Approximate Privacy-Preserving Data Mining on Vertically Partitioned Data

Robert Nix[1], Murat Kantarcioglu[1], and Keesook J. Han[2]

[1] Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
800 West Campbell Road
Richardson, Texas, USA
{rcn062000,muratk}@utdallas.edu
[2] Air Force Research Laboratory
Information Directorate
525 Brooks Road
Rome, New York, USA
Keesook.Han@rl.af.mil

Abstract. In today's ever-increasingly digital world, the concept of data privacy has become more and more important. Researchers have developed many privacy-preserving technologies, particularly in the area of data mining and data sharing. These technologies can compute exact data mining models from private data without revealing private data, but are generally slow. We therefore present a framework for implementing efficient privacy-preserving secure approximations of data mining tasks. In particular, we implement two sketching protocols for the scalar (dot) product of two vectors which can be used as sub-protocols in larger data mining tasks. These protocols can lead to approximations which have high accuracy, low data leakage, and one to two orders of magnitude improvement in efficiency. We show these accuracy and efficiency results through extensive experimentation. We also analyze the security properties of these approximations under a security definition which, in contrast to previous definitions, allows for very efficient approximation protocols.[3]

1 Introduction

Privacy is a growing concern among the world's populace. As social networking and cloud computing become more prevalent in today's world, questions arise about the safety and confidentiality of the data that people provide to such services. In some domains, such as medicine, laws such as HIPAA and the Privacy Act of 1979 step in to make certain that sensitive data remains private. This is great for ordinary consumers, but can cause problems for the holders of the data. These data holders would like to create meaningful information from the data that they have, but privacy laws prevent them from disclosing the data

[3] Approved for Public Release; Distribution Unlimited: 88ABW-2011-4946, 16-Sep 2011

to others. In order to allow such collaboration between the holders of sensitive data, *privacy-preserving* data mining techniques have been developed.

In privacy-preserving data mining, useful models can be created from sensitive data without revealing the data itself. One way to do this is to perturb the data set using anonymization or noise addition [7] and perform the computation on that data. This approach was first pioneered by Agrawal and Srikant [3]. These methods can suffer from low utility, since the data involved in the computation is not the actual data being modeled. In addition, these protocols can suffer from some security problems[18, 13, 21], which can lead to the retrieval of private data from the perturbed data given.

The other way to do this is using secure multiparty computation techniques to compute the exact data mining result, on the actual data. Secure computation makes use of encryption schemes to keep the data secret, but relies on other tactics, such as encrypting the function itself, or homomorphic properties of the encryption, to perform the computation. This approach was first used by Lindell and Pinkas [20]. These schemes generally rely on very slow public key encryption, which results in a massive decrease in information output. The exact computation of data mining models can take thousands of times longer when using these public key cryptosystems.

While many functions are very difficult to compute using secure multiparty computation, some of these functions have approximations which are much easier to compute. This is especially true in those data mining tasks that deal with aggregates of the data, since these aggregates can often be easily estimated. Approximating the data mining result, however, can lead to some data leakage if the approximation is not done very carefully. The security of approximations has been analyzed by Feigenbaum, et al., [8], but the results of their analysis showed that to make an approximation fully private, the process of the computation must be substantially more complex. Sometimes, this complexity can make computing the approximation more difficult than computing the function itself!

Here, we present another security analysis that, while allowing some small, parameter defined data leakage, creates the opportunity to use much simpler and less computationally expensive approximations securely. We then use this model of security to show the security of two approximation methods for a subprotocol of many vertically partitioned data mining tasks: the two-party dot product. The dot product is used in association rule mining, classification, and other types of data mining. We prove that our approximations are secure under our reasonable security definitions. These approximations can provide one to two orders of magnitude improvement in terms of efficiency, while sacrificing very little in accuracy.

1.1 Summary of Contributions

A summary of our contributions are as follows:

– We outline a practical security model for secure approximation which allows simple protocols to be implemented securely.

– We showcase two sketching protocols for the dot product and prove their security under our model.
– Through experimentation, we show the practicality of these protocols in vertically partitioned privacy-preserving data mining tasks. These protocols can lead to a two order of magnitude improvement in efficiency, while sacrificing very little in terms of accuracy.

In section 2, we summarize the current state of work in this area. Section 3 provides the standard definitions of secure approximations, and our minor alteration thereof. Section 4 outlines the approximation protocols we use. Section 5 gives the proof that these simple approximation protocols are secure under our definition of secure approximation. In section 6, we give experimental results for different data mining tasks using the approximations. Finally, we offer our overall conclusions and future directions in section 7.

2 Related Work

Privacy-preserving data mining (PPDM) is a vast field with hundreds of publications in many different areas. The two landmark papers by Agrawal and Srikant [3] and Lindell and Pinkas [20] began the charge, and soon many privacy preserving techniques emerged for computing many data mining models [16, 27, 5, 24]. Other techniques can be found in the survey [2]. For our purposes, we will focus on those works which are quite closely related to the work in our paper.

