Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 04-023

Privacy Leakage in Multi-relational Databases via Pattern based Semi-supervised Learning

Hui Xiong, Michael Steinbach, and Vipin Kumar

June 01, 2004

Privacy Leakage in Multi-relational Databases via Pattern based Semi-supervised Learning

Hui Xiong, Michael Steinbach, and Vipin Kumar

Department of Computer Science, University of Minnesota-Twin Cities,
Minneapolis MN 55455, USA
{huix, steinbac, kumar}@cs.umn.edu

Abstract. In multi-relational databases, a view, which is a context- and content-dependent subset of one or more tables (or other views), is often used to preserve privacy by hiding sensitive information. However, recent developments in data mining present a new challenge for database security even when traditional database security techniques, such as data access control via a view, are employed. This paper presents a data mining framework using semi-supervised learning that demonstrates the potential for privacy leakage in multi-relational databases. Many different types of semi-supervised learning techniques, such as K-nearest neighbor (KNN) methods, can be used to demonstrate privacy leakage. However, we also introduce a new approach to semi-supervised learning, hyperclique pattern based semi-supervised learning (HPSL), which differs from traditional semi-supervised learning approaches in that it considers the similarity among groups of objects instead of only pairs of objects. Our experimental results show that both the KNN and HPSL methods have the ability to compromise database security, although HPSL is better at this privacy violation (has higher accuracy) than KNN methods. Finally, we provide a principle for avoiding privacy leakage in multi-relational databases via semi-supervised learning and illustrate this principle with a simple preventive technique whose effectiveness is demonstrated by experiments.

1 Introduction

In multi-relational databases, a view, which is a context- and content-dependent subset of one or more tables (or other views), is often used to preserve privacy by hiding sensitive information. For instance, a view might be created to allow a sales manager to see only that portion of a customer table that is related to customers in the manager's own territory. This view might also be limited to selected columns from the base tables in which the subset of customer information is contained. Since a key purpose of database views is to hide sensitive information by controlling data access, the results of security breaches in database views can be serious, ranging from financial exposure to disrupted operations.

A concern for database security was initially raised by Codd's fundamental work on relational databases [2]. Indeed, one of the main issues faced by database

security professionals is avoiding or limiting the inference capabilities of database users. Specifically, the goal is to limit the ability of database users to employ information available at one security level to infer facts that should be protected at a higher security level.

In this paper, our concern with database security is different from the inference problem mentioned above. More specifically, if some information from a higher security level is known by a user at a lower security level,[1] then this user may be able to predict additional information that should be protected at a higher security level. Indeed, the focus of this paper is to present a framework based on semi-supervised learning that illustrates the potential for this type of privacy leakage in multi-relational databases.

Semi-supervised learning techniques use both labeled and unlabeled data for predicting class labels for unlabeled objects. Among such techniques, there is a promising family of methods which are analogous to the traditional K-Nearest-Neighbor (KNN) method used in supervised learning [12]. The hypothesis behind these methods is that similar data objects tend to have similar class labels.

More recently, we have defined a new pattern for association analysis—the *hyperclique pattern* [16]—that demonstrates a particularly strong connection between the overall similarity of a set of attributes (or objects) and the itemset (local pattern) in which they are involved. The hyperclique pattern, described in more detail later, possesses the *strong affinity property*, i.e., the attributes (objects) in a hyperclique pattern have a guaranteed level of global pairwise similarity to one another as measured by the cosine or Jaccard similarity measures [14]. Intuitively, a hyperclique pattern includes objects which tend to be from the same class category. In this paper, we propose a new semi-supervised learning approach that is based on this observation, the hyperclique pattern based semi-supervised learning (HPSL) method. By considering the similarity among all objects in a hyperclique pattern instead of the similarity between only pairs of objects, we can improve semi-supervised learning results over those based on KNN approaches.

The main contributions of this paper can be summarized as follows.

- We describe a new challenge for database security from the data mining/machine learning perspective. More specifically, we show that classic database security techniques may be inadequate in light of developments in semi-supervised learning. To demonstrate this, we present a framework that illustrates the potential for privacy leakage in database views with respect to semi-supervised learning.

- We introduce a new semi-supervised learning approach, the hyperclique pattern based semi-supervised learning (HPSL) method. We use this technique, along with a couple of KNN semi-supervised learning approaches, to illustrate privacy leakage in database views. (However, other semi-supervised

[1] Information from a higher security level could be obtained in a variety of ways, for example, by eavesdropping—electronic or otherwise—but the details are outside the scope of this paper.

learning techniques could also be used.)

- We conduct extensive experiments on several real data sets to show the effectiveness of the HPSL method. Our experimental results show that the HPSL approach can achieve better prediction accuracy than traditional K-Nearest Neighbor (KNN) techniques.

- We provide a pseudo-attribute perturbation method that protects databases from a potential semi-supervised learning attack by invalidating the 'similar data objects tend to have similar class labels' assumption on which semi-supervised learning techniques are based. Experimental results indicate that this method can effectively decluster objects in the same category, thus providing enhanced database security.

