Privacy Preserving Decision Tree Mining from Perturbed Data

Li Liu
Global Information Security
eBay Inc.
liiliu@ebay.com

Murat Kantarcioglu and Bhavani Thuraisingham
Computer Science Department
University of Texas at Dallas
muratk, bhavani.thuraisingham@utdallas.edu

Abstract

Privacy preserving data mining has been investigated extensively. The previous works mainly fall into two categories, perturbation and randomization based approaches and secure multi-party computation based approaches. The earlier perturbation and randomization approaches have a step to reconstruct the original data distribution. The new research in this area adopts different data distortion methods or modifies the data mining techniques to make it more suitable to the perturbation scenario.

Secure multi-party computation approaches which employ cryptographic tools to build data mining models face high communication and computation costs, especially when the number of parties participating in the computation is large. In this paper, we propose a new perturbation based technique. In our solution, we modify the data mining algorithms so that they can be directly used on the perturbed data. In other words, we directly build a classifier for the original data set from the perturbed training data set.

1. Introduction

Due to increasing concerns related to privacy, various privacy-preserving data mining techniques have been developed to address different privacy issues [17]. These techniques usually operate under various assumptions and employ different methods. In this paper, we will focus on the perturbation method that is extensively used in privacy preserving data mining.

In this paper, we propose a new method that we build data mining models directly from the perturbed data without trying to solve the general data distribution reconstruction as an intermediate step. More precisely, we propose a modified C4.5 [14] decision tree classifier that can deal with perturbed numeric continuous attributes. Our privacy preserving decision tree C4.5 (PPDTC4.5) classifier uses perturbed training data, and builds a decision tree model, which could be used to classify the original or perturbed data sets. Our experiments have shown that our PPDTC4.5 classifier can obtain a high degree of accuracy when used to classify the original data set.

The paper is organized as follows: Section 2 describes related work. Section 3 introduce a privacy metric system used to measure privacy in our work. Section 4 shows the construction of the decision tree. Section 5 describes how to build Naive Bayesian models from the perturbed data sets. In section 6, we explain our PPDTC4.5 in detail. Section 7 presents our experimental results. In section 8 we conclude with a discussion of future work.

2. Related Work and Motivation

Previous work in privacy-preserving data mining has addressed two issues. In one, the aim is to preserve customer privacy by perturbing the data values [3]. In this scheme random noise data is introduced to distort sensitive values, and the distribution of the random data is used to generate a new data distribution which is close to the original data distribution without revealing the original data values. The estimated original data distribution is used to reconstruct the data, and data mining techniques, such as classifiers and association rules are applied to the reconstructed data set. Later refinement of this approach has tightened estimation of original values based on the distorted data [2]. The data distortion approach has also been applied to Boolean values in research work [7, 16, 6].

Perturbation methods and their privacy protection have been criticized because some methods may derive private information from the reconstruction step [9]. Different to the original noise additive method in [3], many distinctive perturbation methods have been proposed. One important category is multiplicative perturbation method. In the view of geometric property of the data, multiplying the original data values with a random noise matrix is to rotate the original data matrix, so it is also called rotated based perturbation. In [4], authors have given a sound proof of "Rotation-invariant Classifiers" to show some data mining tools can

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

be directly applied to the rotation based perturbed data. In the later work [11], Liu et al have proposed multiplicative random projection which provided more enhanced privacy protection. There are some other interesting techniques, such as condensation based approach [1], matrix decomposition [20] and so on. As pointed out in [12], these recently research on perturbation based approaches apply the data mining techniques directly on the perturbed data skipping the reconstruction step. Choosing the suitable data mining techniques is determined by the method which noise has been introduce. To our knowledge, very few works focus on mapping or modifying the data mining techniques to meet the perturbation data needs.

The other approach uses cryptographic tools to build data mining models. For example, in [10], the goal is to securely build an ID3 decision tree where the training set is distributed between two parties. Different solutions were given to address different data mining problems using cryptographic techniques (e.g., [5, 8, 18]). This approach treats privacy-preserving data mining as a special case of secure multi-party computation and not only aims for preserving individual privacy but also tries to preserve leakage of any information other than the final result. But when the numbers of parties become bigger, the communication and computation cost grow exponentially.

Our proposed approach is a modified C4.5 decision tree algorithm [14] and adopt noise additive method. Our approach is suitable for the scenarios where many parties want to perform data mining, but each of them only has a small portion of the data. To get the global data mining patterns, the various parties must share their data, but each party has its privacy and security concerns. Our approach is the solution for such a situation. In our approach, each party perturbs its data according to the distribution of a pre-set random noise, and sends its perturbed data to the data miner. The data miner collects the perturbed data sets from each party, and also knows the distribution used to perturb the data. Based on this information, the data miner builds a classifier, and returns the classifier to every participating party. Then each party can use this classifier to classify its data. In this case, each party only knows its own data and the classifier, and it does not have any knowledge about the data of others. The data miner only has access to the perturbed data and the distribution of the noise data. This way, privacy is preserved, and the communication and computation costs for each party are minimized.

