrick Sharkey, Hongwei Tian, Weining Zhang, and Shouhuai Xu

Department of Computer Science, University of Texas at San Antonio
{psharkey, htian, wzhang, shxu}@cs.utsa.edu

Abstract. Privacy-preserving data mining (PPDM) is an important topic to both industry and academia. In general there are two approaches to tackling PPDM, one is statistics-based and the other is crypto-based. The statistics-based approach has the advantage of being efficient enough to deal with large volume of datasets. The basic idea underlying this approach is to let the data owners publish some sanitized versions of their data (e.g., via perturbation, generalization, or ℓ -diversification), which are then used for extracting useful knowledge models such as decision trees. In this paper, we present a new method for statistics-based PPDM. Our method differs from the existing ones because it lets the data owners share with each other the knowledge models extracted from their own private datasets, rather than to let the data owners publish any of their own private datasets (not even in any sanitized form). The knowledge models derived from the individual datasets are used to generate some pseudo-data that are then used for extracting the desired "global" knowledge models. While instrumental, there are some technical subtleties that need be carefully addressed. Specifically, we propose an algorithm for generating pseudo-data according to paths of a decision tree, a method for adapting anonymity measures of datasets to measure the privacy of decision trees, and an algorithm that prunes a decision tree to satisfy a given anonymity requirement. Through an empirical study, we show that predictive models learned using our method are significantly more accurate than those learned using the existing ℓ -diversity method in both centralized and distributed environments with different types of datasets, predictive models, and utility measures.

1 Introduction

Personal data collected by government, business, service, and educational organizations is routinely explored with data mining tools. Although data mining is typically performed within a single organization (data source), new applications in healthcare, medical research, fraud detection, decision making, national security, etc., also need to explore data over multiple autonomous data sources. A major barrier to such a distributed data mining is the concern of privacy: data owners must balance the desire to share useful data and the need to protect private information within the data.

* This work was supported in part by NSF research grant IIS-0524612.

Two approaches of privacy-preserving data mining (PPDM) can be identified in the literature: one is crypto-based and the other is statistics-based. The crypto-based PPDM approach, such as [1,2], requires data owners to cooperatively execute specially designed data mining algorithms. These algorithms provide provable privacy protection and accurate data mining results, but often suffer performance and scalability issues. On the other hand, statistics-based PPDM approach, such as [3,4,5,6,7,8,9,10], lets each data owner release a sanitized dataset (e.g., via perturbation [3] or generalization [10]) to a third party, who can then execute data mining algorithms to explore the published data. Statistics-based methods have efficient implementations, making them more appropriate than crypto-based methods to deal with large volumes of data.

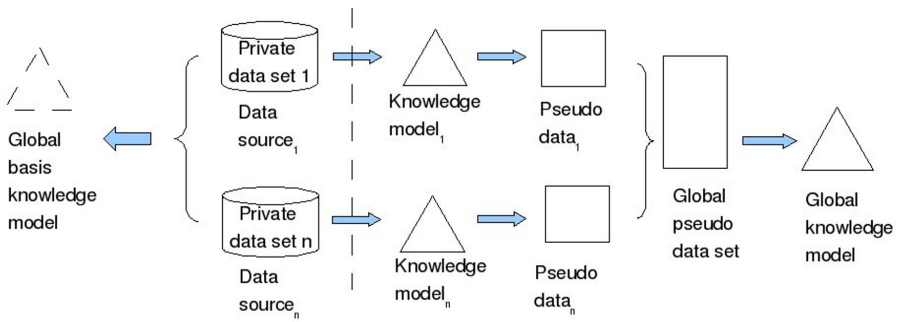


Fig. 1. Privacy-preserving data mining by knowledge model sharing: the goal is that the two resulting global knowledge models are (almost) the same

One problem of statistic-based data publishing methods is that they suffer loss of data quality, probably because they treat data sanitizing and data mining as unrelated tasks. For example, suppose a data miner wants to learn a decision tree classifier from the published data. Since decision tree learning algorithms select attributes for tree node split according to data distributions, in order for these algorithms to learn high quality decision trees from a published dataset, the sanitization methods should preserve the distributions of best splitting attributes. But, since data sanitizing and data mining are not considered together in a data publishing framework, the sanitization methods often indiscriminately alter data values, causing changes to the distributions of those important attributes, thus adversely affect the accuracy of classifiers learned from sanitized data. Even if there are opportunities to satisfy a privacy requirement and to preserve data quality by modifying data of some less important attributes, existing data publishing methods are unable to take advantage of those opportunities.

To address this problem, we introduce in this paper a new approach: *knowledge model sharing* (or simply model sharing), which lets each data source release a privacy-preserving local knowledge model learned from its private data, and let a data miner explore pseudo data generated from the local knowledge models. Specifically, as indicated in Figure 1, each data source learns a type of knowledge

model closely related to what a data miner wants to learn, using conventional data mining algorithms, and modifies the model according to a privacy requirement before releasing it to the data miner. The data miner uses local knowledge models to generate local pseudo datasets and combines them into a single dataset before applying conventional data mining algorithms to learn global knowledge models. In this paper, we focus on using predictive models as global knowledge models, with decision trees or classification rule sets as local knowledge models.

By not releasing any data (not even a sanitized version) and by modifying local knowledge models, model sharing can effectively protect privacy. In addition, as our study shows, this approach obtains much better quality data than data publishing methods do. Model sharing can be easily implemented since for example, data sources can easily offer data mining services [11,12] using widely available data mining tools and service-oriented computing techniques.

While model sharing is instrumental, a number of technical subtleties need to be carefully addressed. In this paper, we make the following specific contributions.