2 Problem Formulation

The purpose of this paper is to investigate privacy leakage in database views via semi-supervised learning. We formalize this problem as follows:

Problem: Privacy Leakage in Multi-relational Data-bases via Semi-supervised Learning

- **Given**:
  - ϑ is a view that contains selected attributes, $I = \{i_1, i_2, \ldots, i_m\}$, potentially from several base tables;
  - $C = \{c_1, c_2, \ldots c_p\}$ is a set of attributes which are to be protected and are not provided in ϑ;
  - $O = \{o_1, o_2, \ldots, o_n\}$ is a set of tuples (objects) in ϑ for which the values of attributes in the set C are unknown;
  - $K = \{k_1, k_2, \ldots, k_l\}$ is a set of tuples (objects) in ϑ for which the values of attributes in the set C are known.

- **Design:** An effective semi-supervised learning mechanism to predict the values of attributes in the set C for objects in O.

- **Objective:** Predict, with high accuracy, the values of attributes in the set C for *some* objects in O.

- **Constraints:** 1) $l << n$, where l is the number of objects with known class labels and n is the total number of objects. 2) The attributes in the set I have predictive power for the attributes in the set C.

Example 1. Figure 1 gives an illustration of the problem. In the figure, the view contains m attributes and n tuples (objects). All the information in the view is known by the user. Also, there are p attributes that are in base tables but not

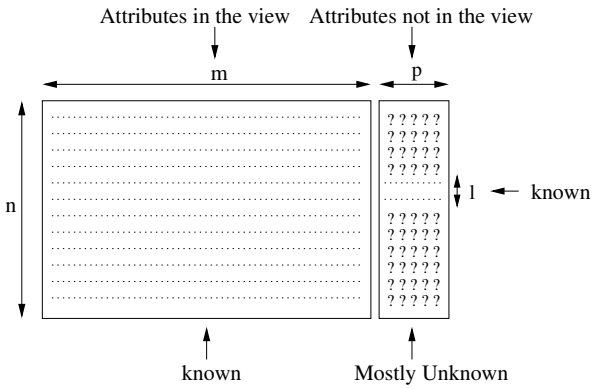


Fig. 1. Illustration of the Problem

in the view. Hence, the information in these p attributes is unknown to a user of the database view. However, if, for some objects, these p attributes are known to a user of the database view, then such a user may use semi-supervised learning techniques to predict these p attributes for other objects. This is the problem addressed in this paper.

3 Background and Related Work

As mentioned, a goal of database security is to limit the ability of users to infer facts that should be protected at a higher security level. Consider the following example.

Example 2. Given a multi-relational database, assume that there is a view named 'cargo' that contains information on the various cargo holds available on each outbound airplane. Each row in this view represents a single shipment and lists the contents of that shipment and the flight identification number. The flight identification number may be cross-referenced with other base tables to determine the origin, destination, flight time, and other information. The 'cargo' view is presented in Table 1.

Table 1. The 'cargo' view.

Flight ID	Cargo	Contents	Classification
1200	A	Boots	Unclassified
1200	B	Toys	Unclassified
1200	C	Guns	Top Secret
1200	D	Butter	Unclassified

If a database user with a top secret security clearance requests information on the cargo carried by flight 1200, then this user would see all shipments. However,

if a user without a security clearance requests the data, then this user would not see the top secret shipment. The above correctly implements the security rules that prohibit someone with lower security levels from seeing information with higher security levels. However, assume that there is a uniqueness constraint on flight ID and cargo (to prevent the scheduling of two shipments for the same hold). When a user without top secret clearance sees that nothing is scheduled for cargo hold C on flight 1200, this user might attempt to insert a new record to reserve this cargo hold. However, the insertion of this record will fail due to the uniqueness constraint. At this point, the user can infer that there is a secret shipment on flight 1200 and could then cross-reference the flight information table to find out the source and destination of the secret shipment, as well as various other information.

There are two basic approaches to avoid such inference. One is polyinstantiation [4], which allows the creation of multiple instances of data records in a way such that a user with lower security levels can see the tuple associated with a particular primary key populated with one set of element values. However, a user with higher security levels may see the 'same' tuple with perhaps different values for some of the attributes or see multiple tuples corresponding to different security levels. Another approach is to perform data access control and ensure that the set of all classification constraints is consistent [3]. In other words, even if users at lower security levels can know about the existence of a classified shipment, they will not have access to information about the contents of that shipment.

Here, we raise another concern from a data mining perspective. Our hypothesis is that, if a portion of information at a higher security level is known to a user at a lower security level, then it is possible that this user may infer more information at a higher security level. This problem can be treated as semi-supervised learning on labeled data (some leaked information at a higher security level) and unlabeled data (information at a lower security level) [12].

Traditional supervised learning methods build a prediction model from labeled data and use this model for predicting the labels of objects with unknown labels. However, in the real world, there are many situations where only a small fraction of the objects are labeled. In this case, many difficulties may arise in building a prediction model based solely on labeled data. This is referred to as the small training sample size problem [6, 11] in machine learning.

Semi-supervised learning techniques [12], which make use of both unlabeled and labeled data, have recently been proposed to cope with the above challenge. Among such techniques, there is a promising family of methods that are similar to the K-Nearest-Neighbor (KNN) technique used in traditional supervised learning [12]. The hypothesis behind these methods is that similar data objects tend to have similar class labels. Our hyperclique pattern based semi-supervised learning method (HPSL) is also based on this hypothesis, but differs from KNN approaches in that it considers the similarity among a group of objects instead of just the similarity between pairs of objects.