3. Privacy Metrics

In the work [2], Agrawal and Aggarwal have proposed a privacy measure based on differential entropy. We briefly repeat the ideas here. The differential entropy $h(A)$ of a random variable A is defined as follows:

$$h(A) = -\int_{\Omega_A} f_A(a) log_2 f_A(a) da \quad (1)$$

where Ω_A is the domain of A. Actually $h(A)$ is a measure of uncertainty inherent in the value of A in the statistics. Agrawal and Aggarwal [2] based on this, proposed that the privacy measure inherent in the random variable A as $\Pi(A)$.

$$\Pi(A) = 2^{h(A)} \quad (2)$$

For example, a random variable U distributed uniformly between 0 and a has privacy $\Pi(U) = 2^{log_2(a)} = a$. Thus if $\Pi(A) = 1$, then A has as much privacy as a random variable distributed uniformly in an interval of length 1. Furthermore if $f_B(x) = 2f_A(2x)$, then B offers half as much privacy as A. This can be easily illustrated as, a random variable uniformly distributed over $[0, 1]$ has half as much privacy as a random variable uniformly distributed over $[0, 2]$. In [2] Agrawal and Aggarwal have also defined conditional privacy and information loss. For more detail please refer the original work [2].

We choose this privacy measure in our work to quantify the privacy in our experiments.

4. Overview of Decision Tree Construction

We propose a modified C4.5 decision tree classifier, which builds the decision tree from perturbed data, and can be used to classify both the original and the perturbed data. The idea behind this approach is the following: when we consider the splitting point of the attribute, we consider the bias of the noise data set as well. We calculate the bias whenever we try to find the best attribute, the best split point and partition the training data.

The C4.5 algorithm is an extension of the ID3 algorithm, and proposed by Quinlan in [14]. After years of improvement, C4.5 algorithm is one of the best algorithms in handling numeric continuous attributes [15]. It finds the best splitting attribute and the best splitting point of the numeric continuous attributes.

4.1. Splitting Criterion

Splitting criterion is very important in building a decision tree. It decides which attribute to use for the splitting, and for the numeric continuous attribute, and also determines which value is used for this splitting. It determines whether or not a decision tree is efficient. It dramatically affects the classification accuracy. ID3 uses information gain as splitting criterion. C4.5 algorithm uses information gain ratio which takes the number of branches into account when examining an attribute. The formulas of information gain and gain ratio are given as follows:

$$Info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{\mid S \mid} \times log_2(\frac{freq(C_j, S)}{\mid S \mid}) \quad (3)$$

$$Info_{Test_A}(S) = \sum_{i=1}^{n} \frac{\mid S_i \mid}{\mid S \mid} \times Info(S_i) \quad (4)$$

$$gain(Test_A) = Info(S) - Info_{Test_A}(S) \quad (5)$$

$$splitInfo(Test_A) = -\sum_{i=1}^{n} \frac{\mid S_i \mid}{\mid S \mid} \times log_2(\frac{\mid S_i \mid}{\mid S \mid}) \quad (6)$$

$$gainRatio(Test_A) = \frac{gain(Test_A)}{splitInfo(Test_A)} \quad (7)$$

Let S be the training set, $\mid S \mid$ is the number of instance in S, and $freq(C_i, S)$ is the number of instance that belongs to class i where i from 1, to n. $\mid S_i \mid$ is the number of instance in the category S_i. $Test_A$ is the attribute chosen.

4.2. Discretizing Continuous Attributes: Binary Split Approach

C4.5 algorithm is also designed to handle numeric attributes. Instead of using the fix range, C4.5 algorithm searches among possible split points to find the best split point. Let us assume that a numeric attribute A_i has the values $\{x_1, x_2, ..., x_m\}$ in increasing order. C4.5 partitions the instances into two groups S_1 and S_2 where S_1 consists of the values up to and including x_j, and S_2 consists those that have values greater than x_j. For each of these partitions, C4.5 computes the gain ratio and chooses the partition that maximizes the gain ratio.

4.3. Stopping Criteria

The stopping criteria decides when to stop growing a decision tree. In C4.5 Algorithm, the tree stops growing when one of the two criteria is met. One is that all the instances at the node have the same class label C_i; we say this node is pure. Another is when the number of instances at the node is less than or equal to a pre-set threshold number; we say this number is minimum instance number of the node. We use the same stopping criteria in our privacy preserving decision tree C4.5 (PPDTC4.5) too.

5. Naive Bayes Classifier Construction over Perturbed Data

As stated in [12], Naive Bayes classifier can be applied directly on the perturbed data. For the completeness, we briefly describe here.

5.1. Naive Bayes Classifier Overview

The Naive Bayes classifier labels a new instance by assigning the most probable class value. Besides, it assumes that attribute values are conditionally independent given the class value in order to simplify the estimation of the required probabilities. Using the above assumptions, Naive Bayes classifier selects the most likely classification C_{nb} as[13]

$$C_{nb} = argmax_{C_j \in C} P(C_j) \prod_i P(X_i|C_j) \quad (8)$$

where $X = X_1, X_2, ..., X_n$ denotes the set of attributes, $C = C_1, C_2, .., C_d$ denotes the finite set of possible class labels, and C_{nb} denotes the class label output by the Naive Bayes classifier.