1. For data miners, we define the problem of generating pseudo-data from a predictive model and present an efficient heuristic algorithm that generates a pseudo dataset from a decision tree, according to its paths. This algorithm can be easily extended to generate pseudo datasets from disjoint classification rules.
2. To measure privacy of a predictive model, we show how to adapt privacy measures of data, such as k -anonymity and ℓ -diversity, to measure the privacy of decision trees whose class labels represent sensitive information. To our knowledge, privacy measure of knowledge models has not been proposed in the literature.
3. To protect private information contained in a predictive model, we present an algorithm for a data owner to prune a decision tree according to a given anonymity measure. This tree pruning technique can be viewed as a special form of generalization that preserves important patterns in the raw data. Used together with a good pseudo data generation method, this technique can result in higher data quality, compared to data publishing methods.
4. We perform an empirical study of the pseudo-data generation and tree pruning techniques. Specifically, we measure the utility of the pseudo data by the quality, such as classification accuracy, of global predictive models. For pseudo-data generation, we compare predictive models learned from global pseudo datasets with those learned directly from the local raw data. For tree pruning, we compare predictive models learned respectively from raw data, pseudo data, and ℓ -diversified data. In addition to the decision tree, we also study other types of global predictive models, such as Naive Bayes and conjunctive rules. Our results show that predictive models learned using our method are significantly more accurate than those learned using the ℓ -diversity method in both centralized and distributed environments with different types of datasets, predictive models, and utility measures.

We focus on data quality and limit the comparison of our tree pruning technique to the ℓ -diversity technique because, while data mining provides

a framework for comparing data utility of different privacy-preserving techniques, no metric currently exists for comparing privacy assurance of these techniques, making it impossible to fairly compare different privacy protection methods. We are able to compare pseudo data produced by tree pruning method with ℓ -diversified data only because the ℓ -diversity measure of data and the ℓ -diversity measure of decision trees are very similar. Finding a metric that can be used to compare privacy assurances of all the PPDM techniques is still an open problem.

The rest of this paper is organized as follows. In Section 2, we define the problem of pseudo-data generation from a decision tree and present a path-based pseudo-data generation algorithm for data miners. In Section 3, we present the l -diversity privacy measure for decision tree and an algorithm that prunes a decision tree to satisfy a given anonymity-based privacy requirement for data owners. In Section 4, we describe our empirical study and present experiment results. In Section 5, we discuss previous work related to our work presented in this paper. We draw conclusions in Section 6.

2 Pseudo-data Generation from Decision Trees

We consider n data sources whose private data tables (or datasets) form a horizontal partition¹. We assume that private data tables contain no personal identifier, such as name or identity number, but may contain duplicate tuples.

The goal of a data miner is to learn a global decision tree (see Figure 1) from a collection of local pseudo tables. In order to use this global decision tree to classify unseen tuples, local pseudo tables must preserve sufficient information contained in local private tables. One way to measure the quality of local pseudo tables is to compare the quality of the global decision tree to that of the global basis decision tree, learned directly from the (unavailable) collection of private tables (see Figure 1). Here, instead of seeking for high absolute quality, we are looking for high relative quality: the quality of the global (learned) decision tree should be as high as that of the global basis decision tree. Thus, one may want to formulate the problem as to find optimal local pseudo tables that guarantee identical quality of the global learned and basis decision trees. But unfortunately, this problem is ill-defined because in practice the global basis decision tree is not available to the data miner.

To overcome this difficulty, we focus on the problem of generating a high quality pseudo table for a single data source. Once this problem is solved, we can construct a high quality global pseudo table by combining high quality local pseudo tables. With this in mind, we define the pseudo-data generation problem as follows.

¹ In a distributed environment, private data tables of data sources can form a horizontal, vertical, or hybrid partition, where tables of a horizontal partition have a common set of attributes and those of a vertical partition have different sets of attributes.

Problem Statement: Find a pseudo dataset D based on a given basis decision tree h so that it minimizes the difference of classification accuracies, measured using any testing set, between h and the decision tree h' learned from D .

Notice that the difficulty involving multiple data sources is avoided here because both the basis decision tree h and the learned decision tree h' are available: h is published by the data source and h' is learned from a pseudo table.

This problem has at least one solution: the private table used to learn the basis tree h . Obviously, if a pseudo table is the same as the private table, the learned and the basis trees can have identical structures, and therefore, identical classification accuracies. However, we are more interested in other solutions to this problem (for an obvious reason).

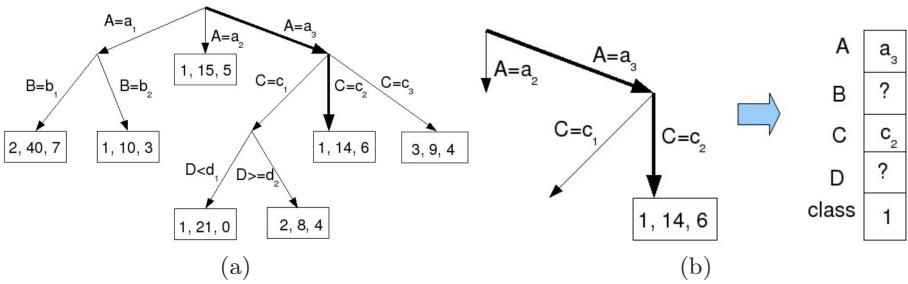


Fig. 2. A sample decision tree with leaf labeled by (class label, hit count, miss count)

Finding an optimal pseudo table from a given basis decision tree is difficult for several reasons. First of all, due to inherent randomness of decision tree algorithms (for example, they make random choices among several possible split attributes of identical class purity), different decision trees may be learned from the same dataset. Thus, even if a pseudo table is indeed the private table, there is no guarantee that the learned and the basis trees will have identical structures. Secondly, if two decision trees have different structures, it is difficult to determine if the difference of their classification accuracies is minimal.

In the rest of this section, we present a heuristic algorithm for a data miner to generate pseudo data from a decision tree. This algorithm uses information contained in a typical decision tree, as we shall describe next.