Finally, the objective of most traditional semi-supervised learning methods is to learn class labels for all the objects while our HPSL method only predicts the class labels for objects around the objects with known class labels. The KNN approaches that we use for our experiments are also employed in the same manner.

4 Hyperclique Patterns

A hyperclique pattern [16] is a new type of association pattern that contains items that are *highly affiliated* with each other. Specifically, the presence of an item in one transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure captures the strength of this association and, for an itemset $P = \{i_1, i_2, \cdots, i_m\}$, is defined as the minimum confidence of all association rules of the itemset with a left hand side of one item, i.e., $hconf(P) = min\{conf\{i_1 \rightarrow i_2, \ldots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \ldots, i_m\}, \ldots, conf\{i_m \rightarrow i_1, \ldots, i_{m-1}\}\}$. An itemset P is a **hyperclique pattern** if $hconf(P) \geq h_c$, where h_c is the minimum h-confidence threshold. A hyperclique pattern is a **maximal hyperclique pattern** if no superset of this pattern is a hyperclique pattern.

For example, consider an itemset $P = \{A, B, C\}$. Assume that $supp(\{A\}) = 0.1$, $supp(\{B\}) = 0.1$, $supp(\{C\}) = 0.06$, and $supp(\{A, B, C\}) = 0.06$, where *supp* is the association rule support [1]. Then

$$conf\{A \rightarrow B, C\} = supp(\{A, B, C\})/supp(\{A\}) = 0.6$$
$$conf\{B \rightarrow A, C\} = supp(\{A, B, C\})/supp(\{B\}) = 0.6$$
$$conf\{C \rightarrow A, B\} = supp(\{A, B, C\})/supp(\{C\}) = 1$$

Hence, $hconf(P) = min\{conf\{B \rightarrow A, C\}, conf\{A \rightarrow B, C\}, conf\ \{C \rightarrow A, B\}\} = 0.6$.

Table 2. Example Hyperclique Patterns

LA1 Dataset		
Hyperclique patterns	Support	H-confidence
{gorbachev, mikhail}	1.4%	93.6%
{photo, graphic, writer}	14.5%	42.1%
{season, team, game, play}	7.1%	31.4%

Table 2 shows some hyperclique patterns identified from a document data set, which includes articles from various news categories such as 'financial,' 'foreign,' 'sports,' and 'entertainment.' For instance, the hyperclique pattern {season, team, game, play} is from the 'sports' category.

The h-confidence measure has three important properties, namely the anti-monotone property, the cross-support property, and the strong affinity property [16]. The anti-monotone property is analogous in definition and use to the anti-monotone property of the support measure used in association-rule mining [1].

The cross-support property allows us to efficiently eliminate patterns involving items with different support levels. Together, the anti-monotone and cross-support properties form the basis of an efficient hyperclique mining algorithm that has much better performance than frequent itemset mining algorithms, particularly at low levels of support. Finally, the strong affinity property guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, h_c, then every pair of items within the hyperclique pattern must have a cosine similarity greater than or equal to h_c.

5 Semi-Supervised Learning Using Nearest Neighbor Approaches

In general, there are two potential directions for semi-supervised learning using nearest neighbors. One is a **K** **N**earest **N**eighbor based **S**emi-supervised (KNNS) learning method. Another one is a **TOP-K** **N**earest **N**eighbor based **S**emi-supervised (TOP-K NNS) learning method.

5.1 The KNNS Method

We first describe the KNNS method. For each given object with a class label, KNNS uses the class label of the given object to label each of the k nearest neighbors. If a predicted object is found to be one of k nearest neighbors of more than one given object, then KNNS assigns the label of the given object with the highest similarity.

The KNNS method is simple and easy to implement. However, the KNNS method only considers pairs of similar objects when labeling the data objects. Indeed, in real world data sets, it is possible that two objects are often nearest neighbors without belonging to the same class [13]. This is illustrated by experimental results, as shown in Section 8.1. From this perspective, the KNNS method works in a greedy fashion.

In addition, the KNNS method predicts an equal number of objects for each given object with a class label; that is, the KNNS method gives each labeled object equal weight as a predictor. This may not be appropriate in real-world data sets. Intuitively, objects from a high-density cluster may predict more objects with a high accuracy. In contrast, objects from a loosely connected cluster may only have limited predictive power. In the worst case, a labeled object can be noise or an outlier that is completely unsuitable for prediction.

5.2 The TOP-K NNS Method

In this subsection, we present the TOP-K NNS method. For n given objects with class labels, the TOP-K NNS method finds the k objects with the highest level of similarity from the neighborhood of these n objects.

The TOP-K NNS method has two appealing characteristics. First, this method is simple and is easy to implement. Second, based on similarity, the TOP-K NNS

method assigns different predictive power to different labeled objects. Hence, unlike the KNNS method, the TOP-K NNS method can avoid many prediction errors when some labeled objects are noise or an outlier.

The TOP-K NNS method is a greedy-choice algorithm. The prediction mechanism of this algorithm is solely based on pair-wise similarity. As already noted, in real world data sets, it is quite possible that two objects can be nearest neighbors without belonging to the same class [13]. As a result, many prediction errors can be generated by this greedy choice approach. Also, this method gives higher predictive power to objects from a dense cluster. If the labeled objects are from both dense and sparse clusters, it is possible that 1) few objects from a sparse cluster can be predicted if the value of k is small and 2) if a very large value of k is specified, then some objects from a sparse cluster can be predicted. However, in the latter case, more prediction errors will be introduced for those objects from dense clusters.