Clearly, we need to calculate the probabilities $P(X_i = x|C_j)$ used in the Equation 8 based on the training data. In practice, for numeric attributes, $P(X_i = x|C_j)$ is estimated by using Gaussian distribution $N(\mu_{ij}, \sigma^2_{ij}$. The required parameters, $\mu_{ij} = E(X_i|c_j)$ and $\sigma^2_{ij} = Var(X_i|C_j)$, are estimated by using the training data. In Section 5.2, we show how to estimate the parameters, μ_{ij} and σ^2_{ij}, using the perturbed training data.

5.2. Over Perturbed Numeric Data

We need to estimate μ_{ij} and σ^2_{ij} for each attribute X_i and for each class label C_j using the perturbed numeric data to construct a Naive Bayes classifier. In the perturbed data case, instead of the original attribute value X_i, we only see the $W_i = X_i + R$ values. Let w^t_{ij} be the i^{th} attribute value of the t^{th} training data instance with class label C_j. In addition, we assume that there are n instances with class label C_j.

We also know that $w^t_{ij} = x^t_{ij} + r^t_{ij}$ where r^t_{ij} is the randomly generated noise with mean zero and known variance σ^2_R. Using the above facts, we can show that the expected value of $\bar{w}_{ij} = \frac{1}{n} \cdot \sum_{t=1}^{n}(w^t_{ij})$ is equal to μ_{ij}.

Since the sample variance $S^2 = \frac{1}{n-1} \cdot \sum_{t=1}^{n} \left(w^t_{ij} - \bar{w}_i j\right)^2$ has an expected value $\sigma^2_{ij} + \sigma^2_R$, we can use S^2 and the known σ^2_R to estimate the σ^2_{ij} (i.e. use $S^2 - \sigma^2_R$ to estimate σ^2_{ij}).

As a result, as long as we do not change the class labels, we can directly construct Naive Bayes classifier from the perturbed data. Even more, since the parameter estimations done by using the perturbed data and the original data have the same expected values, we should be able to get similar classification accuracy in both cases.

To verify the above intuition, we have performed some experiments using the Naive Bayes classifier from the WEKA machine learning toolkit[19]. Using the same data set with all six numeric continuous attributes, we construct the Naive Bayes classifier from the original data set, and

we get 79.7% classification accuracy. Similarly, if we directly construct the Naive Bayes classifier from the perturbed training data set and test it on the perturbed test set, we get 78% classification accuracy. As expected, the two classification accuracy values are very close.

6. Privacy Preserving Decision Tree C4.5 (PPDTC4.5)

In this section we will describe how to build a decision tree classifier from the perturbed training data set. We will show threshold and random path selection two different ways to build classifiers. The threshold algorithm gives reasonably good accuracy for classifying the original data set. The random path selection algorithm uses the probability as weight and finds good splitting points for the attributes. However, as we will see that randomly selecting the path to partition the training data set does not build a good decision tree classifier. We include the random path selection algorithms mainly for comparison purposes as well as to provide some directions for future work to building decision tree classifiers to classify perturbed data set. We believe that with proper improvement on tight bounds of random variables R, the performance of random path selection algorithm can be improved.

The goal is to build a decision tree model from perturbed data which can classify the original data set or perturbed data set accurately. In our case, we do not know the original x_j values due to added noise. Instead, we observe $w_j = x_j + R$ where R is random noise which we know its distribution. Clearly for a split point t for attribute A_i, if we know the x_j values, we use the following rule to split instances into S_1 and S_2:

$$split(x_j, t) = \begin{cases} S_1, & x_j \leq t \\ S_2, & x_j > t \end{cases} \tag{9}$$

Since we do not know the x_j values, therefore we can only calculate the probability of w_j belongs to S_1 that for given split point t and w_j. Note that $Pr\{w_j \in S_1\} = Pr\{w_j - R \leq t\} = Pr\{w_j - t \leq R\}$. Since we know the cumulative distribution function of R, we can calculate the probability easily. Let $p_{S_1}(w_j|t)$ is the probability that w_j belongs to S_1 given the split point t and w_j. Similarly define $p_{S_2}(w_j|t) = 1 - p_{S_1}(w_j|t)$. In general, let us define $p_S(w_j|t)$ is the probability that w_j belongs to set S. Now instead of splitting according to equation (9), we can use the $p_S(w_j|t)$ values to estimate the best split point and partition the w_j just as the original data x_j would have been partitioned.

6.1. Splitting Criterion Using Threshold

We can calculate the probability $p_{S_1}(w_j|t)$, for given split point t for each w_j value. This $p_{S_1}(w_j|t)$ value indicates the likelihood $w_j \in S_1$. We can set a threshold, and count the number of w_j having a class label C_j and $p_{S_i}(w_j)$ value greater than the threshold. This can be expressed in the form of equation as follows:

$$freq'(C_j, S_i) = \sum_{w_j \in S_i} \left(I_{w_j \in C_j}, p_{S_i}(w_j) > threshold\right) \tag{10}$$

In the above equation, we calculate the frequency of a class value by using the $p_{S_i}(w_j)$. $I_{w_j \in C_j}$ is an indicator function and returns 1 if w_j has a class label C_j. The equation (3) is changed to as follow:

$$Info(S) = -\sum_{j=1}^{k} \frac{freq'(C_j, S)}{\mid S \mid} \times log_2\left(\frac{freq'(C_j, S)}{\mid S \mid}\right) \tag{11}$$

Using above equation (11), plus equation (4), (5), (6) and (7) we can find the best splitting attribute and the best splitting point for the numeric continuous attribute by maximizing the $gainRatio$.

