Technical Report

TR-2006-008

Mining Multiple Private Databases using a Privacy Preserving kNN Classifier

by

Subramanyam Chitti, Ling Liu, Li Xiong

Mathematics and Computer Science

EMORY UNIVERSITY

Mining Multiple Private Databases using a Privacy Preserving kNN Classifier

Subramanyam B Chitti
College of Computing
Georgia Institute of Technology
chittis@cc.gatech.edu

Li Xiong
Department of Mathematics and Computer Science
Emory University
lxiong@mathcs.emory.edu

Ling Liu
College of Computing
Georgia Institute of Technology
lingliu@cc.gatech.edu

ABSTRACT

Data mining technologies are popular for identifying interesting patterns and trends in large amounts of data. With the advent of high speed networks and easily available storage, many organizations are able to collect large amounts of data. On one hand, these organizations would like to mine their data to understand and discover interesting patterns; on the other hand, many legal and commercial reasons may prevent the organizations from sharing their data. In such situations, privacy preserving data mining tools become critical - they allow data from many organizations to be mined securely and with a minimum of information disclosure. In this paper, we present a framework for mining multiple private databases using a privacy preserving k Nearest Neighbor (kNN) classifier. We develop a general model for privacy preserving kNN classification and present algorithms for realizing this model. We specify requirements that all privacy preserving classifiers should strive to achieve and analyze how well our algorithm achieves these requirements. This is the first paper to show how kNN classification can be achieved in a privacy preserving manner. A novel feature of our algorithm is that it offers a trade-off between accuracy, efficiency and privacy. Thus, it can be applied in a variety of problem settings and can meet different optimization criteria.

1. INTRODUCTION

We are entering a highly connected and data intensive world. The information age has enabled many organizations to collect large amounts of data. Many organizations wish to discover and study interesting patterns and trends of both their own private databases and their competitors information bases, such as learning the sales trend and sales patterns of Intel computers in US markets or Asian markets. Therefore, privacy-preserving data mining [3, 25] becomes an important enabling technology for mining data from multiple private databases provided by different and possibly competing organizations. As a field, data mining has introduced new algorithms like association rule learning, and new classification algorithms like Naive Bayes, Decision Tree and k-Nearest Neighbor classifiers, to identify interesting trends in data without assuming any a priori hypothesis on the data.

In privacy preserving data mining across multiple private databases, two or mode nodes owning confidential databases wish to run a data mining algorithm on the union of their data without revealing any unnecessary information. Privacy preserving data mining enables organizations to share statistical information of their private data while simultaneously protecting privileged information residing in individual private databases.

To see why privacy requirements play an important role in distributed data mining, consider the following scenarios [6]. Many insurance companies collect data on disease incidents, seriousness of the disease and patient background. The Center for Disease Control would like to identify disease outbreaks. One way to do this is to mine the data held by the various insurance companies for patterns that are indicative of disease outbreaks. However, commercial and legal reasons prevent the insurance companies from revealing their data. In this situation, it will be important and beneficial to have a distributed data mining algorithm that is capable of identifying potential outbreaks while respecting the privacy requirements of its participants.

Privacy preserving data mining algorithms also play an important role in industry collaborations. Industry trade groups want to identify best practices to help members. However, some practices may be trade secrets. We would like to be able to discover patterns (like "manufacturing using chemical supplies from supplier X have high failure rates") while preserving secrets of individual organizations (like "manufacturing process Y gives low failure rates").

The above real world problems can be modeled as data classification problems, and solved using a k-Nearest Neighbor classifier. A k-nearest neighbor classifier is a conceptually straightforward way of approximating any real valued or discrete valued classification function. In its training phase, a k-nearest neighbor classifier simply stores all the training examples. Generalizing beyond these examples is postponed until a new query instance must be classified. In a

distributed setting, this means that the overhead of training a classifier is avoided until a query instance must be classified. Every time a new query instance is encountered, its relationship to the previously stored instances is examined and a classification is assigned to the new instance. A kNN classifier has many advantages over other machine learning classifiers. Instead of estimating the classification function once for the entire instance space, the kNN classifier can estimate it locally and differently for each new instance to be classified. This is extremely useful in situations in which the databases of nodes are dynamic and the classification function that we wish to determine is very complex and thus cannot be represented by a single, global function, but instead can be represented by a collection of less complex local approximations.

For instance, to determine whether a pattern of disease is indicative of a outbreak, we can train a kNN classifier to recognize disease outbreak patterns and use it to classify a query pattern as an outbreak or not. For the second problem, the various organizations can use a kNN algorithm to determine the answer to their queries (such as "Is the failure rate higher than 0.5 when manufacturing using supplies from supplier X") without revealing private information (such as "My failure rate for manufacturing using supplies from supplier X is 0.36"). However, current distributed kNN classifiers are not designed to guarantee the privacy requirements of the participating nodes.

One solution to privacy preserving data classification across multiple private databases using a kNN classifier is to have a trusted third party (TTP). The nodes send their data along with the query to the TTP, which constructs the kNN classifier using the data, classifies the query and sends the results back to all the nodes. However, in practice it is difficult to find a TTP which is trusted by all the nodes. Also, this solution presents a single point of failure which is the TTP. If the TTP is compromised, the privacy of all the nodes is compromised. To overcome the above limitations, we propose a distributed solution to the privacy preserving classification problem.

Another possibility one might consider is to construct a privacy preserving kNN classifier using secure multiparty computation techniques [10], a subject that has received significant attention in cryptography research. However, these techniques have a high communication overhead and are feasible only if the inputs are very small. In a real world scenario, the inputs to the kNN classifier would consist of massive data sets. Thus, generic secure multiparty computation protocols are too inefficient to be applied to these problems. This deficiency of secure multiparty schemes has led to the search for efficient, privacy preserving data mining algorithms. A practical idea is to look for algorithms that can provide a desired level of tradeoff between the accuracy of the classifier constructed and the stringency of the privacy requirements while maintaining efficiency.

With these design objectives in mind, we present a privacy-preserving framework for constructing a kNN classifier across multiple private databases. This framework consists of a general model for privacy preserving kNN classification, a suite of concrete algorithms for realizing this model, and a set of requirements that all privacy preserving classifiers should strive to achieve. We discuss how well our algorithm achieves the specified requirements through analytical study and experimental evaluation. To the best of our knowledge, this is the first paper to show how kNN classification can be achieved in a privacy preserving manner without a centralized trusted third party. Our approach has an important trait − it offers a trade-off between accuracy, efficiency and privacy, allowing our privacy-preserving kNN classifier to be applied in a variety of problem settings and meeting different optimization criteria.

The rest of this paper is organized as follows. In section 2, we present the general issues and design goals in privacy preserving data classification algorithms. In Section 3, we present a model for the construction of a privacy preserving kNN data mining algorithm. In Section 4, we present algorithms for realizing the privacy preserving kNN classifier construction according to our model. In section 5, we analyze our solution and discuss how well we are able to address the issues mentioned in section 2. In Section 6, we present experimental results of the evaluation of our algorithm. We present related work in Section 7 and conclude in Section 8 with some thoughts on future research directions.

2. DESIGN ISSUES

In this section, we identify the design goals that any privacy preserving classification algorithm must seek to fulfill. These design goals also define dimensions along which any such algorithm can be evaluated.