6 HPSL: Hyperclique Pattern Based Semi-supervised Learning

In this section, we present a hyperclique pattern [16] based semi-supervised learning (HPSL) approach and show the difference between HPSL and KNN approaches.

6.1 The HPSL Algorithm

The problem of privacy leakage in multi-relational databases via semi-supervised learning assumes that only a small portion of the objects have class labels. For an object with a class label, our purpose is to find a maximal hyperclique pattern that contains this object and then label all other objects in the pattern with the label of this object.

Figure 2 shows the pseudocode of the HPSL algorithm. This algorithm consists of two phases. In phase I, HPSL finds maximal hyperclique patterns with object constraints. For a given object O, we will use the maximal hyperclique pattern in which it occurs for semi-supervised learning. In Phase II, HPSL labels all the unlabeled objects in the discovered maximal hyperclique patterns.

The detailed steps of this algorithm are explained as follows. Line 1 finds a set of unlabeled objects. Line 2 sorts labeled and unlabeled objects based on their support. Line 3 generates an itemset for single items. Line 4 prunes the single itemsets by using the minimum support threshold. Lines 5 through 9 generate hyperclique patterns from size 2 to size k. Line 6 generates candidate size k patterns by joining pairs of patterns with size $k-1$. Note that we only need to generate candidate patterns that contain objects with class labels, which reduces the search space significantly. Next, in line 7, we prune candidate patterns using the anti-monotone and cross-support properties of h-confidence [16] and generate hyperclique patterns. Line 10 generates maximal hyperclique patterns. Finally, lines 11–15 generate semi-supervised learning results.

HPSL Algorithm
Input: D: a data set.
O: a set of objects with class labels.
θ: a minimum h-confidence threshold.
α: a minimum support threshold.
Output: SLR: Semi-supervised Learning Results.
Method
Phase I: Hyperclique Mining with Object Constraints
1. $O_l \leftarrow$ unlabeled_item(D, O)
2. O $\leftarrow$ sort(O); $O_l \leftarrow$ sort(O_l)
3. m $\leftarrow O \cup O_l$
4. $C_1 \leftarrow$ prune(m, α)
5. **for** the size of itemsets, i, in $(2, 3, \ldots, k-1)$ do
6. $C_i \leftarrow join(C_{i-1}, C_{i-1})$
7. $C_i \leftarrow$ prune(C_i, θ, α)
8. S $\leftarrow S \cup C_i$
9. **endfor**
10. MS $\leftarrow$ maximal_hyperclique_pattern(S)

Phase II: Semi-supervised Learning
11. **for** each pattern p in MS
12. SL $\leftarrow$ semi_label(p)
13. SLR $\leftarrow$ SLR $\cup$ SL
14. **endfor**
15. OUTPUT SLR
16. **End**

Fig. 2. The HPSL Algorithm

There are several benefits of the HPSL method. First, this method only predicts class labels for objects strongly connected to objects with known class labels. Recall that the KNNS and TOP-K NNS methods also have this characteristic. Second, unlike the KNNS method, HPSL considers the similarity among groups of objects instead of just pairs of objects. Third, hyperclique patterns represent unique concepts that may potentially help guide better information inference in databases. Finally, the application of the HPSL method for attacking database security reveals an interesting direction for multi-relational data mining [5].

6.2 A Comparison of HPSL and KNNS

In this subsection, we illustrate the difference between the KNNS method and the HPSL method in terms of the scope of objects that may be predicted. Assume that we already know class labels for five objects: O_1, O_2, O_3, O_4, and O_5. The KNNS method will find the k most similar neighbors around each object with a class label and label these neighbors using the label of the given object. In this case, the KNNS method treats every object with a class label as an equally good

predictor and will predict the same number of objects for each labeled object. In contrast, for each object with a class label, the HPSL method finds the maximal hyperclique pattern that contains this object and labels all members of this pattern with this class label. Hence, for the HPSL method, each object with a class label has different predictive power. In other words, different numbers of objects are predicted for each object with a class label. This is desirable since it better fits real world situations.

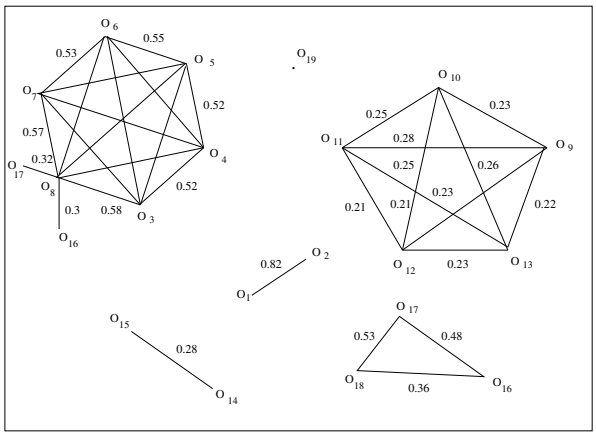


Fig. 3. The Working Mechanism of HPSL.