There are quite a few protocols previously proposed for the secure computation of the dot product. The protocol proposed by [27] is quadratic in the size of the vector (times a security parameter). It does, however, have some privacy concerns accoring to [11]. This same work, along with several others [6, 14] propose other protocols which are based on very slow public key cryptography. [26] proposes a sampling-based algorithm for secure dot product computation which relies on secure set intersection as a sub-protocol. However, the secure set intersection problem is also nontrivial. It either relies on a secure dot product protocol [27] (which would lead to a circular dependency with [26]), or a large amount of extremely expensive cryptographic operations [30].

The sketching primitives used in this work have been applied to data mining in several different capacities. [25] uses Bloom filters to do association rule mining. However, the model employed in this framework requires a server hierarchy, in which the association rule mining is done at the top level, and represents transactions, not itemsets, as Bloom filters. The Johnson-Lindenstrauss theorem is employed for data mining by [22], however, they employ the Johnson-Lindenstrauss theorem as the sole means of preserving privacy, whereas we are using it as part of a process. Other works [9, 31] use Johnson-Lindenstrauss projection as an approximation tool. These, however, do not make use of the projection in a privacy-preserving context, and are merely concerned with fast approximations.

The work of [17] presents a sketching protocol for the scalar product based on Bloom filters. However, its experimentation and discussion of actual data

mining tasks was insufficient. Our protocols perform better on real data mining tasks, especially at high compression ratios.

3 Secure Approximations

Much has been written about secure computation, and the steps one must go through in order to compute functions without revealing anything about the data involved. Securely computing the *approximation* of a function poses another challenge. In addition to not revealing the data through the computation process, we must also assure that the function we use to approximate the actual function must not reveal anything about the data! To this end, we outline a definition of secure approximations given by [8], and then propose an alteration to this framework. This alteration, while allowing a very small amount of data leakage, allows for the use of very efficient approximation protocols, which can improve the efficiency of exact secure computation by orders of magnitude.

3.1 A secure approximation framework

The work of Feigenbaum, et. al. [8] gives a well-constructed and thorough definition of secure approximations. In the paper, they first define a concept called *functional privacy*, then use this definition to define the notion of a secure approximation. First, we examine the definition of functional privacy, as follows:

Definition 1 *Functional Privacy*: Let $f(\mathbf{x})$ be a deterministic, real valued function. Let $\hat{f}(\mathbf{x})$ be a (possibly randomized) function. $\hat{f}$ is *functionally private* with respect to f if there exists a probabilistic, expected polynomial time sampling algorithm $\mathbf{S}$ such that for every input $\mathbf{x} \in X$, the distribution $\mathbf{S}(f(\mathbf{x}))$ is indistinguishable from $\hat{f}(\mathbf{x})$.

Note that the term "indistinguishable" in the definition is left intentionally vague. This could be one of the standard models of perfect indistinguishability, statistical indistinguishability, computational indistinguishability [23], or any other kind of indistinguisability. In these cases, the adjective applied to the indistinguishablity is also applied to the functional privacy (i.e., statistical functional privacy for statistical indistinguishability).

Intuitively, this definition means that the result of $\hat{f}$ yields no more information about the input than the actual result of f would. Note, however, that this does not claim that there is any relation between the two outputs, other than the privacy requirement. This does not require that the function $\hat{f}$ be a good approximation of f. Feigenbaum, et al., therefore, also provide a definition for approximations, which is also used in the final concept of a secure approximation.

Definition 2 *P-approximation*: Let $P(f, \hat{f})$ be a predicate for determining the "closeness" of two functions. A function $\hat{f}$ is a *P-approximation* of f if $P(f, \hat{f})$ is satisfied.

Now, for this definition to be useful, we need to define a predicate P to use for the closeness calculation. The most commonly used predicate P is the $\langle \epsilon, \delta \rangle$ criterion, in which $\langle \epsilon, \delta \rangle (f, \hat{f})$ is satisfied if and only if $\forall \mathbf{x}, Pr[(1-\epsilon)f(\mathbf{x}) \leq$

$\hat{f}(\mathbf{x}) \leq (1+\epsilon)f(\mathbf{x})] > 1-\delta$. We do not refer to any other criterion in our work, but the definition is provided with a generic closeness predicate for the sake of completeness.

Finally, we present the liberal definition of secure two party approximations as outlined in Feigenbaum, et al.

Definition 3 *Secure Approximation (2-parties)*: Let $f(\mathbf{x}_1, \mathbf{x}_2)$ be a deterministic function mapping the two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ to a single output. A protocol p is a secure P-approximation protocol for f if there exists a functionally private P-approx-imation $\hat{f}$ such that the following conditions hold:

Correctness The outputs of the protocol p for each player are in fact equal to the same $\hat{f}(\mathbf{x}_1, \mathbf{x}_2)$.

