Under consideration for publication in Knowledge and Information Systems

A Local Asynchronous Distributed Privacy Preserving Feature Selection Algorithm for Large Peer-to-Peer Networks

Kamalika Das[1], Kanishka Bhaduri[2] and Hillol Kargupta[3]

[1]Stinger Ghaffarian Technologies Inc., IDU Group, NASA Ames Research Center, Moffett Field, CA-94035, kdas1@cs.umbc.edu; [2]Mission Critical Technologies Inc., IDU Group, NASA Ames Research Center, Moffett Field, CA-94035, Kanishka.Bhaduri-1@nasa.gov; [3]Department of CSEE, University of Maryland, Baltimore County, Baltimore, MD-21250 and AGNIK LLC, MD, hillol@cs.umbc.edu

Abstract. In this paper we develop a local distributed privacy preserving algorithm for feature selection in a large peer-to-peer environment. Feature selection is often used in machine learning for data compaction and efficient learning by eliminating the curse of dimensionality. There exist many solutions for feature selection when the data is located at a central location. However, it becomes extremely challenging to perform the same when the data is distributed across a large number of peers or machines. Centralizing the entire dataset or portions of it can be very costly and impractical because of the large number of data sources, the asynchronous nature of the peer-to-peer networks, dynamic nature of the data/network and privacy concerns. The solution proposed in this paper allows us to perform feature selection in an asynchronous fashion with a low communication overhead where each peer can specify its own privacy constraints. The algorithm works based on local interactions among participating nodes. We present results on real-world datasets in order to performance of the proposed algorithm.

Keywords: privacy preserving; data mining; feature selection; distributed computation

1. Introduction

Feature selection, as the name suggests, is a popular machine learning/data mining tool often used for identifying subsets of features from a dataset for the

Received Aug 15, 2008
Revised Jul 17, 2009
Accepted Oct 10, 2009

purpose of building robust statistical models. From a theoretical perspective, optimal feature selection requires exhaustive search of all the possible subsets *i.e.* requires one to compute the power set of the features. This becomes impractical for a large number of features. Much research has therefore focussed on finding heuristic search techniques such as genetic algorithms, simulated annealing, hill climbing and more. Optimality conditions such as Akaike information criterion (AIC) and Bayesian information criterion (BIC) have also been developed to prune the search space efficiently. Finally, researchers have come up with certain feature selection measurements such as gini index, misclassification gain, entropy, χ^2 which are all based on certain heuristics.

Feature selection, when all the data is located at a central database, is a fairly well understood problem. Direct application of one of these techniques may provide a satisfactory result. However, there exist emerging technologies where the data is not located at a central repository, rather distributed across a large number of nodes in a network. The next generation of Peer-to-Peer (P2P) networks provides an example. P2P systems such as Gnutella, BitTorrents, e-Mule, Kazaa, and Freenet are increasingly becoming popular for many applications that go beyond downloading music without paying for it. Users and researches are no longer interested in only using P2P networks as a data repository; they may be interested in extracting the knowledge locked in the data. To harness the power of P2P computing, emerging applications such as bioinformatics[1] and client-side web mining (Liu, Bhaduri, Das, Nguyen and Kargupta, 2006)(Das, Bhaduri, Liu and Kargupta, 2008) have been suggested which would need the support of advanced data analysis techniques. Centralized algorithms cannot be executed in these scenarios due to (1) massive scale of such networks, (2) high cost of centralization, (3) dynamic nature of the data, and (4) privacy constraints. Therefore analysis of data in such networks will require us to develop distributed data mining algorithms capable of working in such large-scale environments.

In this paper we have addressed the problem of feature selection in a large P2P network. The proposed P2P feature selection algorithm (*PAFS*) incorporates three popular feature selection criteria: misclassification gain, gini index and entropy measurement. As a first step we have shown how these measurements can be evaluated in a P2P network without any data centralization. Our algorithm is provably correct in the sense that it converges to the correct result compared to centralization. We have shown that the proposed algorithm has bounded communication and is therefore *local*. Moreover, each peer can specify its own privacy constraint and the algorithm uses a multi-objective optimization criteria to satisfy the privacy constraints of each peer. The proposed algorithm is likely to contribute in distributed search, information retrieval, and link analysis in P2P networks.

The rest of this paper is organized as follows. Section 2, discusses the motivation behind this work. Section 3 briefly presents some existing research in this area. Section 4 presents the problem definition and show how distributed versions of the three feature selection criteria can be developed. It presents the building blocks of the algorithm in Section 5. The paper offers the details of the algorithm in Section 6. In Section 7 we theoretically analyze the algorithm. It demonstrates the performance of the algorithm in Section 8. Finally, the paper

[1] `http://www.bcgsc.ca/chinook`

ends with a conclusion section (Section 9) where we summarize this work and discuss some future directions of research.