6.2. Splitting Training Data by Threshold

The splitting criteria for the training data is straightforward.

$$splitThreshold(w_j, t) = \begin{cases} S_1, & p_{S_1}(w_j|t) > threshold \\ S_2, & p_{S_1}(w_j|t) \leq threshold \end{cases} \tag{12}$$

Noticed that the condition $p_{S_1}(w_j|t) > threshold$ is equivalent to the condition $p_{S_2}(w_j|t) \leq (1 - threshold)$, vice versa. For example, condition $p_{S_1}(w_j|t) > 0.2$ is equivalent to the condition $p_{S_2}(w_j|t) \leq 0.8$.

Pseudo-code is shown in the Algorithm 1.

Using the threshold approach seems like a simple method, but setting the appropriate threshold that will lead to a successful classifier is not trivial. By definition $p_{S_1}(w_j|t)$ is the probability that w_j belongs to S_1 given the split point t and w_j. When the split point t and perturbed instance w_j are given, the only uncertainty is the random noise R. In another words, the choice of threshold is related to the distribution of the random noise. In our experiments we have used both the Uniform and Gaussian distributions for random noise. The thresholds are different for these two kinds of distributions. For Gaussian distribution, when the threshold is set to 0.30, the classifier gives higher accuracy; for Uniform distribution, when the threshold is set to 0.50, the classifier gives higher accuracy. The experimental results are discussed in next section.

Another good way to decide the threshold is to keep it flexible, just like what the C4.5 algorithm does to find the best splitting point t for the numeric continuous attribute. This would lead to good results, but the computation costs are increased.

```
1  Partition (Node N) ;
2  if Stopping Criteria is Met then
3  |  return;
4  else
5  |  Using ThresholdSplittingCriterion
   |  Compute the Best Attribute Best_A and the
   |  Best Splitting Point t ;
6  |  for each Instance w_j in Node N do
7  |  |  Calculate the p_{S_1}(w_j|t) = p_1
8  |  |  Based on the best splitting attribute Best_A
9  |  |  and the best splitting point t;
10 |  |  if p_1 > threshold then
11 |  |  |  addChild(N.leftChild, w_j);
12 |  |  else
13 |  |  |  addChild(N.rightChild, w_j);
14 |  |  end
15 |  end
16 end
17 Partition (N.leftChild);
18 Partition (N.rightChild);
```

Algorithm 1: Partition Training Instances Using Threshold

6.3. Classifying the Original Instance

Using the algorithm described in the previous two subsections, we choose the splitting attributes and splitting points that actually belong to the original data. We also partition the perturbed training data as decided by our estimation of its original values. This way we can build a decision tree classifier for the original data. Then when we use it to classify the original data, we just classify the data based on whether an attribute value is less than or greater than the certain attribute splitting point. The pseudo-code of classifying the original data is shown in Algorithm 2.

6.4. Splitting Criterion Using Probability as Weight

We use the $p_{S_1}(w_j|t)$ and $p_{S_2}(w_j|t)$ values as weight to rewrite all the equations as follows:

$$freq''(C_j, S_i, S) = \sum_{w_j \in S} \left(I_{w_j \in C_j} \times p_{S_i}(w_j) \right) \quad (13)$$

In the above equation, we calculate the frequency of a class value by using the $p_{S_i}(w_j)$ as weights. $I_{w_j \in C_j}$ is an indicator function and returns 1 if w_j has a class label C_j. In order to normalize, we need to calculate the sum of the total weights (i.e., sum of the all $p_{S_i}(w_j)$ values)

$$w(S_i, S) = \sum_{w_j \in S} (p_{S_i}(w_j)) \quad (14)$$

```
1  for each Instance x_j in X from the root node do
2  |  Classify(Node N, Instance x_j);
3  |  if N is a leaf node then
4  |  |  use the rule given at the leaf
5  |  |  return class value;
6  |  else
7  |  |  Based on the attribute A_i used in N
8  |  |  and the split point t;
9  |  |  if x_j(A_i) ≤ t then
10 |  |  |  return Classify(N.leftChild, x_j);
11 |  |  else
12 |  |  |  return Classify(N.rightChild, x_j);
13 |  |  end
14 |  end
15 end
```

Algorithm 2: Classify Original Instances

Now using the above two definitions, we are ready to redefine the conditional entropy using the $p_{S_i}(w_j$ values.

$$Prob(S_i, S) = \frac{freq''(C_j, S_i, S)}{\mid w(S_i, S) \mid} \quad (15)$$

$$Info'(S_i, S) = -\sum_{j=1}^{k} Prob(S_i, S) \times log_2(Prob(S_i, S)) \quad (16)$$

$$Prob(S_i, S) = \frac{freq''(C_j, S_i, S)}{\mid w(S_i, S) \mid} \quad (17)$$

$$Info'(S_i, S) = -\sum_{j=1}^{k} Prob(S_i, S) \times log_2(Prob(S_i, S)) \quad (18)$$