Privacy There exist probabilistic polynomial-time algorithms $\mathbf{S}_1, \mathbf{S}_2$ such that

$$\{(\mathbf{S}_1(\mathbf{x}_1, f(\mathbf{x}_1, \mathbf{x}_2), \hat{f}(\mathbf{x}_1, \mathbf{x}_2)), \hat{f}(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1,\mathbf{x}_2)\in X} \stackrel{c}{\equiv}$$
$$\{(\text{view}_1^p(\mathbf{x}_1, \mathbf{x}_2), \text{output}_2^p(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1,\mathbf{x}_2)\in X},$$
$$\{(\hat{f}(\mathbf{x}_1, \mathbf{x}_2), \mathbf{S}_2(\mathbf{x}_1, f(\mathbf{x}_1, \mathbf{x}_2), \hat{f}(\mathbf{x}_1, \mathbf{x}_2)))\}_{(\mathbf{x}_1,\mathbf{x}_2)\in X} \stackrel{c}{\equiv}$$
$$\{(\text{output}_1^p(\mathbf{x}_1, \mathbf{x}_2), \text{view}_2^p(\mathbf{x}_1, \mathbf{x}_2))\}_{(\mathbf{x}_1,\mathbf{x}_2)\in X}$$

where $A \stackrel{c}{\equiv} B$ means that A is computationally equivalent to B. Note that in the above definition all instances of $\hat{f}(\mathbf{x}_1, \mathbf{x}_2)$ have the same value, as opposed to being some random value from the distribution of $\hat{f}$. This limits the application of the simulators to a single output. This definition essentially says that we have a functionally private function $\hat{f}$ which is a P-approximation of f which itself is computed in a private manner, such that no player learns anything else about the input data.

3.2 Our definition

Having defined the essential notions of functional privacy, approximations, and secure approximations, we now define another notion of functional privacy, which, while less secure than the above model, allows for vastly more efficient approximations.

Definition 4 $\langle\epsilon, \delta\rangle$*-functional privacy*: A function $\hat{f}$ is $\langle\epsilon, \delta\rangle$*-functionally private* with respect to f if there exists a polynomial time simulator $\mathbf{S}$ such that $Pr[|\mathbf{S}(f(\mathbf{x}), R) - \hat{f}(\mathbf{x})| < \epsilon] > 1-\delta$, where R is a shared source of randomness involved in the calculation of $\hat{f}$.

Intuitively, this definition allows for a non-negligible but still small acceptable information loss of at most ϵ, while still otherwise retaining security. In practice, the amount of information revealed could be much smaller, but this puts a maximum bound on the privacy of the function. In addition, we allow the simulator access to the randomness function used in computing $\hat{f}$, which allows the simulator to more accurately produce similar results to $\hat{f}$.

The acceptable level of loss ϵ can vary greatly with the task at hand. For example, if the function is to be run on the same data set several times, the

leakage from that data set would increase with each computation. Thus, for applications with higher repetition, we would want a much smaller ϵ. The ϵ can be adjusted by using a more accurate approximation.

In their work describing the original definition above, Feigenbaum, et al. [8] dismissed a simple, efficient approximation protocol based on their definition of functional privacy. This approximation was a simple random sampling based method for approximating the hamming distance between two vectors. The claim was that even if the computation was done entirely securely, some information about the randomness used in the computation would be leaked into the final result. Thus, we simply explicitly allow the randomness to be used by the simulator in our model. We feel this is realistic, as the randomness is common knowledge to all parties in the computation.

In short, the previous definition of [8] aims to eliminate data leakage from the approximation result. Our definition simply seeks to quantify it and reduce it to acceptable levels. In return, we can use much simpler approximation protocols securely. For example, the eventual secure hamming distance protocol given by [8] has two separate protocols (one which works for high distance and one for low distance) each of which requires several rounds of oblivious transfers between the two parties. Under our definition, protocols can be used which use only a single round of computation and work for any type of vector, as we will show in the next section.

4 Scalar Product Approximation Techniques for Distributed Data Mining

Data mining is, in essence, the creation of useful models from large amounts of raw data. This is typically done through the application of machine learning based model building algorithms such as association rules mining, naive bayes classification, linear regression, or other model creation algorithms. Distributed data mining, then, is the creation of these models from data which is distributed (partitioned) across multiple owners. The dot product of two vectors has many applications in vertically partitioned data mining. Many data mining algorithms can be reduced to one or more dot products between two vectors in the vertically partitioned case. Vertical partitioning can be defined as follows:

Let X be a data set containing tuples of the form $(a_1, a_2, ..., a_k)$ where each a is an attribute of the tuple. Let S be a subset of $\{1, 2, ..., k\}$. Let X_S be the data set where the tuples contain only those attributes specified by the set S. For example, $X_{\{1,2\}}$ would contain tuples of the form (a_1, a_2). The data set X is said to be vertically partitioned across n parties if each party i has a set S_i, and the associated data X_{S_i}, and

$$\bigcup_{i=1}^{n} S_i = \{1, 2, ..., k\}$$

In previous work, it has been shown that the three algorithms we test in this paper can in fact be reduced to the dot product of two zero-one vectors in

the vertically partioned case. These algorithms are association rules mining[17], naive Bayes classification[28], and C4.5 decision tree classification[29].