The rationale behind the HPSL method is as follows. If there is a clustering effect in a real life data set, then different clusters often have different cluster sizes. Objects representing noise and outliers may also be present. Hence, it is natural that different objects should have different predictive power, since an object that represents noise or an outlier may not predict anything, while an object from a large cluster may be used to predict class labels for many objects within this cluster. Consider the following example.

Example 3. Figure 3 shows a hyperclique pattern graph of a sample data set with 19 objects at the h-confidence threshold 0.2. If the cosine similarity of two objects is above 0.2, an edge is put between them and the similarity value is the weight of the edge. In this sample data set, we assume that the class labels for objects: O_1, O_3, and O_{19} are known. If the KNNS method is applied, k nearest neighbors around O_1, O_3, and O_{19} will be found and labeled using the class label of O_1, O_3, and O_{19} respectively. Not surprisingly, the prediction accuracy is extremely poor. Since the object O_{19} is a noise point, the class label of O_{19} is expected to be different from the objects found by the KNNS method. Also, for the object O_3, the KNNS method makes a prediction for its k similar neighbors, however the scope of this inference is limited. In contrast, the HPSL method will first

find a maximal hyperclique pattern for each object. Since there is no hyperclique pattern that can be found for the object O_{19}, no objects can be predicted by this object using the HPSL method. Also, since $\{O_3, O_4, O_5, O_6, O_7, O_8\}$ is a hyperclique pattern at the h-confidence threshold 0.2—recall that the cosine similarity of pairwise objects in this pattern is above 0.2—the HPSL method will label all members in this pattern with the label of the object O_3. Hence, it is expected that the HPSL method will achieve better prediction accuracy and will be more suitable for real world situations.

6.3 A Comparison of HPSL and TOP-K NNS

The major difference between the HPSL method and the TOP-K NNS method is that the TOP-K NNS method is a greedy choice algorithm, which is solely based on pairwise similarity. In contrast, the HPSL method takes the similarity among all objects into consideration, rather than relying only on pairwise similarity. Therefore, the HPSL method has the potential for avoiding prediction errors that result from the greedy choice approach. Also, if the labeled objects come from clusters with different cluster densities, it is possible that k objects with top k highest similarity are all from a dense cluster. Hence, no object will be predicted from a cluster with a lower density even though objects in a cluster with a lower density may be tightly coupled. Consider the following example.

Example 4. For the data set in Figure 3, assume that the class labels of objects O_{11} and O_8 are known and we want to predict 5 objects. For the TOP-K NNS method, five objects, O_3, O_4, O_5, O_6, O_7, around O_8 have the top-5 highest similarity, so these objects are labeled by the label of O_8 and no object will be predicted by the object O_{11}. In order to predict some objects around O_{11}, we have to increase the number of objects for prediction. However, this may result in some prediction errors around the labeled object O_8 due to the greedy choice of pair-wise similarity. For instance, before we can predict objects around O_{11}, we must first predict additional objects, such as O_{16} and O_{17}, around the object O_8. This can introduce a lot of prediction errors. In contrast, the HPSL method can find $\{O_9, O_{10}, O_{11}, O_{12}, O_{13}\}$ and $\{O_3, O_4, O_5, O_6, O_7, O_8\}$ as hyperclique patterns and then label all members in these two patterns with the labels of O_8 and O_{11}, respectively.

7 Prevention

The results of security breaches in database views can be extremely serious. In this section, we investigate how to prevent the potential information leakage from semi-supervised learning in multi-relational databases.

In the following, we first give a principle which can be used to guide the design of effective techniques for preventing semi-supervised learning attacks.

Principle 1 *A prevention technique should have the ability to decluster the data, i.e., similar data objects tend to have different class labels.*

Pseudo-Attribute Perturbation

Input: D: a data set.
m: the number of attributes.
n: the number of data objects.
p: the number of classes.

Variables: S: a set of pseudo-attributes.

Output: an extended data set D' with k number of pseudo-attributes

Method

1. S $\leftarrow$ random_sample_attributes(k, D)
2. $(S_1, S_2, \ldots, S_p) \leftarrow$ partition(S, p)
3. **For** i = 1 to p do
4. $\quad$ **c** $\leftarrow$ centroid(S_i)
5. $\quad$ **For** each O in S_i do
6. $\quad\quad$ O $\leftarrow$ perturbation(O, **c**)
7. $\quad$ End for
8. **End for**
9. S $\leftarrow (S_1, S_2, \ldots, S_p)$
10. D' = D $\cup$ S
11. OUTPUT D'
12. **End**

Fig. 4. Pseudo-Attribute Perturbation

The rationale of this principle follows directly from the hypothesis of semi-supervised learning attacks: similar data objects should have similar class labels.

Following this principle, we provide a preventive technique by introducing pseudo-attributes into the database view. The purpose of these pseudo-attributes is to make sure that the objects with the same class label have poor similarity to each other. In other words, pseudo-attributes can decluster objects with the same class label.