Similarly, we need to update information gain of choosing an attribute A with split point t.

$$Info'_{Test_A}(S) = \sum_{i=1}^{2} \frac{\mid w(S_i, S) \mid}{\mid S \mid} \times Info'(S_i, S) \quad (19)$$

Also, in calculating the $splitInfo$, we need to use $w(S_i, S)$.

$$splitInfo'(Test_A) = -\sum_{i=1}^{n} \frac{\mid w(S_i, S) \mid}{\mid S \mid} \times log_2(\frac{\mid w(S_i, S) \mid}{\mid S \mid}) \quad (20)$$

Now we can plug the above modified definitions to original tests to choose the split point and attribute.

$$gain'(Test_A) = Info(S) - Info'_{Test_A}(S) \quad (21)$$

$$gainRatio'(Test_A) = \frac{gain'(Test_A)}{splitInfo'(Test_A)} \quad (22)$$

Now we can use the equation (22) to choose a split point and an attribute. Please note that the main difference between the original equations used in C4.5 and ours is that we use the probability of being in a certain partition as a weight in the formulas. The rational behind this modification is the following: If we write the original formulas, in clear form, we can see that there exists an implicit indicator function that assigns 0 or 1 based on the membership instances to set S_1 and set S_2. Since we can not be sure whether a certain instance is in S_1 or in S_2, we use the probability of being in S_1 or in S_2 as a weight.

6.5. Splitting Training Data Set Using Random Path Selection

This method is an alternative way to split the training data into two after finding a split point and an attribute. Again, in our implementation, we use a random split based on the $p_{S_1}(w_j)$ and $p_{S_2}(w_j)$ values.

$$splitRandom(w_j, t) = \begin{cases} S_1, & \text{with prob. } p_{S_1}(w_j) \\ S_2, & \text{with prob. } p_{S_2}(w_j) \end{cases} \quad (23)$$

Pseudo-code is shown in Algorithm 3.

6.6. Classifying the Perturbed Instance Using Random Path Selection

For each test instance w_j in the perturbed data set W and for each chosen split point in the constructed tree, we calculate $p_{S_1}(w_j|t) = p_1$. Next we place the instance to the left child of the node with prob p_1 and to the right child with prob $1 - p_1$. We continue with this until we reach a leaf node. The Pseudo-code is shown in Algorithm 4.

7. Experimental Results

In our experiments, we use the data extracted from the census database 1994 ("Census Income" or "Income"), which can be downloaded from University of California, Irvine (UCI), machine learning database repository [1]. This data set has fourteen attributes, six continuous and eight nominal. It altogether has 48842 instances, separate as training data 32561 instances and testing data 16281 instances. The data is used to predict whether the income exceeds $50K$ annually. We choose this data set to have fair

[1] http://www.ics.uci.edu/ mlearn/MLSummary.html

```
1  Partition (Node N) ;
2  if Stopping Criteria is Met then
3  |  return;
4  else
5  |  Using RandomSplittingCriterion Compute
   |  the Best Attribute Best_A and the Best
   |  Splitting Point t ;
6  |  for each Instance w_j in Node N do
7  |  |  Calculate the p_{S_1}(w_j|t) = p_1
8  |  |  Based on attribute Best_A and splitting
   |  |  point t;
9  |  |  Let R be a uniform random value between
   |  |  [0, 1];
10 |  |  if R ≤ p_1 then
11 |  |  |  addChild(N.leftChild, w_j);
12 |  |  else
13 |  |  |  addChild(N.rightChild, w_j);
14 |  |  end
15 |  end
16 end
17 Partition (N.leftChild);
18 Partition (N.rightChild);
```

Algorithm 3: Partition Training Instances Using Random Criteria

```
1  for each Instance w_j in W from the root node do
2  |  Classify(Node N, Instance w_j);
3  |  if N is a leaf node then
4  |  |  use the rule given at the leaf
5  |  |  return class value;
6  |  else
7  |  |  Calculate the p_{S_1}(w_j|t) = p_1
8  |  |  Based on the attribute A_i used in N
9  |  |  and the split point t;
10 |  |  Let R be a uniform random value between
   |  |  [0, 1];
11 |  |  if R ≤ p_1 then
12 |  |  |  return Classify(N.leftChild, w_j);
13 |  |  else
14 |  |  |  return Classify(N.rightChild, w_j);
15 |  |  end
16 |  end
17 end
```

Algorithm 4: Classify Noisy Instances

comparison with reconstruction based techniques that require relatively large data sets. Since, in this paper, we focus on the numeric continuous attributes, we only keep the six numeric continuous attributes in our data set. Also, for efficiency purposes, we randomly choose 10000 instances from the training data set, and keep all the instances in the testing data set.