We developed two sketching protocols for the approximation of the dot product of two zero-one vectors. These protocols are used to provide smaller input to an exact dot product protocol, which is then used to estimate the overall dot product, as outlined in figure 1. First, we present a protocol based on the Johnson-Lindenstrauss theorem [15] and the work of [1] and [19]. Then, we present a simple sampling algorithm which is also secure under our model. Finally, we present a proof of the security of these approximations in our security model.

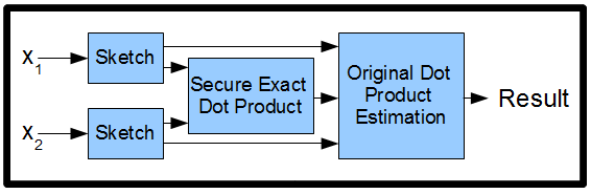


Fig. 1. Dot Product Approximation Concept

4.1 Johnson-Lindenstrauss (JL) Sketching

The Johnson-Lindenstrauss theorem [15] states that for any set of vectors, there is a random projection of these vectors which preserves Euclidean distance within a tolerance of ϵ. More formally, for a given ϵ, there exists a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that for all u and v in a set of points,

$$(1-\epsilon)||u-v||^2 \leq ||f(u)-f(v)||^2 \leq (1+\epsilon)||u-v||^2$$

It is shown in that because of this property, the dot product is also preserved within a tolerance of ϵ. As with any sketching scheme, the probability of being close to the correct answer increases with the size of the sketch.

As outlined in [1] and [19], to do our random projection, we generate a $k \times n$ matrix R, where n is the number of rows in the data set, and k is the number of rows in the resultant sketch. Each value of this matrix has the value 1, 0, or -1, with probabilities set by a sparisity factor s. The value 0 has a probability of $1-\frac{1}{s}$, and the values 1 and -1 each have a probability of $\frac{1}{2s}$. In order to sketch a vector a of length n, we do $\frac{\sqrt{s}}{\sqrt{k}}Ra$, which will have a length of k. This preserves the dot product to within a certain tolerance. So, to estimate the dot product of two vectors a and b, we merely compute $\frac{\sqrt{s}}{\sqrt{k}}Ra \cdot \frac{\sqrt{s}}{\sqrt{k}}Rb$. Note that this will be equal to $s\frac{Ra \cdot Rb}{k}$, and in practice, we typically omit the $\frac{\sqrt{s}}{\sqrt{k}}$ term from the sketching protocol, and simply divide by the length of the sketch and multiply

by the sparsity factor after performing the dot product. This yields the same result. This is shown below as Algorithm 4.1.

According to [19], the sparsity factor s can be as high as $\frac{n}{logn}$ before significant error is introduced, and as s increases, the time and space requirements for the sketch decrease. Nevertheless we still used relatively low sparsity factors, to show that even in the slowest case, we still have an improvement.

Algorithm 4.1 Johnson-Lindenstrauss(JL) Dot Product Protocol

RandomMatrixGeneration(n,k):
Matrix R
for $i \leftarrow 1...n$ **do**
  for $j \leftarrow 1...k$ **do**
    $R_{j,i} \xleftarrow{\$} \{\frac{1}{2s} : -1, 1 - \frac{1}{s} : 0, \frac{1}{2s} : 1\}$
  end for
end for
return R

DotProductApproximation(Vector u,Vector v, k):
Matrix $R \leftarrow$ RandomMatrixGeneration($|u|, k$)
$u' \leftarrow Ru$
$v' \leftarrow Rv$
return $\frac{s \cdot \text{SecureDotProduct}(u', v')}{k}$

4.2 Random Sampling

In addition to the more complicated method above, to estimate the dot product of two vectors, one could simply select a random sample of both vectors, compute the dot product, then multiply by a scaling factor to estimate the total dot product. Note that this works fairly well on vectors where the distribution of values is known, such as zero-one vectors, but can work quite poorly on arbitrary vectors. The sampling algorithm is shown below in Algorithm 4.2.

5 Approximation Protocol Security

We now provide a proof that each of the above protocols provides a secure approximation in the sense outlined above. We first show the $\langle 2\epsilon, \delta^2 \rangle$-functional privacy of the protocols, then show that the protocols are secure under the liberal definition of secure approximations.

Theorem The protocols outlined in section 4 are both $\langle 2\epsilon, \delta^2 \rangle$-functionally private, and meet the liberal definition for secure approximations (definition 3).