2.1 Problem Definition and Threat Model

Consider a network of n nodes ($n \geq 3$), each having a private database. Assume that we want to train a machine learning classifier on the union of these n databases, under the condition that each node wishes to reveal as little information about its database as possible during the (distributed) training of the classifier and during the possibly distributed classification of the test instances.

We note that in any multi-party computation, a malicious adversary can always alter its input. In the data classification setting, this can be very damaging since the adversary can define its input to be the empty database and thereby gain knowledge of the classifier for free, or can alter its input to invalidate the results of the classifier. In this paper we assume that the nodes participating in our decentralized protocol for computing the kNN classification among n private databases are cooperating to achieve the common goal of data classification. Thus all nodes behave in a semi-honest manner in the sense that all nodes correctly follow the protocol specification, yet attempt to learn additional information about other nodes by analyzing the transcript of messages received during the execution of the protocol. One of our ongoing efforts is to develop a decentralized kNN classification protocol that is resilient against malicious nodes.

2.2 Evaluation Metrics

One of the most important dimensions along which any classification algorithm is evaluated is its *accuracy*. The accuracy of a classification algorithm measures its ability to correctly classify instances that it has not seen before, i.e., not present in the test set. When we are designing a privacy preserving classification algorithm, we are interested in the *relative accuracy* of this algorithm as compared to a non-privacy preserving classification algorithm. Ideally, we would like the accuracy of a privacy-preserving classification algorithm to be at least as high as that of its non-privacy preserving counterpart. We note that the accuracy of a privacy

preserving classifier could be lower if it utilizes randomization to ensure privacy of its computations. In this case, the design challenge is to determine the "right" amount of randomization that the algorithm should insert into its computation such that both the accuracy and privacy requirements of the users can be met.

Another important consideration in designing privacy preserving algorithms is their *efficiency*. Classification algorithms typically operate on databases containing very large amounts of data; thus the many secure multi-party computation protocols proposed earlier, such as [10, 11], are simply too inefficient to be used in these classification algorithms. This is especially the case when the number of nodes involved in the construction of the classifier is large. Given the above scenario, we would like the privacy preserving classification algorithm to allow for a trade-off between the accuracy of the results and the efficiency of the algorithm used.

Finally, any privacy preserving algorithm must meet the privacy requirements of its participating nodes. Ideally, during the training phase of a classification algorithm, no information other than the trained classifier should be revealed to any node. During the classification phase, no information other than the classification of the query instance should be revealed to the nodes. However, achieving the above objectives completely might make the classification algorithm very inefficient. In practice, we would like to design highly efficient classification algorithms while maintaining the minimum information disclosure.

We distinguish between two types of privacy guarantees that have been proposed in the literature: *data anonymity* and *data privacy*. The data anonymity requirement refers to the requirement that nodes may learn the fact that some actual data point is present in the database of the other nodes, but no node should be able to infer which other node this data point belongs to with certainty. The term "data privacy" has been used to refer to a stronger requirement that no node may learn about the value of any other node's data points during the algorithm. We concentrate on achieving *data privacy* in this paper.

Data privacy provides a more stringent privacy guarantee than Data anonymity, and is closer to the privacy guarantees of traditional secure multi-party schemes. There exist situations in which we require a privacy preserving algorithm to guarantee data privacy and not just data anonymity. This is especially the case if given the knowledge that a data point belongs to *some* node, information could be obtained about *which* node this data belongs to by means other than the information leaked by the algorithm. An example illustration of this scenario is as follows: 10 nodes participate in a privacy preserving algorithm. 5 of these nodes are located in the US, 3 nodes are located in Europe and 2 nodes are located in Asia. If, during the execution of the algorithm, nodes learn the value of a data held by *some* other node; they may be able to infer which node this data belongs to from their knowledge of the location of each node.We formalize the above intuition below.

Let $P(d \in node_i | setminusd = v_i)$ be the probability that a data point d belongs to $node_i$ and has the value v_i. This probability may or may not be identical across all nodes. In those cases where this probability is not identical across all nodes, data anonymity has the potential to reveal information about a node's data to other nodes. This is because data anonymity reveals the information that $d = v_i$ and using this, we could infer that it is more probable for $node_i$ to have this data point than $node_j$ depending on certain external knowledge about the nodes. Depending on the user's requirements, this may be unacceptable. Thus algorithms that guarantee data privacy will be needed in these cases.

We summerize below the three performance metrics for privacy preserving classification algorithm over multiple private databases:

1. **Accuracy:** Ideally, the privacy preserving classification algorithm must be as accurate as a classifier designed without privacy constraints.

2. **Efficiency:** The privacy preserving classifier construction must be as efficient as the non-privacy preserving classifier construction algorithm. Ideally, the complexity of the two algorithms must be of the same order, but we allow for a larger constant in the privacy preserving classifier algorithm.

3. **Privacy:** The privacy preserving classifier construction algorithm should reveal as little information as possible about the values held by each individual node during both the training of the algorithm and the classification of new instances.

Thinking of the design space in terms of these three dimensions presents many advantages to the designer of a privacy preserving classifier. At one end of the spectrum, we have the secure multi-party computation protocols, using which we can construct classifiers which are provably secure in the sense that they reveal the least amount of information and have the highest accuracy; however these protocols are very inefficient. At the other end of the spectrum, we have the non-privacy preserving classifier algorithms, which are highly efficient but are not secure. Thus, the challenge is to design an efficient privacy preserving classifier while sacrificing as little as possible on accuracy and privacy.

3. PRIVACY PRESERVING kNN CLASSIFIER MODEL

In this section, we describe the kNN classification problem and discuss how we can solve it in a distributed, privacy preserving manner. We describe the privacy goals of our protocol and show how our classifier model achieves these privacy goals.

3.1 Classification problem and the kNN classifier

A classification problem consists of a set of instances, partitioned into a training set and a test set. Each instance is represented by a vector of attributes, and belongs to a class. A classifier is a function of the attributes that returns a class value. A classifier construction algorithm takes the training set as input, and returns a classifier. The accuracy of a classifier is measured using the test set: The predictions made by the classifier for the instances in the test set are checked against the class of these instances, and the percentage of correct predictions is the accuracy of the classifier.

The k-Nearest Neighbor (kNN) classifier is one of the most basic instance-based classification methods. In an instance

Algorithm 1 Distance-weighted kNN Classification Algorithm

Training Algorithm:

- For each training example $< x, f(x) >$, add the example to the list $training - examples$.

Classification Algorithm:

- Given a query instance x_q to be classified,

1. Let $x_1, x_2, \ldots, x_k$ denote the k instances from $training - examples$ that are nearest to x_q
2. RETURN
   $Classification(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w(d(x_q, x_i)) * \delta(v, f(x_i))$,
   where $\delta(a, b) = 1$ if $a = b$, and 0 otherwise, and w is any function of the distance $d(x_q, x_i)$ between the points x_q and x_i.

based classifier, when a query instance is encountered, similar instances are retrieved from memory and used to classify the query instance. In a kNN classifier, given a query instance, the k nearest (according to a distance function) instances to this query instance are retrieved , and the class assigned to the query instance is the most common class among these instances. The basic distance-weighted kNN algorithm is presented in Algorithm 1.