We use the noise addition frame work proposed in [3], and add both Gaussian and Uniform random noise to each attribute. When using Gaussian random noise, we know that the variance σ^2 can dramatically affect the results. We use four different Gaussian distribution noise data with different variance values. To quantify the relative amount of noise added to actual data, we used the Signal-to-Noise Ratio (SNR), that is the ratio of variance σ^2 of actual data to variance σ^2 of noise data [9]. We also use the privacy measure mentioned in section 3, to quantify the privacy loss. The table 1 shows the five perturbed data sets with their SNR values and privacy measures. In our experiments, we only use one data set which perturbed by uniform noise, shown as data5. We can see from table 1, when the SNR value is higher, the variance σ^2 of noise data is lower, thus the perturbed data preserves less privacy. The uniform distributed noise is generated by a given data range. We can calculate the SNR for each attribute for uniform noise data, in our experimental data set, the SNR values for six attribute are 1.3, 2.7, 1.2, 53.6, 1.9 and 1.1 respectively.

Table 1. Privacy measure of different data sets

	Data1	Data2	Data3	Data4	Data5
Noise Distribution	Gaussian	Gaussian	Gaussian	Gaussian	Uniform
SNR	1.7	1.3	1.0	0.5	N/A
Privacy loss	0.2183	0.1909	0.1619	0.1026	0.2604

7.1. Local vs. Global Data Mining

First note that by local data mining, each participant mines its own data. By global data mining, we mean that the participants share the data and mine to obtain global patterns. As we have mentioned before, our approach is suitable for the scenarios where many parties are participating to perform global data mining without compromising their privacy. The data sets distributed among each party can be horizontally or vertically partitioned. Horizontally partitioned data means the instances are split across the parties, and vertically partitioned data means the attributes are split across the parties. Experimental results show that for both types of partitioning local data mining results are less accurate compared with those obtained from global data mining. This supports the fact that extracting information from globally shared data is better.

Table 2. C4.5 decision tree classifier accuracy over horizontally partitioned data

Accuracy(%)	data1	data2	data3	data4	data5
50 Instance	73.47	74.23	73.13	73.23	73.63
100 Instance	78.54	73.33	77.43	78.13	75.63
Accuracy(%)	data6	data7	data8	data9	data10
50 Instance	74.63	72.37	74.23	76.27	77.73
100 Instance	78.77	75.4	78.13	76.2	77.77

We use the data set described in the previous sub-section with six attributes. We randomly choose instances to form small data sets with different sizes, denoted as group 1, group 2 to group 10. We apply standard C4.5 classifier on these data sets, and the accuracy numbers are shown in Table 1. It is clear that when the number of instances are increased, the C4.5 decision tree algorithm has better performance.

Table 3. C4.5 decision tree classifier accuracy over vertically partitioned data

Accuracy(%)	2 Attributes	3 Attributes	4 Attributes
5K Instance	77.54	77.84	78.9
32K Instance	78.06	78.51	79.05

Similarly we have removed some attributes from the "Income" data set, and then applied standard C4.5 decision tree classifier on the new data sets, and the accuracy of the classification results are shown in Table 2. We can see that when the attribute number is increased the C4.5 decision tree algorithm performs better.

7.2. Reconstruction Based Approaches Results

For comparison purposes, we report the data mining results obtained by using original data distribution reconstruction methods. We apply two notable reconstruction techniques to the perturbed data set. The first technique is Bayesian inference estimation (BE) based approach proposed by Agrawal et al [3]. The second technique is the principal component analysis (PCA) based approach proposed by Kargupta et al [9]. Please refer to the original work for the algorithms' details.

We apply the two techniques on the five data sets. We first reconstruct the original distribution, and then use this

Table 4. Data Mining accuracy of applying data mining techniques directly on 10k perturbed training data set.

Data mining on perturbed data set					
Decision Tree C4.5 Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on Original	79.61	79.04	77.73	77.32	80.29
Test on Perturbed	78.56	78.14	77.45	77.05	80.47
Naive Bayes Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on Original	78.45	78.21	78.17	77.99	80.45
Test on Perturbed	78.08	77.78	77.46	76.47	80.29

Table 5. Data Mining accuracy with BE based reconstruction technique.

BE based reconstruction technique					
Decision Tree C4.5 Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on BE-recon	91.91	90.50	86.26	93.97	86.35
Test on Original	24.45	25.17	69.32	35.96	78.72
Test on Perturbed	38.29	36.47	58.45	43.63	74.06
Original Data Mining Accuracy (%)				83.40	
Naive Bayes Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on BE-recon	88.68	87.09	84.32	94.59	77.53
Test on Original	26.30	21.66	23.01	23.08	76.46
Test on Perturbed	37.19	39.91	33.09	37.30	49.66
Original Data Mining Accuracy (%)				79.87	

Table 6. BE based reconstruction technique data mining accuracy.

PCA based reconstruction technique					
Decision Tree C4.5 Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on PCA-recon	99.23	98.16	98.34	95.10	99.61
Test on Original	70.31	71.71	72.35	68.42	76.86
Test on Perturbed	54.76	63.49	59.17	61.08	61.49
Original Data Mining Accuracy (%)				83.40	
Naive Bayes Classifier Accuracy (%)					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on PCA-recon	97.83	98.10	97.42	94.33	98.63
Test on Original	66.12	64.29	63.21	59.84	64.81
Test on Perturbed	46.08	41.71	39.07	28.05	51.08
Original Data Mining Accuracy (%)				79.87	

Table 7. Proposed PPDTC4.5 data mining accuracy.