The key idea of the pseudo-attribute perturbation method is illustrated as follows. First, consider Equation 1 which defines the cosine similarity measure for two vectors $x = \{x_1, x_2, \ldots, x_m\}$ and $y = \{y_1, y_2, \ldots, y_m\}$. Our purpose is to introduce some pseudo-attributes into these two vectors such that the value $\sum xy$ is decreased and the value $\sqrt{\sum x^2 \sum y^2}$ is increased. As a result, the cosine similarity between x and y is reduced.

$$cos(x, y) = \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}} \tag{1}$$

Figure 4 shows the pseudo-attribute perturbation algorithm. Line 1 randomly samples k attributes from the data set for perturbation. For the data set with the selected k attributes, Line 2 partitions objects based their class labels into p groups. For each group, Line 4 computes the centroid vector **c** and Line 6

reduces the cosine similarity between each object O and the centroid $\mathbf{c}$ using the above mentioned idea. The data with perturbed attributes are combined with the original data set to form a new data set.

Finally, we should point out that the above proposed preventive technique may not be the best approach to deter semi-supervised learning attacks. However it does increase the complexity for an attacker to a higher level.

8 Experimental Evaluation

In this section, we demonstrate the potential information leakage in database views via a hyperclique pattern based semi-supervised learning technique (HPSL) with experiments on several real-world data sets. The relative performance between HPSL and KNNS as well as TOP-K NNS is also presented.

Experimental Data Sets. For our experiments, we used three real life data sets, which are from several different application domains. Some characteristics of these data sets are shown in Table 3.

Table 3. Characteristics of data sets.

Data Set	LA1	RE0	WAP
Number of Documents	3204	1504	1560
Number of Words	31472	11465	8460
Number of Classes	6	13	20
Min Class Size	273	11	5
Max Class Size	943	608	341
Min Class Size/Max Class Size	0.29	0.018	0.015
Source	TREC-5	Reuters	WebAce

The `LA1` data set is part of the TREC-5 collection [15] and contains news articles from the Los Angeles Times. The `RE0` data set is from the Reuters-21578 text categorization test collection Distribution 1.0 [9]. The data set `WAP` is from the WebACE project (WAP) [7]; each document corresponds to a web page listed in the subject hierarchy of Yahoo!. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [10].

Entropy Measure. In our experiments, we applied the entropy measure for evaluating the clustering effect in a data set. To compute the entropy of a set of clusters, we first calculate the class distribution of the objects in each cluster, i.e., for each cluster j we compute p_{ij}, the probability that a member of cluster j belongs to class i. Given this class distribution, the entropy, E_j, of cluster j is calculated using the standard entropy formula $E_j = -\sum_i p_{ij} log(p_{ij})$, where the sum is taken over all classes and the *log* is *log* base 2. The total entropy for a set of clusters is computed as the weighted sum of the entropies of each cluster, as shown in the equation $E = \sum_{j=1}^{m} \frac{n_j}{n} * E_j$, where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data points.

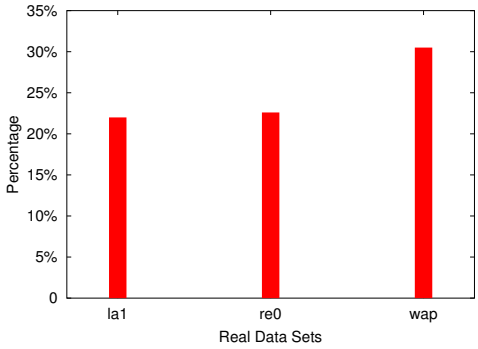


Fig. 5. The Percent of Documents Whose Nearest Neighbor is of a Different Class

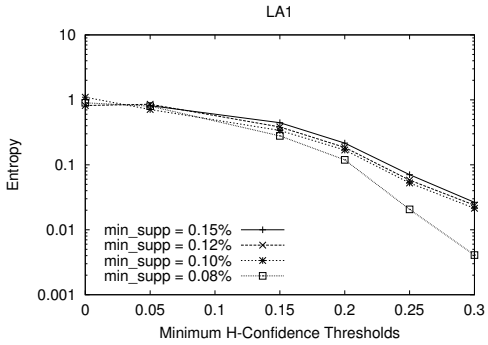


Fig. 6. Cluster nature of hypercliques.

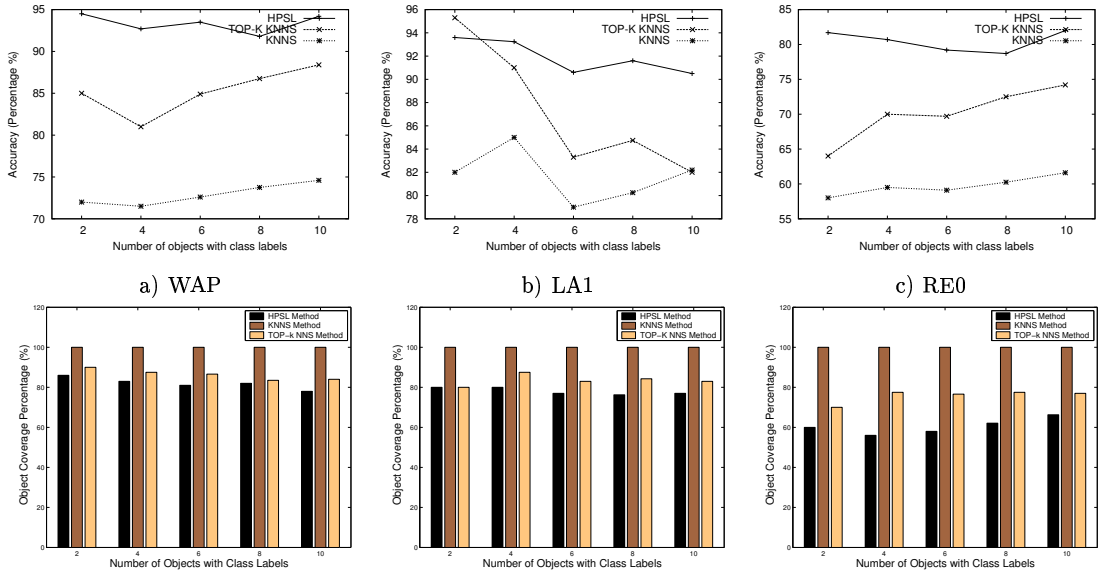
8.1 A Potential Problem with NN Approaches