3.2 Privacy preserving kNN classification

The privacy preserving kNN classification problem is described as follows. There exist n private databases D_1 , D_2 , ... , D_n distributed at n different nodes. We assume that all the databases have the same schema, i.e. data is horizontally partitioned. These n nodes want to train a kNN classifier on the union of their databases. However, each node would like to reveal as little information as possible about its data to the other nodes, during the construction of the classifier (the) *training* phase and while classifying a new query (the) *test* phase.

To solve the privacy preserving kNN classification problem, we need to adapt the basic distance weighted kNN classification algorithm (Algorithm 1) to work in a distributed setting in a privacy preserving manner. One way to do this is to break the kNN classification problem into two subproblems and solve each of them in a distributed, privacy preserving manner.

A kNN classifier views instances as points in a $|A|$-dimensional space, where $|A|$ is the number of attributes of each instance. Given a query instance (point), a kNN classifier uses the k nearest neighbors of the query to classify it. In a distributed setting, the k nearest neighbors of a query point could be distributed among the n nodes. Thus, each node will have some points in its database which are among the k nearest neighbors of the query point. So, for a node to calculate its local classification of the query point, it has to first determine which points in its database are among the k nearest neighbors of this query point.

Let the distance between the query and its k^{th} nearest neighbor be $\triangle$. All points which are closer than $\triangle$ from the query are among the k nearest neighbors of the query. If a node knows $\triangle$, then it can determine which points in its database are among the k nearest neighbors of the query. Thus, the first sub-problem the nodes have to solve is to determine $\triangle$, which is the distance of the query to its k^{th} nearest neighbor.

Once a node knows which points in its database are among the k nearest neighbors of the query, it can calculate its classification of the query instance. Then, the nodes need to combine their local classifications, in a privacy preserving manner, to compute the global classification of the query point over the n private databases.

Thus we can divide the kNN classification problem into the above two sub-problems. We call the first sub-problem "Privacy preserving nearest neighbor selection" and the second sub-problem "Privacy Preserving Classification". Let us denote the query to be classified by x. We summarize the two sub-problems are follows:

1. **Privacy preserving nearest neighbor selection:** In this step, the n databases work together to determine the distance of the k^{th} nearest point (instance) to x among all the points in the union of their databases. Knowing the distance of the k^{th} nearest neighbor helps each node identify all points in its database which are among the k nearest neighbors of x.
2. **Privacy Preserving Classification:** With the knowledge of which points in its local database which are among the k nearest neighbors of x, each node can contribute to the global classification in two steps. First, it calculates the local classification of x. Second, with the local classification of x as input, all nodes engage in a privacy preserving computation of the global classification of x.

3.3 Analysis of the model

The central thesis of our privacy preserving kNN classification model is to divide the problem of classification into two steps, and to ensure that each step is accomplished in a privacy preserving manner. Concretely, given a query instance, a node has to decide which of its data points are among the k nearest neighbors of the query; these are the points which determine the classification of the query. Our model encourages each node to first calculate the distance of the query to its k^{th} nearest neighbor, using only the distances of the query to the node's data points as input. Then, each node can use this information to determine its local classification of the query point, and combine these local classifications in a privacy preserving manner to determine the global classification. This two step division ensures that a node does not have to reveal its data points in order to determine if a point in its database is among the k nearest neighbors of the query.

It is important to note that if executed naively, the above steps can violate the privacy requirements of the individual databases. Given a query point x, in the first step, we should ensure that the instances in a database are not revealed to other databases during the computation of the distance of the k^{th} nearest neighbor to x. In the second step, we should ensure that the local classification of each database is not revealed to other databases during the computation of the global classification.

Before describing concrete algorithms for privacy preserving kNN classification, we emphasize the fact that the model

suggested is one possible model for the construction of a privacy preserving kNN classifier, and, to the best of our knowledge, it is the first such model to be suggested. Other models may be possible. It would be interesting to study what characteristics determine the best model under different privacy requirements.

4. THE ALGORITHM

Based on our privacy preserving kNN classification model, the problem of mining multiple private databases using a kNN classifier can be addressed in two phases. In the privacy preserving nearest neighbor selection phase, each node identifies the points in its database which are among the k nearest neighbors of the query. In the privacy preserving classification phase, the nodes compute their local classifications of the query and combine them to calculate the global classification. We now present algorithms for each phase that help nodes achieve their ojectives in each phase in a privacy preserving manner.

4.1 Privacy preserving nearest neighbor selection

Privacy preserving nearest neighbor selection: Let x be the query instance to be classified. Assume that y_1, y_2, ..., y_k are the k nearest instances to x in that order, and let $d(y_1)$, $d(y_2)$, ..., $d(y_k)$ be the distances of the k nearest points to x. In order to determine the points in their database which are among the k nearest neighbors of x, the nodes need to determine $d(y_k)$. The n databases engage in a secure computation of this value, which is the distance of the k^{th} nearest neighbor in $D_1 \cup D_2 \cup \ldots \cup D_n$to x.

The algorithm to compute $d(y_k)$ in a privacy preserving manner is as follows. Each node calculates the distance of every point in its database from x, and selects the k smallest distances. With these k smallest distances as input, the nodes participate in a privacy preserving computation of the k smallest values among all their inputs. The k smallest values so determined will be the distances of the k nearest points to x among all the points in the n databases. Thus, the k^{th} smallest value will be the distance of the k^{th} nearest neighbor to x among all the points in the union of the n databases.

The crucial step in the above algorithm is the privacy preserving computation of the k smallest values among all the nodes, with each node contributing k values as input to the computation. This can be accomplished by using the "privacy preserving TopK selection" algorithm ($PP-TopK$), recently suggested in [27]. The $PP-TopK$ algorithm determines the "top k" values among n distributed nodes in a privacy preserving manner, with each node providing any number of values as input to the algorithm; the "top k" values are defined by a linear order among all the values held by the nodes. In our case, the linear order is induced on the values held by the nodes by the "less than" relation. We note that it suffices for each node to provide k values as input to the algorithm, since the number of values in the output is at exactly k.

We describe in detail the $PP-TopK$ algorithm introduced in [27]. In the $PP-TopK$ algorithm, nodes are mapped onto a ring randomly; thus each node has a predecessor and successor. We assume secure communication channels between a node and its successor. Each node executes a local algorithm and communicates the result of this algorithm to its successor. The local algorithm is a standalone component that each node executes. An initialization module is designed to select the starting node among the n participating nodes, and initialize a set of parameters used in the local computation algorithms.

Once the starting node is selected, it initializes a global vector of length k and sends this vector to its successor. This vector is initialized to the maximal elements of the total order. In our case, this would be some number larger the maximum distance between any query and any data point. From this point on, each node, upon receiving a vector from its predecessor, executes the local algorithm with inputs as the vector it received and its own vector of k smallest distances, and sends the result of this computation to its successor. This continues for a number of rounds, at the end of which the global vector is broadcast to all the nodes. The global vector at the end of the protocol contains the global k smallest distances.

The local algorithm is a probabilistic algorithm that injects certain amount of randomization into the local computation at each node, such that the probability of data value disclosure at each node is minimized while the eventual result of the protocol is guaranteed to be correct. The randomization injected by each node is a monotonically decreasing function of which round the protocol is currently executing, ensuring that the protocol outputs the correct result after sufficient number of rounds. In our protocol, we define the probability that a node injects randomization in its local computation to be equal to $P_0 * d^r$, where P_0 is the initial randomization probability, r is the current round of the protocol, and d is a randomization factor. The local algorithm that a node executes is shown in 2.