Our Proposed PPDTC4.5 Classifier Accuracy (%)					
PPDTC4.5 Threshold Method					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on Original	80.74	79.69	76.63	77.02	80.29
Test on Perturbed	76.09	76.14	74.41	76.03	80.52
PPDTC4.5 Random Path Selection Method					
Data Set	Data1	Data2	Data3	Data4	Data5
Test on Original	78.72	77.21	77.67	78.01	80.31
Test on Perturbed	78.40	77.77	77.30	77.06	80.32

estimated distribution to build the data mining models. We perform three different tests to compare data mining accuracy. In the first case, we test the classifier on the reconstructed test data; in the second case, we test the classifier on the original test data; and in the third case, we test the classifier on the perturbed test data. The data mining models' prediction accuracy is shown in the table 5 and table 6. As comparison table 4 shows the data mining accuracy obtained directly from the perturbed data sets.

Our results indicate that, both reconstruction techniques fail to produce good data mining models. This result is not surprising, since ,in general, estimating data distributions on finite data is a very hard problem. If we use this original data distribution reconstruction phase as a intermediate step to do privacy preserving data mining, we may not always get good performance results. In the work [12], the authors have investigated three different real world data sets, and the reconstruction based approaches have failed on all those data sets. These results support our motivation of finding direct ways to perform privacy preserving data mining from perturbed data.

7.3. PPDTC4.5 Classifier Accuracy

Using the data sets described earlier, we perform different experiments. Applying WEKA [19] C4.5 algorithm on the original training data set to build the decision tree, and classify the original testing data set, we get 83.40% accuracy. table 4 shows the data mining accuracy when apply data mining tools directly on the perturbed data sets. Table 7 shows data mining accuracy of our proposed PPDTC4.5 algorithms. We can see, when we use our proposed PPDTC4.5 Threshold Method on these five data sets to build the decision tree, and classify on the original data set. We get higher accuracy than which classify on the perturbed data for data 1 and data 2; equivalent accuracy for

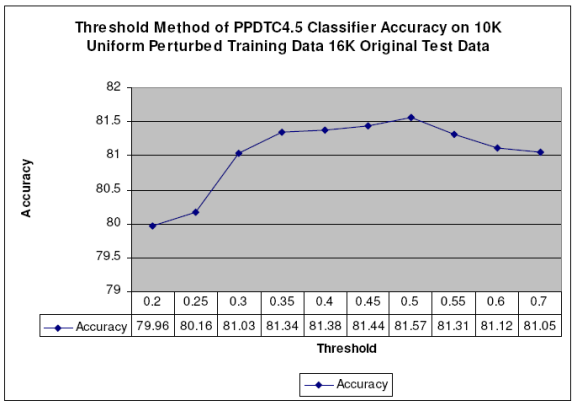

Figure 1. Threshold Method of PPDTC4.5 Classifier Accuracy on 10K Uniform Perturbed Training Data 16K Original Test Data.

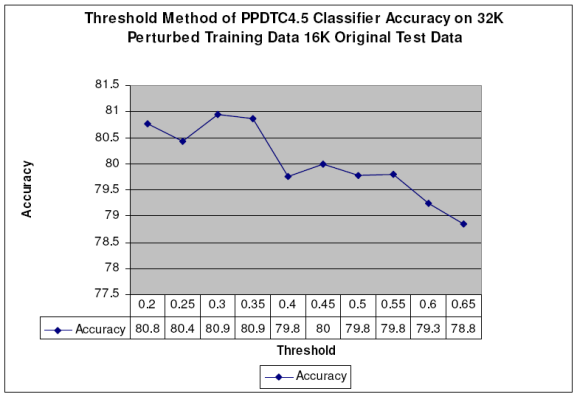

Figure 2. Threshold Method of PPDTC4.5 Classifier Accuracy on 32K Gaussian Perturbed Training Data 16K Original Test Data.

data 5; lower accuracy on data 3 and data4. The reason is SNR value plays an important role here. Table 1 shows different perturbed data sets with different SNR values. When SNR less than 1.0 means the variance σ^2 of noise data greater than the variance σ^2 of actual data. In other words data 3 and data 4 are introduced more noise. Our algorithm get good results when the SNR value is greater than 1.0. When The threshold method builds a decision tree classifier which is not suitable to classify the perturbed data set, because our algorithm estimates the splitting point and partition the training data as the original data would have.

To build a classifier to classify the perturbed data, we use the probability as weight to find the best splitting point, and use random path selection to partition the training data, so far the classifier accuracy has not been improved much compared with directly applying WEKA C4.5 algorithm. The reason is when partitioning the training data set, and classifying the test data set, the random path selection method does not bound the random noise R well. In the future, we would like to find a better way to bound the R value, to build a better classifier for classifying the perturbed data.

As we have seen in our experimental results, our proposed PPDTC4.5 classifiers may not get very excited high accuracy comparing with those obtained from directly applying data mining techniques to the perturbed data sets. But comparing with reconstructed based approaches, our methods obtain very good results. We try to represent the message here is, avoiding to solving the hard distribution problem, in stead, mapping the data mining functions to construct privacy preserving data mining methods. This is a promising direction. Furthermore, our experimental results have also indicated that when huge data set is available, white noise is no longer can prevent data mining tools to abstract patterns. So directly mining the perturbed data set is also a good approach when the data set is big enough.