In real world data sets, it is possible that two objects can often be nearest neighbors without belonging to the same class. To illustrate this, let us consider real document data sets. Figure 5 shows the percent of documents whose nearest neighbor is not of the same class. While this percentage varies widely from one data set to another, the chart confirms the previous claim about nearest neighbor behavior in documents.

Since, in many cases, the nearest neighbors of an object are of different classes, the nearest neighbor based semi-classification approach will often assign objects of different classes to the same class. To cope with this challenge, our HPSL method considers the similarity among all objects instead of the two most similar objects.

8.2 Cluster Nature of Hyperclique Patterns

In this experiment, we explain why the hyperclique pattern is a good candidate to use for a pattern based semi-supervised learning. Figure 6 shows, for



a) WAP b) LA1 c) RE0

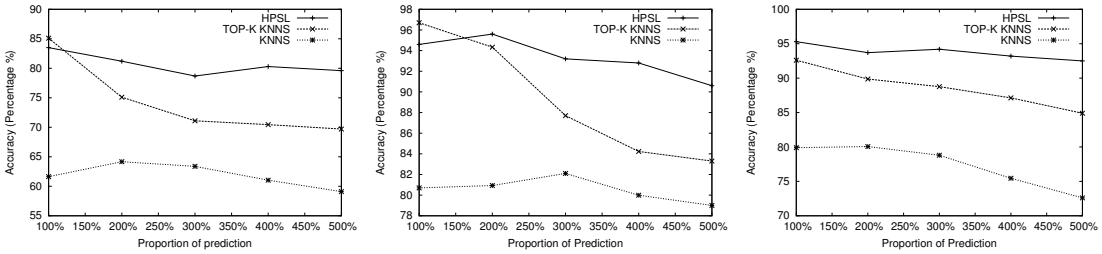
d) WAP e) LA1 f) RE0

Fig. 7. The effect of changing the number of instances with known class labels

the LA1 data set, the entropy of the discovered hyperclique patterns for different minimum h-confidence and support thresholds. As Figure 6 shows, when the minimum h-confidence threshold increases, the entropy of hyperclique patterns decreases dramatically. For instance, when the h-confidence threshold is higher than 0.25, the entropy of hyperclique patterns is less than 0.1 at all the given support thresholds. This indicates that, at certain h-confidence thresholds, hyperclique patterns tend to include objects from the same class.

8.3 The Effect of Changing the Number of Objects with Known Class Labels

Here, we show the relative performance of the HPSL method and the nearest neighbor based approaches including KNNS and TOP-K NNS as the number of objects with known class labels is increased. More specifically, we present the prediction accuracies and the object coverage—the percentage of the given labeled objects that have been used for prediction—when the number of objects with class labels is 2, 4, 6, 8, and 10, respectively. In this experiment, we specify the total number of predicted objects to be approximately five times more than the number of objects with class labels. We also performed random sampling to select objects with class labels. Finally, in order to reduce the random effect, we conducted 10 trials for each experiment.



(a) $RE0$ (b) $LA1$ (c) WAP

Fig. 8. The effectiveness of changing the proportion of predicted objects.

Figures 7 a), b), and c) show the classification accuracies of HPSL, KNNS, and TOP-K NNS on WAP, LA1, and RE0 data sets respectively. As can be seen, for most observed numbers of objects with known class labels, the achieved accuracies of the HPSL method are significantly and systematically better than that of KNNS and TOP-K NNS methods. This is due to the fact that the HPSL method has the power to eliminate the isolated data objects that often result in prediction errors in nearest neighbor approaches, such as KNNS and TOP-K NNS. Another observation is that the TOP-K NNS method performs much better than KNNS in terms of accuracies. Indeed, the KNNS method is forced to predict the same number of objects for each labeled object. If noise or outliers are picked, the KNNS method tends to make a wrong prediction. In contrast, the TOP-K NNS method only predicts objects with the top k highest similarity. Hence, it is possible that no prediction will be made for noise or outliers, thus reducing the chance of incorrect predictions.