Proof:

Functional Privacy Let ϵ, δ be the approximation guarantees granted by the above protocols. That is, $Pr[|u \cdot v - \text{DotProductApproximation}(u, v)| > \epsilon] <$

Algorithm 4.2 Sampling Protocol

```
Sketch(Vector v, samplingFactor ∈ [0...1]):
sketch ← []
for i ← 1...n do
  r ←$ [0...1]
  if r < samplingFactor then
    sketch.append(v_i)
  end if
end for
return sketch

DotProductApproximation(u,v,samplingFactor)
u' ← Sketch(u)
v' ← Sketch(v)
return SecureDotProduct(u',v')·|u| / |u'|
```

$1-\delta$. For JL, these bounds are provided by the Johnson-Lindenstrauss theorem itself, as shown by the work of [22]. For sampling, we can use the Hoeffding inequality [12] to establish a bound on the error:

$$Pr[|\hat{f}(\mathbf{x}) - f(\mathbf{x})| \geq \epsilon] \leq 2e^{-2\epsilon^2 n^2}$$

Where n is the sample size. As $\hat{f}$ can be taken to be an estimate of the mean of the product of the random variables, the Hoeffding inequality holds for the dot product of the samples. So, we set our δ to $2e^{-2\epsilon^2 n^2}$.

Note that, with both of these approximation protocols, adjusting the size (for JL, the matrix size, and for sampling, the sample size), allows us to adjust the ϵ of the functional privacy requirement. This would allow us to adjust the ϵ value to be as low as we deemed necessary for our purposes.

Now, let our simulator $\mathbf{S}(f(\mathbf{x}), R)$ generate two random zero-one vectors u and v such that $f(u, v) = u \cdot v = f(\mathbf{x})$. We then apply the randomness given to perform a calculation of the dot product approximation $\beta = \hat{f}(u, v)$. Now, the probability that $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \geq \epsilon$ is $1-\delta$. The probability that $|f(\mathbf{x}) - \beta| \geq \epsilon$ is also $1-\delta$, since $f(\mathbf{x}) = f(u, v)$. As these are independent events, the probability that neither occurs is δ^2. In the case this occurs, we have $|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq \epsilon$ and $|f(\mathbf{x}) - \beta| \leq \epsilon$, which means that $-\epsilon \leq f(\mathbf{x}) - \hat{f}(\mathbf{x}) \leq \epsilon$ and $-\epsilon \leq f(\mathbf{x}) - \beta \leq \epsilon$. Because of this, the difference between the two quantities $(f(\mathbf{x}) - \hat{f}(\mathbf{x})) - (f(\mathbf{x}) - \beta) = \beta - \hat{f}(\mathbf{x})$ can be no more than 2ϵ. If our simulator returns β, then we have shown that $\hat{f}$ is $\langle 2\epsilon, \delta^2 \rangle$-functionally private with respect to f.

Secure Approximation (under Definition 3) For the approximation to be considered secure, it must compute the same value for both players (which is trivially true for both protocols), and be private with respect to the views of each player. Now, consider, in each case, what each player sees. Player 1 sees his input, a sketch of that input, and the inputs and outputs of a secure dot product protocol. Our simulator can take that input, sketch it, and simulate the

secure dot product protocol, altering its output to be $\hat{f}(\mathbf{x})$ to player 1. Since this output is all player 1 sees outside of the secure dot product protocol, it cannot distinguish this from the true output. Player 2 sees the same thing, his input, a sketch of that input, and the operations of a secure dot product protocol on the inputs. Since the subprotocol is secure, neither player can learn anything about the inputs that the sketches would not tell them.

Having shown that the sketching protocols are $\langle \epsilon, \delta \rangle$-functionally private, and that the computation protocol is secure under definition 3, we now claim that the entire protocols are secure under our model. □

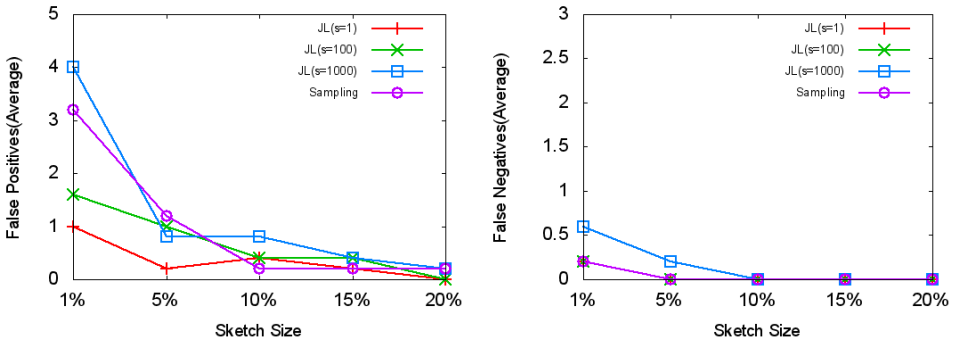
6 Experiments

In order to determine the efficiency and effectiveness of the algorithms proposed, we conducted several experiments. Each of the sketching protocols presented were inserted into the data mining process for three different data mining tasks: association rules mining, naive Bayes classification, and C4.5 decision tree classification. We used three separate sparsity values for JL sketching: $s = 1$, which results in a matrix completely full of 1 and -1, $s = 100$, and $s = 1000$. The efficiency of JL increases with s, and these values are much lower than what is required to achieve good accuracy [19].