In the PPDTC4.5 threshold method, we know that choosing different threshold values affect the data mining accuracy. Choosing the threshold to get good data mining results is related to the distribution of the random noise added to the data and the data itself. In our experiments, when using Uniform distribution random noise to distort the data, 0.5 is a good threshold to get a classifier with high accuracy; when using Gaussian distribution random noise to distort the data, 0.3 is a good threshold to get a classifier with high accuracy. The relationship between data mining accuracy and threshold values are shown in figure 1 and figure 2. The best threshold should change from data to data. In other words, this is dependent on the data property.

7.4. Algorithm Complexity

Given n instances, m attributes, and p label values, the number of potential splitting points t of numeric continuous attribute at most is $n-1$. The complexity C4.5 algorithm on training phase is $O(nlgn + tmp)$. Our algorithm evaluates the probability for instance w_j for every given potential splitting point t, which increases the complexity of algorithm in the worst case scenario to $O(ntmp)$. Since our algorithm skips the steps of reconstruction the original distribution for each attribute, the running time is very reasonable comparing with the BE reconstruction algorithm given in [3]. In BE reconstruction algorithm, there is a stop parameter to determine when to stop the calculation of the estimated distribution. The fact is, more loops calculation, more running time and better accuracy of the estimated distribution. In our experiments, based on different choice of the stop parameter, the running time of the BE reconstruction algorithm is raged from three to five times longer than our proposed algorithm running on the same configuration computers.

8. Conclusion

We have proposed a modified C4.5 decision tree classifier which is suitable for privacy preserving data mining. The classifier is built from the perturbed data set, and the model can be used to classify the original data with high accuracy. In the scenarios where many parties are participating to perform global data mining without compromising their privacy, our algorithm decreases the costs of communication and computation compared with the cryptography-based approaches. Our algorithm is based on the perturbation scheme, but skips the steps of reconstructing the original data distribution. The proposed technique has increased the privacy protection with less computation time.

In the future, we will investigate various ways to build the classifiers which can be used to classify the perturbed data set. As we have mentioned before, with a better bound of the random noise data R, using the probability as weighting is an approach that needs further investigation.

As pointed out before, some data mining techniques can be directly applied to perturbed data due to the perturbation process still preserve some nature of the data. Naive Bayes classifier can be directly applied to the additive perturbation data, and Euclidean based data mining tools, e.g. k - Nearest Neighbor Classifier, Support Vector Machines, and Perceptrons Neural Network can be applied to the multiplicative perturbation data. But the data mining accuracy is reduced due to the information loss in the process and some data mining methods themselves may not have good performance. As we know k - Nearest Neighbor is a simple but poor performance classifier. Do we have the flexibility to choose different data mining tools? In this paper we provide a new direction which is modifying data mining functions to suit the perturbed data. This absolutely enable us more choices. Our proposed method skips the reconstructing the original data distribution from the perturbed data. In this way, the method performs privacy preserving data mining without solving the hard distribution problem.

Data mining techniques are used to derive patterns and high level information from data. Data mining results do not cause the violation of privacy. One thing bring to our notice is that when data set is big enough, perform data mining techniques directly on the perturbed data sets, can obtain good data mining accuracy. For example, applying decision tree classifier to additive perturbation data can get good data mining accuracy. This can be observed in our experimental results. Privacy as a security issue in data mining area is still a challenge.

References

[1] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*. ACM, 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.

[4] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *ICDM*, pages 589–592, 2005.

[5] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[6] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD*, pages 505–510, 2003.

[7] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, 2002.

[8] M. Kantarcioglu and C. Clifton. Privately computing a distributed k-nn classifier. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2004.

[9] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, pages 99–106. IEEE Computer Society, 2003.

[10] Y. Lindell and B. Pinkas. Privacy preserving data mining. In M. Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2000.

[11] K. Liu, H. Kargupta, and J. Ryan. Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(1):92–106, January 2006.

[12] L. Liu, M. Kantarcioglu, and B. Thuraisingham. The applicability of the perturbation based privacy preserving data mining for real-world data. *Data and Knowledge Engineering Journal*, 2007.

[13] T. M. Mitchell. *Machine Learning*. mcgraw-hill, 1997.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[15] J. R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Intell. Res. (JAIR)*, 4:77–90, 1996.

[16] S. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, pages 682–693. Morgan Kaufmann, 2002.

[17] B. M. Thuraisingham. Privacy constraint processing in a privacy-enhanced database management system. *Data Knowl. Eng.*, 55(2):159–188, 2005.

[18] J. Vaidya and C. Clifton. Privacy-preserving -means clustering over vertically partitioned data. In *KDD*, pages 206–215, 2003.

[19] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2 edition, 2005.

[20] S. Xu, J. Zhang, D. Han, and J. Wang. Singular value decomposition based data distortion strategy for privacy protection. *Knowl. Inf. Syst.*, 10(3):383–397, 2006.