Algorithm 2 Local Algorithm for Top$k$ Protocol (executed by node i at round r)

INPUT: $G_{i-1}(r)$, V_i, OUTPUT: $G_i(r)$
$P_r(r) \leftarrow p_0 * d^{r-1}$a
$G'_i(r) = \text{topK}(G_{i-1}(r) \cup V_i)$
$V'_i \leftarrow G'_i(r) - G_{i-1}(r)$
$m \leftarrow |V'_i|$
if $m = 0$ **then**
  $G_i(r) \leftarrow G_{i-1}(r)$
else
  with probability $1 - P_r(r)$: $G_i(r) \leftarrow G'_i(r)$
  with probability $P_r(r)$:
  $G_i(r)[1:k-m] \leftarrow G_{i-1}(r)[1:k-m]$
  $G_i(r)[k-m+1:k] \leftarrow$ sorted list of m random values from $[G'_i(r)[k]\ ,\ max(G'_i(r)[k] + \delta\ ,\ G_{i-1}(r)[k-m+1])]$
end if

The inputs to the algorithm are (1) the global vector node i receives from its predecessor $i-1$ in round r, denoted as $G_{i-1}(r)$, and (2) its local top$k$ vector, being the k smallest distances in our case, denoted as V_i. The output of the algorithm is the global vector denoted as $G_i(r)$. Note that the global vector is an ordered multiset that may include duplicate values.

The algorithm first computes the correct top$k$ vector, denoted as $G'_i(r)$, over the union of the set of values in $G_{i-1}(r)$ and V_i. It then computes a sub-vector of V_i, denoted as V'_i, which contains only the values of V_i that contribute to the current top$k$ vector $G'_i(r)$ by taking a set difference of the set of values in $G'_i(r)$ and $G_{i-1}(r)$. Note that the union and set

difference here are all multiset operations. The algorithm then works under two cases.

Case 1: The number of elements in V_i', m, is 0, i.e. node i does not have any values to contribute to the current top$k$. In this case, node i simply passes on the global top$k$ vector $G_{i-1}(r)$ as its output. There is no randomization needed because the node need not expose its own values.

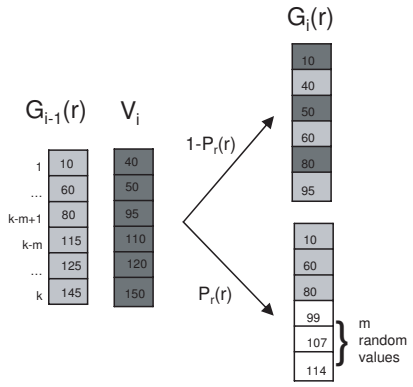


Figure 1: Illustration for Top$k$ Local Algorithm

Case 2: Node i contributes $m(0 < m \leq k)$ values in the current top$k$. Figure 1 gives an illustrative example where $m = 3$ and $k = 6$. In this case, node i only returns the real current top$k$ ($G_i'(r)$) with probability $1 - P_r(r)$. Note that a node only does this once, i.e. if it inserts its values in a certain round, it will simply pass on the global vector in the rest of the rounds.

With probability $P_r(r)$, it copies the first $k - m$ values from $G_{i-1}(r)$ and generate last m values randomly and independently from $[G_i'(r)[k], max(G_i'(r)[k] + \delta, G_{i-1}(r)[k - m+1])]$, where $G_i'(r)[k]$ denotes the kth (last) item in $G_i'(r)$, $G_{i-1}(r)[k-m+1]$ denotes the $k-m+1$th item in $G_{i-1}(r)$, and δ denotes a minimum range for generating the random values. The reason for generating m random values is because only the last m values in the output are guaranteed to be shifted out in a later round when the node inserts its real values if the global vector has not been changed by other nodes. The range is designed is such a way that it decreases the values in the global vector as much as possible while guaranteeing the random values are not smaller than the largest value in the current top$k$ so they will be eventually replaced.

After the execution of the protocol, the global vector contains, with high probability, the k smallest distances to the query instance among all the nodes. Knowing this value, the nodes can proceed to execute the next step in the protocol, namely, *privacy preserving classification.*

4.2 Privacy preserving classification

Privacy Preserving Classification: Given the k^{th} smallest distance $d(y_k)$, each node can determine the points in its database which are within this distance from x, and compute its local contribution to the classification of x as a function of these points. The global classification of x is a function of the n local classifications of x. The nodes engage in a secure computation of the global classification of x, with each node providing as input its local contribution to this computation. The global classification of x is known to every node at the end of the computation.

Recall that the kNN classifier works as follows (Algorithm 1): The classification of the point x is determined by its k nearest neighbors. Each of these neighbors effectively "votes" in favor of its own class, the strength of the vote being determined by its distance to x. We add all these votes and select the class that has the highest vote to be the classification of x.

Thus the local classification of x that is determined by each node is just the class vector of v votes, assuming that the number of classes is v. The i^{th} element of this vector is the amount of vote the i^{th} class received from the points in this node's database which are among the k nearest neighbors of x. However, the class with the local majority at this node may not be the same as the class with the global majority of the votes. In order to classify x, we need a way to determine the class with the global majority of the votes. Our solution to this problem is for each node to compute its local classification vector; and then to participate in a privacy preserving term-wise addition of these local classification vectors to determine the global classification vector. Once each node knows the global classification vector, it can find the class with the global majority of the vote by determining the index of the maximum value in the global classification vector. Thus the nodes can determine the classification of x without disclosing its local classifications.

In the above step, one may use any privacy preserving addition protocol. For sake of completeness, we briefly describe one privacy preserving addition protocol, suggested in [7]. Assume that there are n nodes, each having a value v_i. The nodes arrange themselves into a ring. The first node selects a random number and transmits this to its successor. From this point on, every node adds its value to the value it receives and passes it on. When the first node gets back a value, it adds the difference between the initial random number and its own value to this value to determine the sum of the n values. It then broadcasts the sum to all the nodes. This is the simplest privacy preserving summation protocol we are aware of. For improvements to this protocol to make it robust against collusions, and for a proof of privacy, please refer to [7]. We summarize the privacy preserving addition protocol in Algorithm 3.

4.3 Privacy Preserving kNN classification

Having presented algorithms for the two sub-problems in our model, we now show how to integrate the two solutions to build a privacy preserving kNN ($PP - kNN$) classifier:

1. Given an instance to be classified x, each node computes the distance of every point in its database from x, and selects the k smallest distances. These k values are stored in a local distance vector.

2. The n nodes participate in the privacy preserving top-k selection (Algorithm 2), with inputs as their local distance vectors. At the termination of the algorithm, each node knows the distances of the k nearest neighbors to x in the union of their databases.

3. Each node can determine the distance of the k^{th} nearest neighbor to x from the information above. We call

Algorithm 3 Privacy Preserving Addition of local classification vectors

Input: The local classification vectors lcv_i of each node.
Output: A global classification vector $gcv = \sum_{i=1}^{n} lcv_i$

- The nodes are arranged in a ring, and a random starting node is chosen.
- The starting node initializes a global classification vector (gcv) to a random vector rv, and sends it to its successor.
- Every node, upon receiving a vector vi from its predecessor, transmits $lcv_i + vi$ to its successor.
- When the starting node receives a vector vi from its predecessor, it broadcasts ($vi - rv + lcv$) to all the nodes.

this distance Δ.