For association rules mining, we used the retail data set found at [10], which lists transactions from an anonymous Belgian retail store. We considered three variables in the association rules experiments: the required support, the required confidence, and the compaction ratio of the sketching protocol. For testing the required support, we used 2%, 3%, 4%, 5%, and 6%, while holding the confidence constant at 70% and the compaction ratio constant at 10%. For the confidence, we used 60%, 65%, 70%, 75%, and 80% while holding the support constant at 4% and the compaction ratio constant at 10%. Finally, for the compaction ratio, we used 1%, 5%, 10%, 15%, and 20%, holding the support constant at 4% and the confidence constant at 70%. For naive Bayes and C4.5 decision tree classification, we used the Adult data set from the UC Irvine Machine Learning Repository [4], which consists of data from the 1993 US Census. As there were no paramaters to set for naive Bayes or the decision tree, we varied only the compaction ratio as above. We did, however, discretize each attribute of the data set before performing the data mining, as continuous data would not function under our model. For each task and variable set, we ran ten separate experiments, using different initialization values for the inherent randomness in the sketching protocols. We employed ten-fold cross-validation for the classification tasks. The accuracy results were then averaged over all ten trials to come up with the final result.

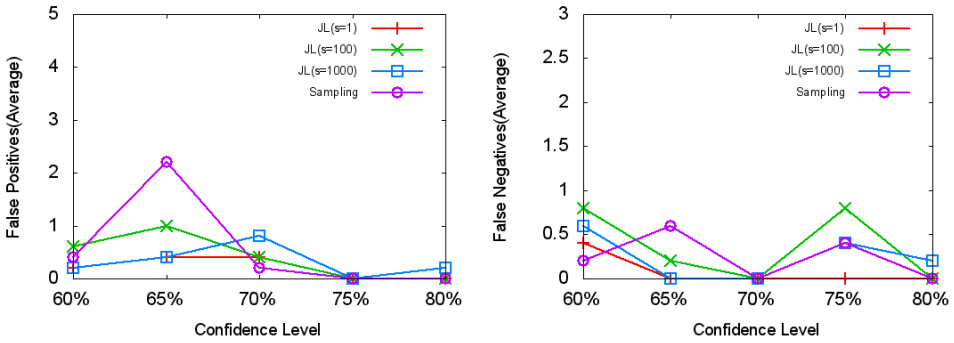
6.1 Accuracy

Association Rules Mining To assess the accuracy of the algorithms on association rules mining, we look at both the number of false positives (that is,



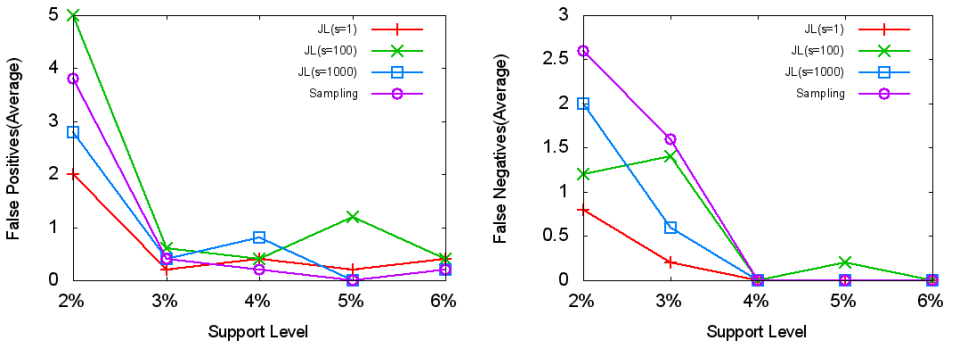
False Positives False Negatives

Fig. 2. Association Mining Results Varying Sketch Size



False Positives False Negatives

Fig. 3. Association Mining Results Varying Confidence

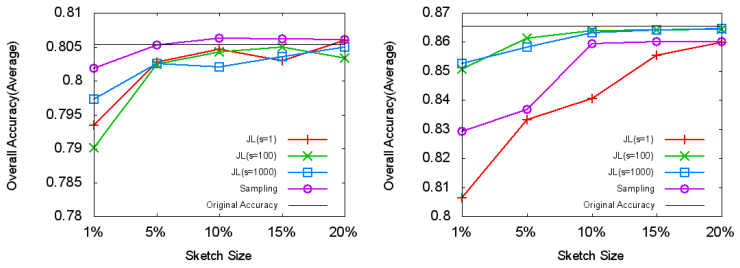


False Positives False Negatives

Fig. 4. Association Mining Results Varying Support

the number of invalid associations returned by the algorithm) and false negatives (the number of valid assocations not returned by the algorithm). For the association rules mining, this is a better picture of the accuracy than overall accuracy, since the true positives are so much rarer than the true negatives. Figure 2 shows the results when we varied the compaction ratio. JL and sampling are very similar in terms of accuracy, with a slight overall edge to JL. Note that by the time we reach a compression ratio of 10%, no more false negatives arise in any JL sketching (regardless of sparsity), or in the sampling protocol.