2.1 Decision Tree and Pseudo-data

A decision tree (see Figure 2(a) for an example) consists of labeled nodes and edges. The label of an edge is a predicate $A = a$ specifying that attribute A (of a tuple) has a value a , where a is a simple value if A is categorical and an interval if A is numeric. All the outgoing edges of a node are labeled by the same attribute and the values in these labels partition the domain of the attribute.

Each leaf node v is labeled by a class $v.class$, the majority (or plurality) class of training tuples assigned to the node during decision tree learning, and two counts: a hit count $v.hit$ representing the number of training tuples² in the majority class at this node, and a miss count $v.miss$ representing the number of misclassified training tuples (in other classes) at the node.

Each root-to-leaf path p of the tree has a path label $label(p)$, a class label $class(p)$, a hit count $hit(p)$ and a miss count $miss(p)$, where $label(p)$ is a conjunction of edge labels, the class label and the two counts are those of the leaf node of the path. In the path label, each attribute appears at most once and if $A = a$ appears, a is the most specific value of A in the entire path. We denote the set of attributes of path p by $attr(p)$, and the value of an attribute $A \in attr(p)$ by $value(A, p)$. For example, let p be the second leftmost path in the decision tree of Figure 2(a). Then, $label(p) = (A = a_1 \wedge B = b_2)$, $class(p) = 1$, $hit(p) = 10$, $miss(p) = 3$, $attr(p) = \{A, B\}$, and $value(A, p) = a_1$.

A decision tree can be used to predict the class of unseen tuples. If a tuple t satisfies a path p of a decision tree, that is, if each attribute of t satisfies the condition specified in the path label, or equivalently $\forall A \in attr(p) t[A] \in value(A, p)$, the class of t will be $class(p)$. In addition, a decision tree is also a descriptive model. In particular, the decision tree local to a data source can be viewed as an abstract representation of the local private table, providing a basis for generating a pseudo table.

By considering the learned the the basis decision trees that are structurally identical, we obtain following characterization of an important type of optimal pseudo tables.

Proposition 1. *Let D be a pseudo table generated from a given basis decision tree h , and h' be a decision tree learned from D . If h and h' have identical structure, that is $h = h'$, then every tuple in D satisfies a path of h . Furthermore, for each path $p \in h$, D contains exactly $hit(p) + miss(p)$ tuples that satisfy p .*

Proof. Since h and h' have identical structure, every path of h' is also a path of h , that is $\forall p(p \in paths(h') \iff p \in paths(h))$. According to the information contained in a decision tree, since D is the training set of h' , each tuple of D must satisfy exactly one path of h' , therefore, it also satisfies the same path of h . Consequently, the hit and miss counts of each path in h' report exactly the number of tuples in D that satisfy the path.

Although the properties described in Proposition 1 are not necessary, because not all optimal pseudo datasets result in identically structured learned and basis trees, they provide a basis for generating pseudo tuples: paths can define templates of pseudo tuples. For example, the label of the path in Figure 2(a), highlighted by thick lines, defines the template tuple in Figure 2(b), where values of attributes A and C are specified by the path label, and values of attributes B and D , absent from the path, are yet to be determined (indicated by question marks).

² To ease the presentation, we consider only the actual counts. It is straightforward to extend this to relative frequencies.

2.2 Path-Based Data Generation

It is difficult to determine optimal values of a template tuple for attributes absent from a given path. Let us consider a significantly simplified case. Assume that a decision tree contains s paths, where each path has r edges labeled with distinct attributes and is responsible of generating k pseudo tuples. Suppose each tuple has m attributes and each attribute has exactly K distinct values. Since a pseudo table may contain duplicate tuples, the k pseudo tuples of a path can be generated independently. Similarly, since paths of the decision tree partition tuples into disjoint groups, pseudo tuples of different paths can also be generated independently. Therefore, there are $N = ((K^{m-r})^k)^s$ possible pseudo datasets that satisfy the properties stated in Proposition 1. To the best of our knowledge, there is no known efficient algorithm for finding an optimal pseudo table using template tuples defined by decision tree paths. An exhaustive search is obviously too expensive.

We solve the problem with a simple heuristic that exploits impurity measures (such as information gain, Gini-index, or miss-classification rate) used by decision tree learning algorithms. We view the generation of a pseudo table as an inverse process of decision tree learning. Since at each tree node, the attribute that leads to the highest gain of class purity, based on a given impurity measure, is chosen by the learning algorithms to split the training tuples in the node, a pseudo-data generation algorithm can try to force the learning algorithm to select the same attribute, by making sure that the class purity of non-split attributes in the pseudo data is lower than that of the split attribute. Although distributions of non-split attributes in raw data are unknown, we can still minimize the class purity of these attributes in pseudo data by assigning random values from a uniform distribution.

Based on this heuristic, algorithm PGEN (in Figure 3) uses each path of a decision tree to generate a set of (not necessarily distinct) tuples. In line 2 of the algorithm, each path is considered independently. The number of tuples to be generated by a path (see line 3) can be determined in several ways. For example, it can be the sum of the hit and miss counts, or a proportion of a user specified table size.

In line 4 of the algorithm, an empty template tuple is generated and an unspecified value is assigned to each attribute. Subsequently, for each attribute that appears in the path, a value is assigned by function $value(A, p)$ according to the type of the attribute: if the attribute is categorical, the value is the one specified in the path label (line 7); if the attribute is numeric, the value is chosen within the interval specified in the path label (line 8), which can be a fixed default value, such as the middle point or an end point of the interval, or a random value drawn from the interval according to a given distribution. For each attribute that is absent from the path, a random value from its domain is assigned (line 9). Finally, in line 10, a class label is assigned according to the distribution of classes indicated by the hit and miss counts of the path. Non-majority classes can be randomly assigned to the tuple according to a known distribution or a uniform distribution, the latter is used in our experiments.