4. We assume that there are v classes. Each node calculates a local classification vector (lcv) as follows: $\forall 1 \leq i \leq v$, $lcv(i) = \sum_y w(d(x,y)) * \delta(f(y), i) * [d(x,y) \leq \Delta]$, for all points y in its database, where $d(x,y)$ is the distance between x and y and $\delta(a,b) = 1$ if $a = b$ and 0 otherwise, $[p]$ for a predicate p evaluates to 1 if the predicate is true, and evaluates to 0 otherwise.
5. The nodes use the privacy preserving addition protocol (Algorithm 3) to do an element-wise addition of their local classification vectors to calculate the global classification vector gcv: $gcv(i) = \sum_{j=1}^{n} lcv_j(i)$.
6. Each node assigns the classification of x as $classification(x) \leftarrow \arg\max_{i \in V} gcv(i)$.

5. ANALYSIS

In this section, we analyze the privacy preserving kNN algorithm using the three performance metrics – *relative accuracy, efficiency* and *privacy.*

5.1 Accuracy

The relative accuracy of our privacy preserving algorithm is determined by the accuracies of its two steps. The accuracy of the *privacy preserving nearest neighbor selection* step is determined by the accuracy of the $PP-TopK$ algorithm. As $PP-TopK$ is a randomized algorithm, we can only present probabilistic guarantees on its accuracy.

The output of the $PP-TopK$ algorithm is a vector containing k distances. The probability that this vector contains the smallest k distances can be calculated as follows:

The output (a vector of size k) contains values that have been contributed by at most k nodes. If each of these (at most) k nodes behave correctly in at least one round of the algorithm, then the output of the algorithm is guaranteed to be correct. This is because a node which behaves correctly in a round contributes its share of the final result to the global vector in this round. The probability that a node behaves randomly in all the r rounds of the protocol is equal to $\prod_{j=1}^{r} P_0 * d^j = P_0^r * d^{r(r-1)/2}$, where d is the randomization factor. Thus the probability that a node behaves correctly in at least one round out of r rounds is equal to $(1 - P_0^r * d^{r(r-1)/2})$. The probability that the k nodes which contribute towards the final result behave correctly in at least one round is equal to $(1 - P_0^r * d^{r(r-1)/2})^k$.

For a fixed k, the above probability is monotonically increasing with an increasing value of r, i.e. it is more likely that the final result is correct if we execute the protocol for a larger number of rounds. For a fixed k, we also note that the probability of the output being correct increases with decreasing P_0 and decreasing d.

We further note that if we want the final output to be correct with a probability greater than $(1 - \epsilon)$, for some ϵ between 0 and 1, then the number of rounds that we have to run the protocol to achieve this accuracy is proportional to $\log 1/\epsilon$. Thus, we can achieve high accuracies for the first step of our protocol by running the $PP-TopK$ protocol for a small number of steps.

The second step, *privacy preserving classification*, is deterministic and provably accurate. This is because its accuracy depends on the accuracy of the privacy preserving addition protocol, which was proved to be accurate in [7]. Thus, the relative accuracy of our protocol is determined wholly by the accuracy of the first step, *privacy preserving nearest neighbor selection.*

The relative accuracy of the privacy preserving classification algorithm is best measured by extensive experiments. In these experiments, we train a privacy preserving classifier and a non-privacy preserving classifier on the same training sets; and then test the performance of each on the same test sets. We present a large number of experimental results in Section 6 to show that the accuracy of our algorithm is very close to the accuracy obtained by a distributed kNN classifier.

5.2 Efficiency

An important design goal for privacy preserving classification algorithms is their efficiency. We would ideally like the complexity of the privacy preserving classification algorithm to be of the same order as the complexity of a non-privacy preserving classification algorithm. There are two main considerations when we evaluate the efficiency of a privacy preserving classification algorithm - computational complexity and communication complexity. Ideally, we would want both these complexities to be of the same order as that of a non-privacy preserving, distributed classification algorithm. We would further like to have a choice between optimizing the algorithm for efficiency and optimizing the algorithm for privacy. Our privacy preserving kNN classification algorithm has both these important properties.

If the $PP-TopK$ algorithm is executed for s number of rounds, then the probability that the global vector contains the correct output is at least $(1 - P_0^s * r^{s(s-1)/2})^k$ (Section 4.1). If we want the final output to be correct with a probability greater than $(1 - \epsilon)$, for some ϵ between 0 and 1, then the number of rounds that we have to run the protocol to achieve this accuracy is asymptotic with respect to k, i.e. we can pick a value of s (the number of rounds) such that for any k, we achieve a correct final output with a probability greater than $(1 - \epsilon)$.

Thus the communication complexity of the $PP-kNN$ algorithm is $\Theta(n)$, where n is the number of nodes involved in the classification. This implies that the communication complexity of the privacy preserving nearest neighbor selection step is $\Theta(n)$.

The communication complexity of the *privacy preserving classification* step is $2 * n$, which is exactly the communication complexity of the privacy preserving addition protocol. The computational complexity of $PP-kNN$ is also of the same order as that of an ordinary, distributed kNN classifier, as there is no cryptography involved in any of the computations. Thus our $PP-kNN$ algorithm is easily able to accommodate a large number of nodes as well as large databases at every node, and scales well along these dimensions.

5.3 Privacy

Privacy breaches can be of two different types in our protocol - those that occur due to the kNN classification model and those that occur due to our implementation of the privacy preserving kNN algorithm. We discuss both of these below.

In our privacy preserving kNN classifier model, we divide kNN-classification into two steps - privacy preserving nearest neighbor selection and privacy preserving classification (section 3). Information about the data held by individual nodes can be compromised just by following this model, even the individual steps in the model are executed in a perfectly secure way. To see why this is true, consider the case when all the k nearest neighbors of a query are in the database of a single node n. In this case, after the execution of the privacy preserving nearest neighbor selection step, n can infer that all the k nearest neighbors of the query are within its database, in other words, there are no data points in any other node's database that are among the k nearest neighbors of the query.

The above breach of privacy indicates that care is required in the design of the kNN classification algorithm. We can overcome the probability of the above privacy breach by increasing k, the number of neighbors of the query node that we consider. To see why this is the case, consider the case when $k = \mid D_1 \cup D_2 \cup \ldots \cup D_n \mid$ - we are considering all the points in all the database's to classify the query instance. Thus the result of the nearest neighbor selection step will include all the points in all the databases. In fact, if we use the above classifier, in which all points in all the databases are considered when making a classification, then we can do away with the privacy preserving nearest neighbor selection step in our model. This is because every point in every node is by definition one of the k nearest neighbors of every query instance. This change increases the privacy preserving nature of the protocol, however, we pay a price in the accuracy of the kNN classifier constructed, as we demonstrate in our experiments, and also in efficiency when each database has a large number of points.

Privacy breaches may also occur due to the algorithms chosen in implementing the two steps of our model. The privacy preserving nearest neighbor selection step uses the PP-TopK algorithm. We note that in our model, we utilize only the distances of points from the query instance and not the actual points in a node's database. Thus, whatever information is leaked is only about the distance of a point from the query instance, and never the actual co-ordinates of an instance itself. This is an inherent strength of our model and guarantees that the actual coordinates of a point in a node's database is never revealed to other nodes.