2. Motivation

Consider a company X that wants to sell products Y and Z in a consumer market. It wants to start a marketing campaign in order to attract the potential customers from a large population. With the current technology, X can achieve this in several ways: (1) by broadcasting (for example, spamming) its advertisement to a large number of people with the hope that some of it will fall in the hands of interested people, (2) using the social networking sites for advertisements, (3) sending to only those whose address X knows. Broadcasting can be costly from the service provider's perspective and it is certainly quite inefficient. This is a fairly primitive concept and unlikely to scale to produce a large number of interested clients. If everyone starts spamming the network then it may not take a whole lot of time before the system is brought down to its knees. Also, most individuals view spams as an unwelcome mode of communication. On the other hand, the social networking sites have business interests that are driven by the economics of profit-making. That means if a match-making or a social communication creates value for the business then eventually those are the ones that will be promoted by the site. Option number (3) may work only when company X knows about the targeted customer addresses ahead of time possibly through extensive market research.

Scalable match-making between the source and the consumer of some information or a service in a large distributed population connected over a network like the Internet would require developing technology for at least the following two key problems:

1. Finding a balance between our reliance upon single-entity owned centralized client-server model of computation and decentralized emergence of global behavior through local interactions.
2. Privacy-sensitive content analysis and match-making between the source and the interested parties in a distributed decentralized environment.

Although we have a long way to go in solving these problems, we are starting to see some possible directions. The methodology for achieving global data analysis through efficient but strictly local interactions is drawing attention in many domains. For example, peer-to-peer (P2P) networks have been gaining popularity in many domains. P2P systems work by using the computing power, storage, and bandwidth of the participants of a network. Unlike client-server systems, P2P systems do not rely upon the servers to carry out most of the computation and storage-intensive tasks. P2P systems such as Gnutella, Napster, e-Mule, Kazaa, and Freenet are increasingly becoming popular for many applications that go beyond downloading music without paying for it.

Match-making between the source and the consumers of some information in such P2P environments would require distributed data clustering, feature selection, similarity search and classification algorithms that work in a decentralized communication efficient manner. P2P distributed data mining algorithms (Kargupta and Sivakumar, 2004)(Datta, Bhaduri, Giannella, Wolff and Kargupta, 2006) offer many interesting applications such as P2P search, interest-

based community formation, and P2P electronic commerce. Clearly, maintaining users' privacy will be an important issue to address before making P2P algorithms our *modus operandi*.

P2P data mining may offer some pieces of the solution to many problems that require such match-making. Usually, product placement in a wide market requires extensive market research in order to identify the appropriate consumer-base. Consumer segmentation plays an important role in that and various data mining techniques such as feature selection, clustering, and classification are often very useful for that. In order to do that we need consumer behavior data. There are several ways one can collect such data and there exist several techniques that can be used for Internet-marketing. P2P consumer networks may provide one possible way to get access to such data. Recommender systems based on P2P file sharing system activities are already gaining popularity. Similar P2P environments connecting the web-browsers of multiple users (Liu et al., 2006) may provide wealth of information as long as we pay due attention to privacy issues.

In this paper we illustrate the possibilities by considering the well-known feature selection problem in a distributed P2P environment. It proposes a technique for selecting important features from the dataset by using a communication efficient privacy preserving distributed algorithm. This algorithm is provably correct and uses only local interactions among peers for a scalable solution, ideal for P2P data mining. In our model, each peer has the flexibility to define its own privacy requirements.

3. Related Work

Related work is presented in the following three subsections.

3.1. Peer-to-Peer Data Mining

Distributed data mining (DDM) deals with the problem of data analysis in environments with distributed data, computing nodes, and users. Kargupta (Kargupta and Sivakumar, 2004) presents a detailed overview of this topic. P2P data mining has emerged as an active area of research under DDM for which the proposed algorithms are asynchronous, communication-efficient and scalable. Examples include the association rule mining algorithm (Wolff and Schuster, 2004), k-Means clustering (Datta, Giannella and Kargupta, 2006), top-l inner product identification (Das et al., 2008), decision tree induction (Bhaduri, Wolff, Giannella and Kargupta, 2008) and more.

3.2. Privacy in P2P Network

P2P networks have recently emerged as huge information systems for collaborative computation such as file sharing, distributed computing, real-time telephone and tv, and academic applications. However, free flow of information is frequently prohibited by legal obligations or by commercial and personal concerns. Privacy preserving data mining in P2P networks aims to solve this problem by allowing users to share their data without revealing the sensitive information. In a

large scale cooperative computation environment, each node has a private input x_i. They wish to jointly compute the output $f(x_1, x_2, \ldots, x_n)$ of some common function f. At the end of the data mining process, nothing but the output should be revealed. PPDM solves this problem by allowing useful data patterns to be extracted without revealing sensitive data (Clifton, Kantarcioglu, Vaidya, Lin and Zhu, 2003). Gilburd *et al.* presents a privacy model called k-TTP for large-scale distributed environment (Gilburd, Schuster and Wolff, 2004). Kargupta *et al.* (Kargupta, Das and Liu, 2007) presents a game theoretic solution for dealing with the collusion problem in secure sum computation in a large distributed environment. Teng and Du (Teng and Du, 2009) present a hybrid technique for PPDM. They combine randomization and SMC protocols to achieve better accuracy and lower running times than these algorithms acting independently.