Figures 3 and 4 show the results varying the required confidence and required support, respectively. As one might expect, there is no discernable correlation between these variables and the accuracy of the approximation for it. A larger error rate generally indicates that there are more itemsets near the exact required value, which means a smaller error in the dot product might result in the incorrect rejection or acceptance of an itemset. This is especially true for a support value of 2%, since below 2%, the number of supported itemsets increases dramatically.



Naive Bayes C4.5 Decision Tree

Fig. 5. Naive Bayes and C4.5 Results Varying Sketch Size

Naive Bayes Classification Figure 5 (left side) shows the results for naive Bayes classification. JL and sampling, again, perform quite similiarly. The accuracy, as expected, increases with the sketch size. The thin black line on the graph represents the accuracy of the naive Bayes classification on the original, uncompacted data. The accuracy of the approximation for both JL and sampling hovers right around the original accuracy, and in some cases performs better. This is understandable due to the machine learning phenomenon of *overfitting*. When a model is built on some data, it performs quite well on the data it was trained with, but the model will not perform as well on test data. When this happens, the model is said to overfit the training data. Often some noise is added to the model to remove the overfitting problem. The approximation of the dot product can provide such noise. Thus, the approximations can achieve higher accuracy than the exact result.

C4.5 Decision Tree Figure 5 (right side) shows the results for C4.5 decision tree classification. The results are consistent with our findings in other tasks. Interestingly enough, the more sparse versions of JL outperformed the unabridged ($s = 1$) version. This is likely due to the fact that the sparse vectors provided slightly less distortion in the multiplication, resulting in a closer approximation for the dot product. In this case, as opposed to the naive Bayes case, the original tree provides a higher degree of accuracy, mainly because the C4.5 algorithm implements noise introduction by pruning the tree after building it.

6.2 Efficiency

In order to gauge the efficiency of our sketching protocols, we ran several timing experiments. The machine used was an AMD Athlon(tm) 64X2 dual core processor 4800T at 2.5 GHz with 2GB of RAM, running Windows Vista, and running on the Java 6 Standard runtime environment update 24. As our sub-protocol for exact dot product computation, we use the protocol of Goethals, et al [11], as it is provably secure, and lends itself well to improvement from our sketching protocols.

First, we ran several timing experiments computing the complete dot product of zero-one vectors of size 1000. The average time for the computation was 105 seconds. To ensure that the algorithm scaled linearly, we then ran it on vectors of size 2000, and the average computation time was 211 seconds. So, we determined the time-per-element in the dot product protocol to be .105 seconds. From this point forward, we computed the runtime of the approximate protocol in terms of the run time of the exact protocol by counting the time not involved in the computation of dot products, then adding it to the estimated dot product calculation time based on the previous timing experiments. The actual formula used was:

$$\frac{t_i + .105s \cdot n_d \cdot compactionRatio \cdot n}{.105s \cdot n_d \cdot n}$$

Where t_i is the time involved in the sketching, n_d is the number of dot products performed, n is the length of the vectors involved, and *compactionRatio* is the fraction of the original vector's size which is retained by the sketching protocol. The results for three different sketching algorithms and five different compaction ratios are be are seen in figure 7.

In all cases, the algorithms are much faster than the exact algorithm. Because it produces a matrix with 1 or -1 for every value, JL with $s = 1$ has a large amount of pre-processing before it can apply the projection to each vector, which again, takes time. This runtime can be improved by using the sparsity factor. We chose, however, to present the worst case, as it is still much better than the original runtime. The association rules mining process involved the fewest number of dot products computed. Therefore, the preprocessing and other portions of the algorithms took up a greater percentage of the time in association rules mining. The Naive Bayes process had orders of magnitude more dot product calculations, so the overall time was dominated by the number of dot product calculations necessary.

In the decision tree case, the number of dot products computed varied with the algorithm involved. This is because we use the dot products to determine if a node is to be split. If a split is found to be not useful, the split will not occur. The compaction introduced enough error into the calculation that splits with very little information gain were not even attempted, resulting in much fewer dot products being calculated. The different algorithms all calculated far fewer dot products at every compaction level, resulting in a much greater efficiency increase.