The PP-TopK protocol also guarantees that with a high probability, the inputs of any node are not revealed to other nodes during the execution of the protocol. In our application, this means that with a high probability, other nodes do not learn about the distances of a node's k nearest points to the query point. This is a subtle but important requirement for privacy. To see why this is the case, consider a two dimensional space and a protocol in which every node learns about the distances of other node's k nearest points to a query instance. In two dimensional space, a point can be uniquely identified if we know its distance from two known points. Thus, a dishonest node might ask queries in such a way as to find out the distances of a point belonging to some other node from these query instances, and might thus be able to infer the co-ordinates of the point belonging to the other node. We show in our experiments that the probability that a node leaks information about its distance values during the $PP-TopK$ algorithm is very low.

The privacy preserving classification step uses the privacy preserving addition protocol and some local computation at the nodes. Thus its privacy is determined by the privacy of the addition protocol. In [7], it has been proved that this protocol is privacy preserving. Thus, the same holds true for our privacy preserving classification step.

6. EXPERIMENTAL EVALUATION

In this section, we evaluate the privacy preserving kNN classification algorithm in terms of its relative accuracy as compared to a distributed, non-privacy preserving kNN classification algorithm. We conduct experiments to demonstrate the sensitivity of our algorithm to its various parameters.

6.1 Experimental Setup

We use three publicly available datasets in our experiments. The first dataset, GLASS [9], contains data regarding various physical characteristics of different types of glass. The classification problem is to identify the type of glass from its physical characteristics. The study of classification of types of glass was motivated by criminological investigation - at the scene of a crime, the glass left behind can be used as evidence if it is correctly identified. This data set contains 214 instances belonging to 7 different classes, with each instance having 9 attributes. The second dataset, PIMA [20], is a medical dataset used for diagnostic purposes - for predicting whether a patient shows signs of diabetes given data like the 2-hour serum insulin concentration and Body Mass Index. This dataset contains 768 instances belonging to 8 different classes, with each instance having 8 different attributes. The third dataset, ABALONE [26], was used to predict the age of abalone from its physical characteristics. This data set contains 4177 instances belonging to 29 different classes, with each instance having 8 attributes.

In each experiment, we performed 100 separate runs on each different dataset. In each run, we randomly partitioned the data into two parts - a training set containing $\frac{3}{4}$ of the data and a test set containing $\frac{1}{4}$ of the data. We trained the classifier on the training set and tested it on the instances in the test set. The results reported are averaged over the 100 runs of each experiment.

We summarize the notation we use to describe the experimental results in Table 1.

6.2 Relative accuracy of privacy preserving kNN classification

Param.	Description
n	# of nodes in the system
k	parameter in kNN classification
P_0	initial randomization prob. used in $PP-TopK$ algorithm
d	dampening factor for randomization prob. used in $PP-TopK$ algorithm
r	Number of rounds for which $PP-TopK$ algorithm is executed.

Table 1: Experiment Parameters

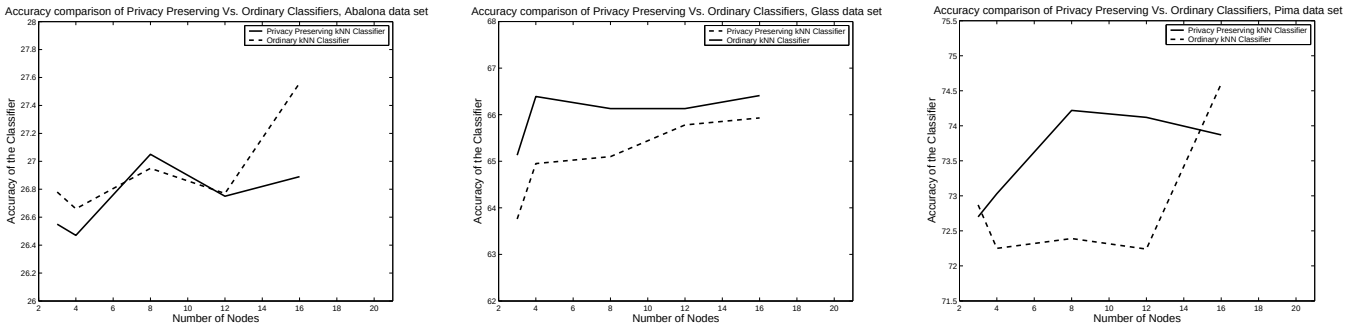


Figure 2: Accuracy comparison of privacy preserving classifier Vs. distributed, ordinary kNN classifier

In this experiment, we compare the accuracy of privacy preserving kNN classification against a distributed kNN classifier. In these experiments, we run the nearest neighbor selection step for one round, with the initial probability $P_0 = 1$ and the randomization factor $d = 0.5$. Thus, with these parameter values, our algorithm is very efficient - its communication complexity is only 3 times that of an ordinary, distributed kNN classifier. We would like to determine the accuracy of this classifier which is optimized for efficiency..

We notice from Figure 2 that although we have chosen very conservative settings for our privacy preserving classifier - we run our nearest neighbor selection algorithm for only one round, the accuracy obtained by our classifier is still very high as compared to an ordinary classifier. Note that in some of the cases, the accuracy of the privacy preserving classifier is actually higher than that of a distributed classifier. This is not a contradiction, because an incorrect result of the nearest neighbor selection step could actually produce a value of "k" for which the kNN algorithm performs better.

Thus, this experiment indicates that our privacy preserving algorithm matches the accuracy of a distributed, non-privacy preserving kNN classifier without sacrificing its efficiency. In the next few experiments, we try to reduce the difference in accuracy of the two classifiers, at the cost of increasing the communication complexity of the privacy preserving classifier.

6.3 Effect of number of rounds on Relative Accuracy

Of the two steps in our privacy preserving kNN classifier model, the first step, nearest neighbor selection, is probabilistic in nature while the second step, classification, is deterministic and provably accurate. Thus, the overall accuracy of our classifier is determined by the accuracy of the nearest neighbor selection step. The accuracy of the nearest neighbor selection step is determined by the accuracy of the $PP-TopK$ algorithm. We recall that the accuracy of this algorithm is a function of three parameters - P_0, the initialization probability, d, the randomization factor and r, the number of rounds the protocol is executed. From our discussion of this algorithm, we know that the accuracy of the $PP-TopK$ protocol increases when we increase r, or decrease either P_0 or d. In this experiment, we verify whether the accuracy of the privacy preserving kNN classifier approaches that of the distributed kNN classifier when we run the $PP-TopK$ protocol for a larger number of rounds. To do this, we measure the absolute value of the difference between the accuracies of the two classifiers when trained and tested on the same data. The results of our experiments are presented in Figure 3.

We note from Figure 3 that we are able to make the privacy preserving classifier as accurate as an ordinary classifier by running the $PP-TopK$ algorithm for a larger number of rounds. This is because the accuracy of the $PP-TopK$ protocol increases if we run it for a larger number of rounds; thus, with a high probability,the value of "k" determined by the nearest neighbor selection step will be the same as the "k" used by the ordinary algorithm.

6.4 Effect of randomization factor (d) on Relative Accuracy

In this experiment, we investigate the effect of varying the amount of randomization inserted into the computation in each round on the accuracy of the classifier. We do this by varying the value of d, the randomization factor. We note that if d is large, this means that more randomization is inserted into the computations, thus increasing the privacy guarantees of the algorithm. A very low value of d makes the computations almost deterministic and thus provides almost no privacy guarantees.