Algorithm Path-based Pseudo-data Generation (PGEN)
Input: A decision tree T and a set of attributes \mathcal{A} of data
Output: A set S of labeled tuples in \mathcal{A}
Method:

1. $S = \emptyset$;
2. for each path p of T do
3. for pseudo tuple t to be generated by p do
4. $t =$ a new empty tuple;
5. for each attribute $A \in \mathcal{A}$ do
6. if $A \in attr(p)$ then
7. if A is categorical then $t[A] = value(A, p)$;
8. else $t[A] =$ a value selected from $value(A, p)$;
9. else $t[A] =$ a value selected from the domain of A ;
10. $t[class] =$ a class selected based on counts of p ;
11. append t to S ;
12. return S ;

Fig. 3. Path-based Pseudo-data Generation Algorithm

3 Privacy-Preserving Tree Pruning

So far, we only considered the pseudo-data generation by the data miner. We now consider what a data owner needs to do. If the local predictive model of a data source does not leak private information, the data owner needs to do nothing more than releasing the local model. However, a predictive model may leak private information. For example, assume that the class labels of leaf nodes in the decision tree in Figure 2(a) are values of a new attribute, say SA , referred to here as a sensitive attribute, representing some private information. Then the path labeled by $A = a_3, C = c_1, D < d_1$ is open to the homogeneous attack described in [9], because its hit and miss counts reveal that all the tuples in that path have the same sensitive value $SA = 1$.

Although it is possible for a decision tree learned from a private table to include no sensitive attribute, hence to leak no private information, it is common that a decision tree has a sensitive attribute as its class, just like the tree in Figure 2(a). Thus it is important to prevent a knowledge model from leaking private information. A critical question is how to measure the privacy of a knowledge model. In the following, we show that anonymity measures of dataset, such as ℓ -diversity, can be easily adapted to measure privacy of decision trees. In doing so, the privacy of published data and that of pseudo data can be compared with each other.

3.1 Anonymity Measures of Decision Trees

Interestingly enough, there is a close analogy between paths of a decision tree learned from private tables and equivalence classes in an anonymous dataset

produced by generalization methods, such as k -anonymity and ℓ -diversity. In generalization methods, tuples in a private table are generalized to satisfy a privacy requirement according to taxonomies of quasi-identifier (QI) attributes. The generalized tuples form equivalence classes, called QI-groups, in which all tuples have the same QI value. Each QI-group in a generalized dataset needs to satisfy a privacy requirement, such as containing at least k tuples (under k -anonymity) or having well-represented sensitive values (under ℓ -diversity). On the other hand, paths in a decision tree are similar to QI-groups of a generalized dataset. Like QI-groups, paths also partition tuples into equivalence classes: all the tuples in one path have the same values under attributes that are present in the path. We can adapt anonymity measures of QI-groups to measure the privacy of decision tree paths. For example, to measure paths with k -anonymity, we can require each path p of the decision tree to satisfy $p.hit + p.miss \geq k$.

Similarly, we can also measure paths with an ℓ -diversity measure. For example, one of the ℓ -diversity measures proposed in [9] is (c, ℓ) -diversity, which requires every QI-group to satisfy $d_1 \leq c \cdot \sum_{i=\ell-1}^m d_i$, where d_i , $1 \leq i \leq m$, is the number of tuples of the i th largest class within the QI-group. To apply (c, ℓ) -diversity to measure a decision tree, we can require every path p to satisfy $\ell \leq p.miss + 1$ and $p.hit < c \times \frac{p.miss}{\ell-1}$. Intuitively, the largest class of the path, $p.class$, contains $p.hit$ tuples, and all the other classes collectively contain $p.miss$ tuples. If $p.miss > 0$, we need to estimate the sizes of non-majority classes. Specifically, if $p.miss < \ell - 1$, there are at most $\ell - 2$ other classes (each contains one tuple), therefore, the path trivially violates ℓ -diversity. If $p.miss \geq \ell - 1$, there may be ℓ or more classes in the path. Since the class distribution of misclassified tuples is not provided by the decision tree, the adversary may have to assume that misclassified tuples are equally distributed over $\ell - 1$ non-majority classes, with $\frac{p.miss}{\ell-1}$ tuples per class, with the smallest class representing all remaining classes (if there are more than ℓ classes). For example, the highlighted path in Figure 2(a) indicates that there are 14 tuples in class 1, and 6 tuples in classes 2 and 3 (assuming there are total three classes), with three tuples in each class. As a result, this path is $(5, 3)$ -diversified.

3.2 Anonymity-Based Tree Pruning

With generalization methods, QI-groups that does not satisfy a user-specified anonymity requirement will be combined with other QI-groups. The corresponding operation for a decision tree is to combine paths of a common prefix into a single path, which effectively prunes the decision tree. We now present a simple algorithm that prunes a decision tree based on a given anonymity requirement (such as k -anonymity or (c, ℓ) -diversity).

Algorithm APT (see Figure 4) makes two traversals of the tree: a depth-first traversal (lines 2-4) that propagates class and hit/miss counts from leaf nodes to their ancestor nodes, and a bottom-up level-by-level traversal (lines 6-14) that prunes the nodes that does not satisfy a given anonymity requirement. A node is pruned by being combined with some of its sibling nodes (lines 10-13). Once all children nodes of a parent node are pruned, the parent node is transformed