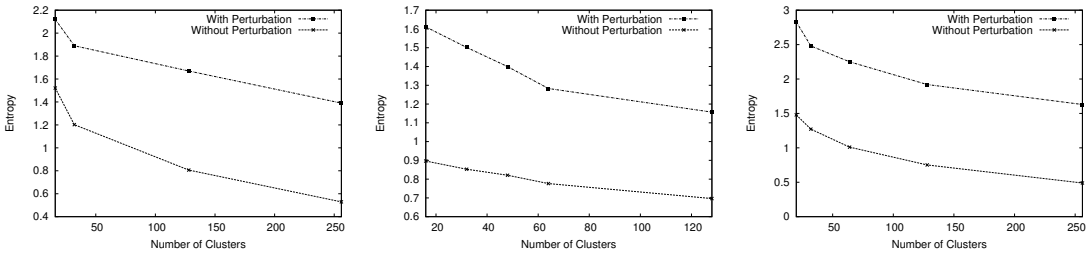
Figures 7 d), e), and f) show the object coverage percentage by HPSL, KNNS, and TOP-K NNS on WAP, LA1, and RE0 data sets respectively. Since KNNS is forced to use each labeled object for prediction, the object coverage of KNNS is always 100%. Also, we observed that the coverage of HPSL is slightly smaller than that of TOP-K NNS.

8.4 Changing the Proportion of Predicted Objects

In this subsection, we show the effect of changing the proportion of predicted objects on the performance of HPSL, KNNS, and TOP-K NNS. In the experiment, we set the number of objects with class labels to be six and did a random sampling to select these six objects. Also, 10 trials were conducted for each observed parameter.

Figure 8 shows the prediction accuracies of HPSL, KNNS, and TOP-K NNS on RE0, LA1, and WAP data sets as the proportion of predicted objects is increased. In the figure, we can observe that the prediction accuracy of HPSL is much better that that of KNNS or TOP-K methods on all three testing data sets. Also, we observed, for all three methods, that there is a downward trend in accuracy when the proportion of estimated instances is increased. This result is

not surprising, since we would expect that predictability would decrease when the similarity of objects decreases.



(a) $RE0$ (b) $LA1$ (c) WAP

Fig. 9. The effectiveness of the pseudo-attribute perturbation.

8.5 The Effectiveness of the Pseudo-Attribute Perturbation Technique

In this experiment, we tested the effectiveness of the pseudo-attribute perturbation technique on three real life data sets: RE0, LA1, and WAP. More specifically, our purpose is to show that the pseudo-attribute perturbation technique can introduce the declustering effect among objects in the same category. In other words, this preventive technique tends to invalidate the basic assumption on which semi-supervised learning attacks on multi-relational databases are based.

Figure 9 shows the entropies achieved by the bisecting K-means clustering method [8] before and after the pseudo-attribute perturbation. As can be seen, if no pseudo-attribute perturbation is involved, the entropy is relatively low for different sizes of cluster on all three data sets. This indicates that there is a strong clustering effect in each category for all three test data sets. After we applied the pseudo-attribute perturbation technique, the entropy jumps significantly for all three test data sets, i.e., there is a declustering effect on objects within the same category.

9 Conclusions

In this paper, we demonstrated a new challenge for multi-relational database security from the data mining perspective. More specifically, we presented a framework for illustrating potential privacy leakage in multi-relational databases via a semi-supervised learning. We also introduced a new semi-supervised learning approach, hyperclique pattern based semi-supervised learning method (HPSL).

The hypothesis behind this method is that similar data objects tend to have similar class labels. As demonstrated by our experimental results, the HPSL method can achieve better prediction accuracy than the traditional KNNS and TOP-K NNS approaches. Finally, we proposed a principle for protecting multi-relational databases from such semi-supervised learning attacks. A simple pseudo-attribute perturbation method was also provided. Our experiments showed that the pseudo-attribute perturbation method can reduce the clustering effect in data sets, and thus reducing the risk of semi-supervised learning attacks.

For future work, we plan to investigate other semi-supervised learning techniques from statistics or machine learning. We also plan to investigate additional approaches for protecting database security in the face of semi-supervised learning attacks.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD*, 1993.
2. E.F. Codd. A relational model for large shared data banks. *Comm. ACM*, 13(6):377–387, June 1970.
3. D.E. Denning, S.G Akl, M. Morgenstern, and P.G. Neumanna. Views for multilevel database security. In *IEEE Symposium on Security and Privacy*, 1986.
4. D.E. Denning, T.F. Lunt, R.R. Schell, M. Heckman, and W.R. Shockley. Views for multilevel database security. In *IEEE Symposium on Security and Privacy*, 1986.
5. Pedro Domingos. Prospects and challenges for multi- relational data mining. *SIGKDD Explorations*, 2003.
6. R.P.W. Duin. Classifiers in almost empty spaces. In *Proc. 15th Int. Conference on Pattern Recognition*, 2000.
7. Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, B. Mobasher, and Jerry Moore. Webace: A web agent for document categorization and exploration. In *Proc. of the 2nd Int'l Conf. on Autonomous Agents*, 1998.
8. George Karypis. Cluto: Software for clustering high dimensional datasets. http://www.cs.umn.edu/karypis.
9. D. Lewis. Reuters-21578 text categorization text collection 1.0. In *http://www.research.att.com/ lewis*.
10. M. F. Porter. An algorithm for suffix stripping. In *Program, 14(3)*, 1980.
11. S. Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitions. *IEEE TPAMI*, 13(3):252–264, 1991.
12. Matthias Seeger. Learning with labeled and unlabeled data. In *Technical Report, University of Edinburgh*, 2001.
13. Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, August 2000.
14. Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Proc. of AAAI: Workshop of Artificial Intelligence for Web Search*. AAAI, 2000.
15. TREC. In *http://trec.nist.gov*.
16. H. Xiong, P. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. of the 3rd ICDM*, pages 387–394, 2003.