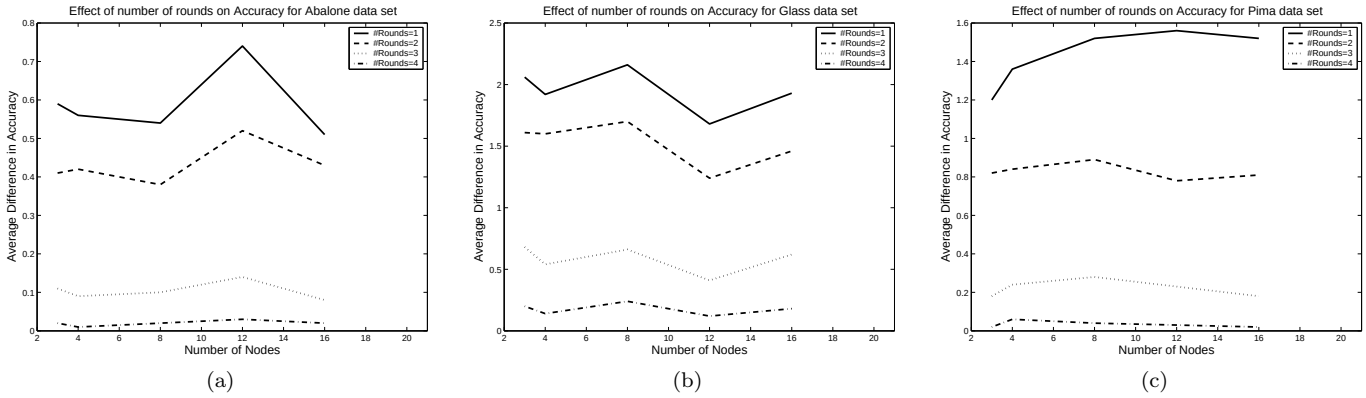


Figure 3: Relative Accuracy of the privacy preserving kNN classifier with varying number of rounds in nearest neighbor selection

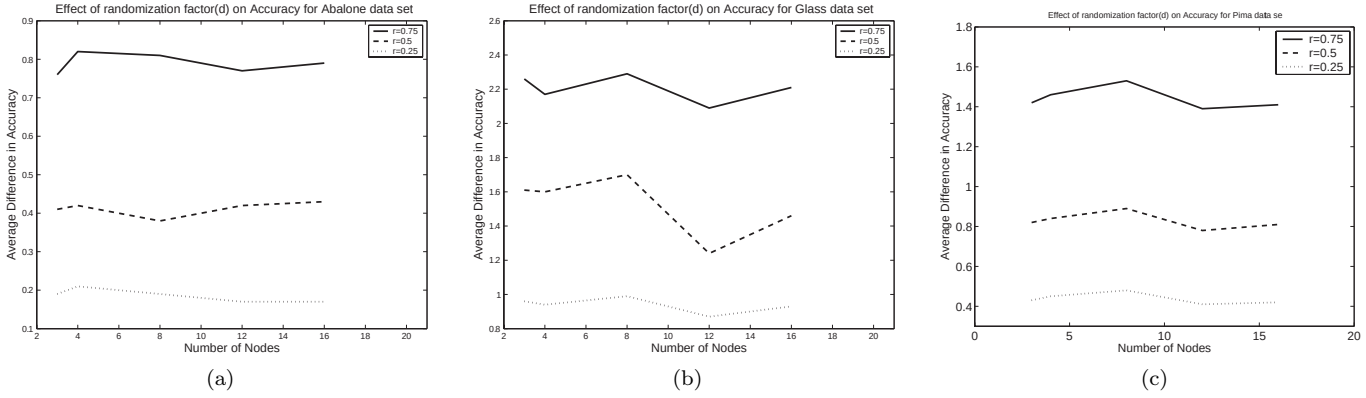


Figure 4: Relative Accuracy of the privacy preserving kNN classifier with varying randomization parameter in nearest neighbor selection

The results of our experiments are shown in Figure 4. We note that with decreasing values of d, the accuracy of the privacy preserving classifier becomes closer to that of the ordinary classifier. This is because smaller values of d increase the probability that the output of the nearest neighbor selection step is correct, and thus increase the probability of the two classifiers performing identical classification.

6.5 Effect of "k" on accuracy

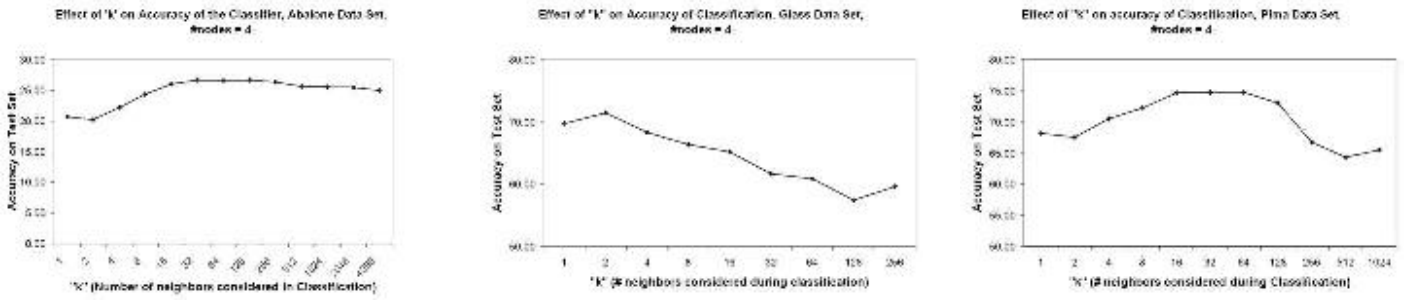
In this experiment, we investigate the effect of "k", the number of neighbors considered during classification, on the accuracy of both privacy preserving and ordinary kNN classifiers. As discussed in section 5.3, setting k equal to all the points in the union of the databases results in a kNN classification algorithm which has very good privacy guarantees. However, our experiments show that this decreases the accuracy of the classifier, and we also note that the efficiency of the algorithm is reduced by this setting. So, we would like k to be large to avoid any information leak due to our model, as discussed in section 5.3. By studying the graphs of accuracy vs. "k", we show that it is indeed possible to set k to a large value without compromising much on the accuracy of the classifier.

We report the average accuracy when we used different values of "k" for classification in Figure 5. We note that the accuracy of the classifier decreases when "k" is very large. However, from the graphs, we note that it is possible to pick a k to achieve both good accuracy and good privacy.