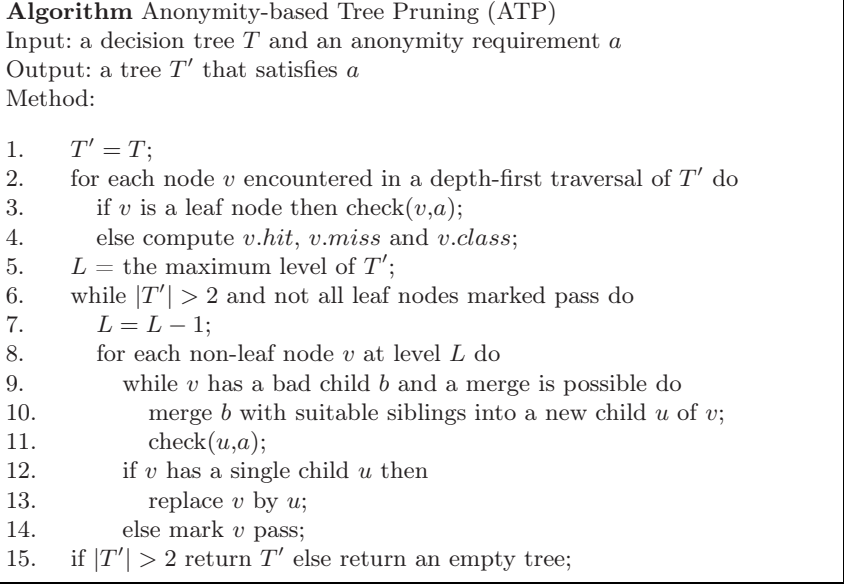


Fig. 4. Anonymity-based Tree Pruning Algorithm

into a new leaf node. If the algorithm terminates successfully, it will produce a decision tree satisfying the anonymity requirement. Otherwise, it produces an empty tree.

During the depth-first traversal, all the leaf nodes are checked against the anonymity requirement, and in the meantime, the hit and miss counts as well as class are calculated for each non-leaf node. A node is marked if it is a leaf node and satisfies the anonymity requirement (Line 3) or all its children are marked (Line 13). In Line 4, the hit and miss counts, as well as the label of the majority class are calculated in a bottom-up fashion. Assume a non-leaf node has k children and the set of classes is C . The majority class of the non-leaf node is the class containing the most tuples in its sub-tree. Due to the recursive traversal, the number of tuples in each class c_j in a node can be estimated as follows, using the hit and miss counts of the node's children nodes:

$$hit_{c_j} = \sum_{v.class=c_j} v.hit + \frac{1}{|C|-1} \sum_{v.class \neq c_j} v.miss \quad (1)$$

where $\frac{v.miss}{|C|-1}$ is the expected number of misclassified tuples in v that belongs to class c_j . Intuitively, if the majority class of a child node is c_j , this child node will contribute $v.hit$ tuples to class c_j of its parent node. If the majority class of a child node is not c_j , it will still contribute a portion of $v.miss$ tuples to class c_j of its parent node. To determine the size of that portion, we assume that all non-majority classes are equally likely among misclassified tuples. This can be

extended easily to use actual distributions of classes in children nodes, if that information is available. Once the hit counts of class labels are calculated, the class and counts of the parent v can be obtained as the following: $v.class = c = \text{maxarg}_{c_j \in C} \{hit_{c_j}\}$, $v.hit = hit_c$, and $v.miss = \sum_{c \in C} (hit_c + miss_c) - v.hit$.

During the bottom-up traversal, starting from the second-to-the-bottom level, the algorithm looks for nodes whose children do not all satisfy the anonymity requirement. Once found, the violating node will be merged with some of its siblings (lines 8-10). Suppose b is the violating node and it has an incoming edge labeled by $A = a$. Different criteria can be used to select the siblings to be merged with b . For example, we can merge all the siblings of b . Alternatively, we can merge b with those siblings whose incoming label is $A = a'$ where a' is a descendant of a . To merge a set of siblings, we remove them together with all their descendants, and then add a single new node under their parent. The hit count, miss count, and class label of this new node are determined in the same way as in line 4, treating the set of sibling nodes as the children of the new node.

If the pruning terminates successfully, the algorithm returns a tree containing a root and at least two leaf nodes, that is $|T'| > 2$. Otherwise, it simply returns an empty tree.

4 Empirical Study

We evaluate the path-based pseudo-data generation and anonymity-based decision tree pruning techniques empirically by comparing the quality of predictive models learned from the pseudo-data with the quality of those models learned directly from the private data. We also compare these techniques with the ℓ -diversity technique of data publishing.

We implemented the PGEN algorithm (described in Section 2.2) and the ATP algorithm (described in Section 3.2) in Java and performed extensive experiments on a Pentium PC with 2GB memory.

The five datasets listed in Table 1, from the UCI Machine Learning Repository, were used in our study. For the purpose of the study, these datasets were preprocessed to remove missing values.

In our experiments, decision trees are always used as local predictive models, but different models, including Naive Bayes classifiers, conjunctive rules, decision

Table 1. Experimental datasets

datasets	#Instances	# Attributes		#Classes
		Nominal	Numeric	
Adult	45204	9	6	2
Car	1728	7	0	4
CMC	1473	8	2	3
Nursery	12960	9	0	5
Train	3000	9	1	2

tables, random forests and decision trees, are used as the global model. These models are considered because they are widely used and are able to handle our datasets. In our study, these predictive models are learned using public-domain Java implementations of their respective learning algorithms.

We consider two pseudo-data generation methods. One method, referred to as BASEGEN, uses the published decision tree to predict and label tuples randomly selected from the data space. The other is PGEN, the path-based pseudo-data generation method.

The quality of predicative models is measured by classification accuracy and class match, defined respectively as the percentage of testing tuples that are correctly classified by the predictive models being tested and the percentage of testing tuples that are classified identically by the models being compared. In addition to these two measures, for decision trees, we also measure path match, defined as the percentage of paths shared by the decision trees under comparison. That is,

$$ss(h, h') = \frac{|paths(h) \cap paths(h')|}{|paths(h) \cup paths(h')|}$$

where for any decision tree h , $paths(h)$ denotes the set of root-to-leaf paths of h .