Mining Task	Sketching Protocol	Compaction Ratio 1%	5%	10%	15%	20%
Association Mining	JL(s=1)	1.23301%	5.97532%	12.06352%	17.93215%	23.90036%
	JL(s=100)	1.10189%	5.50682%	11.01542%	16.62495%	22.19586%
	JL(s=1000)	1.09683%	5.43911%	10.97853%	16.49222%	22.01157%
	Sampling	1.07975%	5.09715%	10.08799%	15.07924%	20.08823%
Naive Bayes	JL(s=1)	1.10388%	5.50989%	11.01977%	16.52684%	22.03317%
	JL(s=100)	1.09024%	5.36809%	10.86241%	16.24925%	21.25196%
	JL(s=1000)	1.08882%	5.33216%	10.71943%	15.98638%	21.05157%
	Sampling	1.01317%	5.01338%	10.01391%	15.01437%	20.01472%
C4.5 Decision Tree	JL(s=1)	0.20356%	0.22841%	0.65546%	1.89563%	2.72094%
	JL(s=100)	0.18926%	0.19452%	0.59234%	1.71828%	2.64378%
	JL(s=1000)	0.17586%	0.19623%	0.58419%	1.65025%	2.61224%
	Sampling	0.02198%	0.19146%	0.80031%	1.44452%	2.53887%

Fig. 6. Efficiency Results: Percent of the Exact Algorithm Runtime

7 Conclusions

We have presented several interesting approximation techniques for the secure compuation of the dot product of two vectors. These protocols can be applied to many different data mining tasks, and can provide an efficiency increase to any protocol that uses a secure dot product as a sub-protocol.

7.1 Future Work

In the future, we plan to explore the use of these dot product protocols in other data mining tasks, such as support vector machines, neural networks, and clustering. We also plan to consider carefully the notion of a secure approximation, and determine to what extent the restrictions posed by our security model can be relaxed.

8 Acknowledgements

This work was partially supported by Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, National Institutes of Health Grant 1R01LM009989,

National Science Foundation (NSF) Grant Career-CNS-0845803, and NSF Grants CNS-0964350, CNS-1016343. It is also based upon work supported by the AFOSR in-house project No. 11RI01COR, the AFRL in-house Job Order Number GGI-HZORR, with AFRL/RI Information Information Institute VFRP No. 57739-1095380-1.

References

1. D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.
2. C. Aggarwal and P. Yu. A general survey of privacy-preserving data mining models and algorithms. *Privacy-preserving data mining*, pages 11–52, 2008.
3. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.
4. A. Asuncion and D. Newman. UCI machine learning repository, 2007.
5. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.
6. W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, page 102. IEEE Computer Society, 2001.
7. C. Dwork. Differential privacy: A survey of results. *Theory and Applications of Models of Computation*, pages 1–19, 2008.
8. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strausse, and R. Wright. Secure multiparty computation of approximations. *ACM transactions on Algorithms (TALG)*, 2(3):435–472, 2006.
9. D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522. ACM, 2003.
10. B. Goethals. Frequent itemset mining implementations repository, 2005.
11. B. Goethals, S. Laur, H. Lipmaa, and T. Mielik
"ainen. On private scalar product computation for privacy-preserving data mining. *Information Security and Cryptology–ICISC 2004*, pages 104–120, 2005.
12. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1965.
13. Z. Huang, W. Du., and B. Chen. Deriving private information from randomized data, 2005.
14. I. Ioannidis, A. Grama, and M. Attallah. A secure protocol for computing the dot-products in clustered and distributed environments. In *International Conference on Parallel Processing, 2002.*, pages 379–384. IEEE, 2002.
15. W. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1–1, 1984.
16. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1026–1037, 2004.
17. M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. *Advances in Knowledge Discovery and Data Mining*, pages 515–524, 2009.

18. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM 2003. Third IEEE International Conference on Data Mining, 2003.*, pages 99–106. IEEE, 2003.
19. P. Li, T. Hastie, and K. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.
20. Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology-CRYPTO 2000*, pages 36–54, 2000.
21. K. Liu, C. Giannella, and H. Kargupta. An attackers view of distance preserving maps for privacy preserving data mining. *Knowledge Discovery in Databases: PKDD 2006*, pages 297–308, 2006.
22. 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*, pages 92–106, 2006.
23. A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC, 1997.
24. B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.
25. L. Qiu, Y. Li, and X. Wu. Preserving privacy in association rule mining with bloom filters. *Journal of Intelligent Information Systems*, 29(3):253–278, 2007.
26. P. Ravikumar, W. Cohen, and S. Feinberg. A secure protocol for computing string distance metrics. In *Proceedings of the Workshop on Privacy and Security Aspects of Data Mining at the International Conference on Data Mining*, pages 40–46. IEEE, 2004.
27. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.
28. J. Vaidya and C. Clifton. Privacy preserving naıve bayes classifier for vertically partitioned data. In *2004 SIAM International Conference on Data Mining, Lake Buena Vista, Florida*, pages 522–526, 2004.
29. J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. *Data and Applications Security XIX*, pages 924–924, 2005.
30. J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.
31. W. Wang, M. Garofalakis, and K. Ramchandran. Distributed sparse random projections for refinable approximation. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 331–339. ACM, 2007.