6.6 Privacy of the $PP-TopK$ Protocol

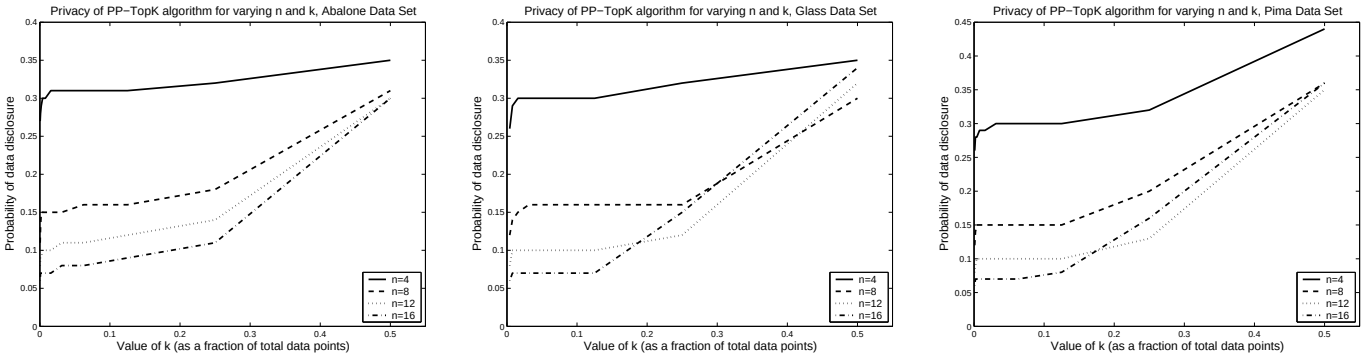
In this section, we measure the amount of information revealed by a node to its successor during the execution of the $PP{-}TopK$ algorithm. In this experiment, we run the protocol for $r = 2$ rounds, with $P_0 = 1$ and $d = 0.5$. We measure the probability that a node is able to correctly identify a value in the global vector it received from its predecessor as belonging to its predecessor. We present these results in Figure 6. From the figure, we note that for all values of n, there is a large range of k such that the probability of a node revealing information to its successor is less than half. With very large values of k however, a node has a higher probability of inserting its values in the global vector and this increases its chances of revealing its values to its successor.

Our experiments indicate that even if we run the algorithm for a larger number of rounds, and use a range of values for the randomization factor d, the probability that a node reveals its values to its successor is still very low.



(a) (b) (c)

Figure 5: Effect of "k" on Accuracy



(a) (b) (c)

Figure 6: Privacy of the $PP-TopK$ algorithm Vs. n and k

However, space restrictions prevent us from presenting the graphs for these results.

7. RELATED WORK

Privacy related problems in databases have been an active and important research area. Research in secure databases, Hippocratic databases and privacy policy driven systems [14, 4, 2] has been focused on enabling access of sensitive information through centralized role-based access control. More recently research has been done in areas such as privacy preserving data mining, privacy preserving query processing on outsourced databases, and privacy preserving information integration.

In database outsourcing scenarios, data partitioning and binning techniques based on generalization principle have been proposed to minimize information leakage [12, 13, 5]. In privacy preserving data mining [25, 8], the main approach is to use data perturbation techniques to hide precise information in individual data records. These techniques may not apply to various data sharing tasks where precise results are desired. Our work utilizes a different randomization approach that randomizes data through multiple rounds while at the end guaranteeing the correct output with a probabilistic bound.

The approach of protecting privacy of distributed sources was first addressed by the construction of decision trees [18]. This work closely followed the traditional secure multiparty computation approach and achieved perfect privacy. A key insight of this paper was to trade off computation and communication costs for accuracy, thereby improving efficiency over the generic secure multiparty methods. There has since been work to address association rules in horizontally partitioned data [15], association rules in vertically partitioned data [21], naive Bayes classification in horizontally partitioned data [17] and vertically partitioned data [23], k-means clustering over vertically partitioned data [22]. As a recent effort, there is also research on privacy preserving top$k$ queries across vertically partitioned data using k-anonymity privacy model [24] and privacy preserving distributed k-NN classifier [16] across vertically partitioned data. On the contrary, our protocol computes top$k$ selection across horizontally partitioned data and uses a different data privacy model. The top$k$ selection can be also served as a primitive function for more complex aggregate queries or data integration tasks such as kNN classifier across horizontally distributed data.

Agrawal et al. [3] also introduced the paradigm of minimal information sharing in information integration domain and proposed a privacy preserving join protocol between two parties. A few specialized protocols have been proposed, typically in a two party setting, e.g., for finding intersections [3], and kth ranked element [1]. Though still based on cryptographic primitives, they achieve better efficiency than traditional multi-party secure computation methods by allowing minimal information disclosure. In contrast, our protocol does not require any cryptographic operations. It

leverages the multi-party network and utilizes a probabilistic scheme to achieve minimal information disclosure and minimal overhead.

Finally, distributed consensus protocols such as leader election algorithms [19] provide system models for designing distributed algorithms. However they are not concerned about data privacy constraints of individual nodes.

8. CONCLUSION

In this paper, we have tackled the problem of privacy preserving data mining using the k-Nearest Neighbor classifier. This is the first paper to show how kNN classification can be achieved in a privacy preserving manner using a decentralized network protocol. We developed a general model for kNN classification and presented algorithms for realizing this model. We have laid out the set of requirements that all privacy preserving classifiers should strive to achieve and have analyzed how well our algorithm achieves these requirements. Our algorithm is characterized by its ability to achieve a balance between three important performance metrics: relative accuracy, efficiency, and privacy. This enables our privacy preserving kNN classification framework to be applied to a variety of problem settings and meet different optimization criteria.

9. REFERENCES

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth ranked element. In *IACR Conference on Eurocrypt*, 2004.

[2] R. Agrawal, P. Bird, T. Grandison, J. Kieman, S. Logan, and W. Rjaibi. Extending relational database systems to automatically enforce privacy policies. In *21st IEEE International Conference on Data Engineering (ICDE)*, 2005.

[3] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *ACM SIGMOD Conference*, 2003.

[4] R. Agrawal, J. Kieman, R. Srikant, and Y. Xu. Hippocratic databases. In *International Conference on Very Large Databases (VLDB)*, 2002.

[5] E. Bertino, B. Ooi, Y. Yang, and R. H. Deng. Privacy and ownership preserving of outsourced medical data. In *21st IEEE International Conference on Data Engineering (ICDE)*, 2005.

[6] C. Clifton. Tutorial on privacy, security, and data mining. In *13th European Conference on Machine Learning and 6th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2002.

[7] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining. In *SIGKDD Explorations*, 2003.

[8] J. Gehrke. Models and methods for privacy-preserving data analysis and publishing. In *ICDE*, 2006.

[9] B. German. In *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/glass*.

[10] O. Goldreich. Secure multi-party computation, 2001. Working Draft, Version 1.3.

[11] S. Goldwasser. Multi-party computations: past and present. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 1997.

[12] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database service provider model. In *ACM SIGMOD Conference*, 2002.

[13] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 1997.

[14] S. Jajodia and R. Sandhu. Toward a multilevel secure relational data model. In *ACM SIGMOD Conference*, 1991.

[15] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9), 2004.

[16] M. Kantarcioglu and C. Clifton. Privacy preserving k-nn classifier. In *IEEE International Conference on Data Engineering (ICDE)*, 2005.

[17] M. Kantarcoglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally partitioned data. In *IEEE ICDM Workshop on Privacy Preserving Data Mining*, 2003.

[18] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3), 2002.

[19] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[20] V. Sigillito. Pima. In *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pima-indians-diabetes*.

[21] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[22] J. vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *SIGKDD '03*, 2003.

[23] J. Vaidya and C. Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[24] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *IEEE International Conference on Data Engineering (ICDE)*, 2005.

[25] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 33(1), 2004.

[26] S. Waugh. In *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/abalone*.

[27] L. Xiong, S. Chitti, and L. Liu. Topk queries across multiple private databases. In *25th International Conference on Distributed Computing Systems (ICDCS)*, 2005.