To account for the randomness in the data, all the quality measures are obtained using the ten-fold validation method and reported here as the average over five iterations. To evaluate the impact of data distribution, we consider distributed environments with 1 to 10 data sources. Specifically, in each run, one fold of data is reserved as the global testing set. The other nine folds are first used to learn a global basis predictive model, referred to as BASIS, and then are randomly partitioned into equal-sized private tables among all the data sources. All the data sources use the same data generation algorithm to create pseudo-data.

To evaluate the tree pruning method, we compare two methods, one method, referred to as PPGEN, is ATP followed by PGEN, and the other method, referred to as LD, is the ℓ -diversity data publishing method. Specifically, we generate a ℓ -diversified dataset using LD and a pseudo dataset using PPGEN from the same raw data, and compare quality measures of predictive models learned from these datasets. Our Java-based implementation of LD is based on [9].

4.1 The Quality of Pseudo-data Generation

In this study, we compare the quality of the two pseudo-data generation methods: BASEGEN and PGEN. To focus on data generation, the decision trees used to generate pseudo-data are not pruned in this study.

Figure 5 shows accuracies of decision trees and Naive Bayes classifiers on three UCI datasets. In the two charts in this Figure, the X-axis represents the number of data sources, ranging from 1 to 10 (showing 1 to 5 in order for the graph to be readable) and the Y-axis is the average classification accuracy. The group of 9 bars consists of three subgroups, each having 3 bars, corresponding to

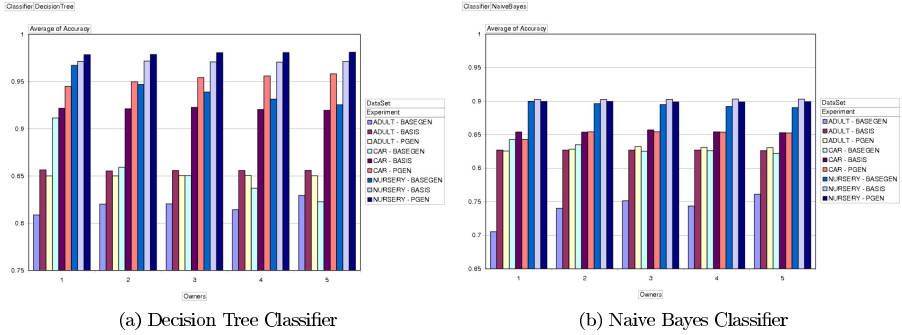


Fig. 5. Accuracy of a single predictive model on Adult, Car and Nursery datasets

BASEGEN, BASIS, and PGEN results with Adult, Car, and Nursery datasets, respectively. The accuracies of BASIS classifiers stay the same across all groups. The results on the other two UCI datasets are similar to those presented here and are omitted to save space. In Figure 5(a), PGEN performs very well as compared to BASIS. BASEGEN on the other hand, consistently performs worse than BASIS. It is interesting that for Car and Nursery datasets, as the number of owners increases, the accuracy of PGEN increases and that of BASEGEN decreases. This trend is not observed with Adult dataset. We observe a similar trend in Figure 5(b), which uses Naive Bayes classifiers rather than decision trees as the basis and learned models, although the difference between BASEGEN and PGEN are not as big as in Figure 5(a) for Car and Nursery data. An exception is with the Adult data, where as the number of data sources increases, the accuracy of BASEGEN improves more significantly than in in Figure 5(a). Similar trends are also observed in results with other predictive models and datasets. From these results, we can conclude that PGEN can generate high quality pseudo-data that effectively preserves the information in the raw data, and BASEGEN can also be useful for some datasets with a small number of data sources.

Figure 6 shows classification accuracies of Conjunctive Rule, Decision Tree and Naive Bayes models with (a) Adult data and (b) Nursery data. Both Decision Tree and Naive Bayes models perform better on Nursery data than on Adult data. The Conjunctive Rule does just the opposite. Accuracy of global decision trees are better than the other two models, which is expected because published local models are also decision trees. It is interesting however that the Naive Bayes model performs equally well with both BASEGEN and PGEN on Nursery data, but does better with PGEN than with BASEGEN on Adult data. Also notice that the conjunctive rule model performs much better with BASEGEN than it does with PGEN on Adult data, which is somewhat surprising. The results obtained with other predictive models on other datasets are similar to those presented in Figure 6(b).

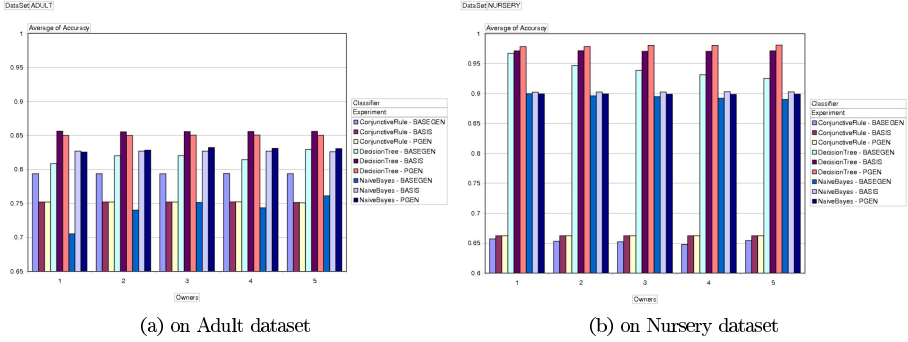


Fig. 6. Accuracy of Conjunctive Rule, Decision Tree, and Naive Bayes classifiers

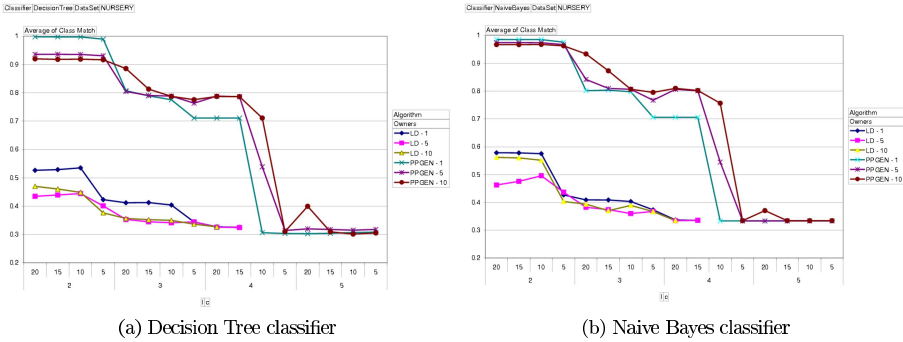


Fig. 7. Privacy vs accuracy of classifiers on Nursery dataset

In addition to classification accuracy, we also obtained results on class match and path match. While the results on class match are very similar to that of classification accuracy, path match results reveal very little similarity between decision trees learned from raw data and those learned from pseudo-data, with typically less than 30% and frequently less than 1% shared paths. This result seems to suggest that the quality of pseudo datasets are not related to the structural similarity of decision trees.

4.2 Effect of Anonymity-Based Decision Tree Pruning

To evaluate ATP method, we compare the quality of PPGEN and LD over a range of (c, ℓ) -diversity privacy requirements, with ℓ ranging from 2 to 5 and c ranging from 5 to 20 (with an increment of 5). In our experiments, we generate an ℓ -diversified dataset using the LD method and learn a global basis predictive model (referred to as BASIS) from each global training set before it is partitioned among data sources. A global predictive model (also referred to as LD) is also learned from the ℓ -diversified dataset. To create a global pseudo dataset, the

global training set is first partitioned among data sources, and subsequently used to learning local decision trees for data sources. These local decision trees are pruned using ATP under the same privacy requirements as used by LD to generate the l -diversified data. Then pseudo datasets are generated from local pruned decision trees using PGEN method, and subsequently combined into the global pseudo dataset. Finally, the global predictive models corresponding to PPGEN are learned. The classification accuracy as well as class match of the BASIS, LD, and PPGEN predictive models are compared here.

Figure 7 shows classification accuracy of decision tree and Naive Bayes models learned from the Nursery dataset. This result is representative of similar results obtained with other datasets. In Figure 7, the X-axis represents privacy requirements roughly ordered from left to right according to increasingly stronger privacy requirements, and the Y-axis represents the average accuracy. The lines represent various combinations of data generation methods (LD vs. PPGEN) and number of data sources (ranging from 1 to 10 with increment of 5). Again, we have to omit some results to make these figures legible.

From Figure 7 (a) and (b), we can see that PPGEN significantly outperforms LD (by more than 30 percentage points) over the entire range of privacy requirements that are tested. Furthermore, PPGEN continues to produce usable pseudo-data (with a 30% accuracy rate) as LD stops to produce any data due to exceedingly high privacy requirement (starting with $l = 4$ and $c = 10$). It is also interesting to see that the performance of PPGEN also falls sharply after $l = 4$ and $c = 10$, and that the increasing number of data sources seems to have a diminishing effect on accuracy.

These results may be attributed to multiple factors. First, LD performs a full dataset generalization on all QI-attributes, but PPGEN supports a hybrid generalization: different paths generalize their own attribute to their own levels, guided by the published decision tree. Second, since LD is applied to the private dataset, it suffers if the dataset is highly skewed or if it lacks a good distribution of classes, but PPGEN is better protected from this because it assumes an even distribution of misclassified tuples at every leaf node. This feature also explains why PPGEN can satisfy higher privacy requirements than LD. Finally, the results support the expected deterioration in quality as privacy increases. Similar results were obtained with the Car dataset.

Notice that, the pseudo dataset generated by PPGEN under a given pair of (l, c) is likely not to satisfy (c, l) -diversity on data. This is because the paths pruned by PPGEN differ from the QI-groups generalized by LD in that they typically involves fewer attributes than do QI-groups. Thus, tuples in the same path may belong to different QI-groups when measured using anonymity measures of datasets. Thus, a pseudo table may have many more smaller QI-groups than the corresponding l -diversified dataset, leading to a violation of the data-oriented l -diversity requirement. This however does not reduce the privacy assurance of PPGEN because the pseudo tuples are randomly generated rather than generalized from the raw data.

5 Related Work

In this section we briefly discuss the relationship between the work described in this paper and the work in several closely related areas. Our work is motivated by the problem of low data utility of existing methods that use perturbation or generalization to protect privacy.

Privacy-preserving data mining based on perturbation techniques was pioneered in [3], which perturbs data values by adding a noise and learns decision trees from an estimation of original data distribution. The adding-noise perturbation method was shown to be flawed in [13], and a number of other perturbation methods have been proposed including matrix perturbation [6,5], ρ_1 -to- ρ_2 privacy breaching [4], and personalized breaching probability [14]. In addition to decision trees, perturbation have also been used in PPDM to find association rules [6,4,15,16] and clustering [17].

The well-known generalization method, K -anonymity, was first studied in [10] for data disclosure. Implementations of k -anonymity include bottom-up generalization [18], top-down specialization [19], and Incognito [20]. The k -anonymity is extended to ℓ -diversity in [9] to respond to two types of attacks against k -anonymity: homogeneous attack and background attack. To improve utility of ℓ -diversified data, [21] presented a method that publishes marginal tables in addition to ℓ -diversified tables. Another extension of k -anonymity, (α, k) -anonymity, is proposed in [22]. Because of the analogy of paths of a decision tree and QI-groups in data produced by these generalization methods, our tree pruning method can be used with any anonymity measure used by these methods.

Protecting privacy by generating pseudo data according to statistics of the private data has been proposed before. The method described in [23] generates pseudo data from statistics of condensed groups, which are sets of predetermined number of data records closest, in a multidimensional space, to randomly selected private records. The method in [7] learns a given number of clusters of at least a given size, and for each cluster, generate pseudo data according to the QI-values of the center, the size of the cluster, and the set of sensitive values in the cluster. Our method of generating pseudo data is different from these methods in that we use the statistics contained in decision trees only for attributes in their paths.

In conventional distributed data mining literature, there has been much work [24,25] on combining classifiers learned from different sources into a meta-classifier. These methods do not teak privacy into consideration and need to resolve conflict among different classifiers using a global testing set, which in our environment will contain private tuples. Our method learns global model from pseudo data rather than by integrating local models directly. This not only avoids using private tuples for testing, but also allows us to learn global models of different types.

The recent work on secure sharing of association rules and frequent patterns [26,27] focus on hiding sensitive association rules and blocking inference channels for the protection of privacy. However, in these papers, what considered private is some association rules or patterns rather than information about individuals.

Crypto-based methods, such as [1,2], do not have the privacy issue of local knowledge models because nothing is shared among data sources. But, in addition to performance and scalability issues, these methods still face the privacy issue of global knowledge models they produce, to which our method for measuring and preserving the privacy of knowledge models remains relevant.

6 Conclusions

In this paper, we present a new PPDM approach that aims to learn high quality knowledge models yet still protect privacy. With this knowledge model sharing approach, each data source releases a privacy-preserving local knowledge model learned from its private data, and a data miner mines pseudo data generated from the local knowledge models. We define the problem of generating pseudo-data from predictive models, such as decision trees and sets of classification rules, and present a heuristic algorithm for a data miner to generate pseudo-data according to paths of a decision tree. For data owners, we show how to adapt anonymity measures of datasets to measure the privacy of decision trees, and present an algorithm that prunes a decision tree to satisfy a given anonymity requirement. Our results, obtained through an empirical study, show that predictive models learned using our method are significantly more accurate than those learned using the existing ℓ -diversity method in both centralized and distributed environments with different types of datasets, predictive models, and utility measures.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments that improved the quality of this paper.

References

1. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000)
2. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. ACM SIGKDD Explorations 4(2), 12–19 (2003)
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: ACM SIGMOD International Conference on Management of Data, pp. 439–450. ACM, New York (2000)
4. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaching in privacy preserving data mining. In: ACM Symposium on Principles of Database Systems, pp. 211–222. ACM, New York (2003)
5. Dowd, J., Xu, S., Zhang, W.: Privacy-preserving decision tree mining based on random substitutions. In: International Conference on Emerging Trends in Information and Communication Security, Freiburg, Germany (June 2006)

6. Agrawal, S., Haritsa, J.R.: A framework for high-accuracy privacy-preserving mining. In: IEEE International Conference on Data Engineering (2005)
7. Aggarwal, G., Feder, T., Kenthapadi, K., Khuller, S., Panigrahy, R., Thomas, D., Zhu, A.: Achieving anonymity via clustering. In: Proceeding of the 25th ACM Symposium on Principles of Database Systems (June 2006)
8. Aggarwal, C.C.: On k-anonymity and the curse of dimensionality. In: International Conference on Very Large Data Bases, pp. 901–909 (2005)
9. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ -diversity: Privacy beyond k-anonymity. In: IEEE International Conference on Data Engineering (2006)
10. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In: Proc. of the IEEE Symposium on Research in Security and Privacy (1998)
11. Xu, S., Zhang, W.: PBKM: A secure knowledge management framework (extended abstract). In: NSF/NSA/AFRL Workshop on Secure Knowledge Management, Buffalo, NY (2004)
12. Xu, S., Zhang, W.: Knowledge as service and knowledge breaching. In: IEEE International Conference on Service Computing (SCC 2005) (2005)
13. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: IEEE International Conference on Data Mining (2003)
14. Xiao, X., Tao, Y.: Personalized privacy preservation. In: ACM SIGMOD International Conference on Management of Data, pp. 229–240 (2006)
15. Evmievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: International Conference on Knowledge Discovery and Data Mining (2002)
16. Rizvi, S.J., Haritsa, J.R.: Maintaining data privacy in association rule mining. In: International Conference on Very Large Data Bases (2002)
17. Merugu, S., Ghosh, J.: Privacy-preserving distributed clustering using generative models. In: IEEE International Conference on Data Mining (2003)
18. Wang, K., Yu, P.S., Chakraborty, S.: Bottom-up generalization: A data mining solution to privacy protection. In: IEEE International Conference on Data Mining (2004)
19. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specification for information and privacy preservation. In: IEEE International Conference on Data Engineering (2005)
20. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: ACM SIGMOD International Conference on Management of Data (2005)
21. Kifer, D., Gehrke, J.E.: Injecting utility into anonymized datasets. In: ACM SIGMOD International Conference on Management of Data (2006)
22. Wong, R.C.-W., Li, J., Fu, A.W.-C., Wang, K. (α , k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In: International Conference on Knowledge Discovery and Data Mining (2006)
23. Aggarwal, C., Yu, P.: A condensation approach to privacy preserving data mining. In: International Conference on Extending Database Technology, pp. 183–199 (2004)
24. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man and Cybernetics 22(3), 418–435 (1992)

25. Woods, K., Kegelmeyer Jr., W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(4), 405–410 (1997)
26. Oliveira, S., Zaane, O., Saygin, Y.: Secure Association Rule Sharing. In: Dai, H., Srikant, R., Zhang, C. (eds.) *PAKDD 2004. LNCS (LNAI)*, vol. 3056, pp. 74–85. Springer, Heidelberg (2004)
27. Wang, Z., Wang, W., Shi, B.: Blocking inference channels in frequent pattern sharing. In: *IEEE International Conference on Data Engineering*, pp. 1425–1429 (2007)