3.3. Distributed Feature Selection

Maulik *et al.* (Maulik, Bandyopadhyay and Trinder, 2001) considered the problem of semi-automatic feature extraction from remotely sensed images. Their proposed solution uses a stochastic optimization technique *viz.* simulated annealing for minimizing the energy of an active contour within a specified image region. Energy of a pixel value is computed based on its distance to the image edges. The feature extraction process starts with a user description of the possible features of interest. The optimization technique then improves this search in a very localized fashion. The authors have compared their technique to existing techniques and shown the superiority of their approach.

Dimensionality Reduction in multidimensional time series for efficient indexing and searching was studied by Keogh *et al.* (Keogh, Chakrabarti, Pazzani and Mehrotra, 2001). Traditionally PCA, SVD, fourier transform, random projection, wavelet transform etc. have been extensively used for dimensionality reduction. All these techniques suffer from one major drawback — one loses comprehensibility of the data in the reduced space. The technique proposed in (Keogh et al., 2001) Piecewise Aggregate Approximation (or PAA) overcomes this drawback while bearing same or better time complexity compared to existing methods. The major idea is to discretize the time series by considering a set of non-overlapping equi-width bins and replacing the amplitude of the time series across each bin by the average amplitude across that bin. Experimental results reported in the paper prove the feasibility of the approach.

An algorithm for distributed document clustering of server web logs have been developed by Sayal and Scheuermann (Sayal and Scheuermann, 2001). In their algorithm first the web logs are distributed across multiple machines. Local models are build at each machine using traditional document clustering algorithms. Finally, these clusterings are then merged using a meta-learning approach.

Das *et al.* (Das et al., 2008) presents an algorithm for identifying significant inner product entries from data horizontally distributed in a P2P network. The proposed algorithm blends the concept of ordinal and cardinal sampling in order to fix the number of samples that needs to be drawn to estimate the top p percentile of the population. This work differs from the work presented in this paper in two important aspects: (1) the algorithm proposed here is eventually correct in the sense that it converges to the correct result when the computation terminates, and (2) unlike (Das et al., 2008), the current work pays careful atten-

tion to user privacy based on a recently proposed multi-objective optimization framework (Das, Bhaduri and Kargupta, 2009).

Several algorithms have been proposed in the literature for knowledge extraction from distributed databases. Clustering using PCA (Kargupta, Huang, Sivakumar and Johnson, 2001), bayesian network learning (Chen, Sivakumar and Kargupta, 2004), classification (Cho and Wüthrich, 2002), association rule mining (Schuster, Wolff and Trock, 2005) are some examples. Information retrieval from distributed data sources have been explored by Jung (Jung, 2009). The author formulates a technique for computing precision and recall for distributed datasets.

4. Distributed Feature Selection Problem Definition

Efficient match-making in a distributed environment requires proper representation construction and scalable similarity search. Feature selection plays an important role in both of these steps. Feature selection has been an active research area in pattern recognition, statistics, and data mining communities. In inductive function learning, the central idea of feature selection is to choose a subset of features which can correctly predict the output (the target function) and thereby remove the features with lesser predictive capability. Overall, feature selection techniques usually make data mining techniques (*e.g.* clustering, classification) stronger by constructing an appropriate representation that considers only the relevant features. In the past, several techniques have been developed for feature selection from high dimensional data. These include information gain, mutual information, Gini index, χ^2 statistic, correlation coefficient, PCA analysis and more. While each of these techniques have their own merits and demerits, we focus on some of the information theoretic techniques in this paper and show how distributed versions of these problems can be developed. Interested readers are referred to Yang and Pederson (Yang and Pedersen, 1997), Liu and Motoda (Liu and Motoda, 1998) for detailed analysis.

4.1. Problem Definition

Consider a P2P network where each node represents a user. Each node has a web-based recommender system (*e.g.* a web-browser plug-in that allows rating music CDs or books). Each user does its own share of rating and that creates a locally labeled data set. In absence of a centralized server that would collect this information (note that we dealing with a P2P application), we need a distributed approach to exploit this information. If an aspiring musician now wants to reach out and inform the targeted audience for his or her kind of music then one could use this distributed rating-data for designing a well directed marketing campaign. One could learn a classifiers from this distributed data and use that classifier to label new users. However, we first need to figure out which features are more relevant to this classification problem. This is essentially a feature selection problem. Rest of this section formally describes this problem.

Let D denote a collection of data tuples with class labels where each tuple is a $p+1$ dimensional vector $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$, the first p dimensions corresponding to the features and the last corresponding to the class label. We assume that each feature is categorical, *i.e.*, $\mathcal{A}_i$ takes a value from the finite set

Feature value (A_i)	$Class$=0	$Class$=1	($Class$=0)+($Class$=1)
0	$x_{i,00}$	$x_{i,01}$	$x_{i,0\cdot}$
1	$x_{i,10}$	$x_{i,11}$	$x_{i,1\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_i - 1$	$x_{i,(m_i-1)0}$	$x_{i,(m_i-1)1}$	$x_{i,(m_i-1)\cdot}$

Table 1. Number of entries of feature $\mathcal{A}_i$ and the class.

$\{0, \ldots, m_i - 1\}$ ($\forall i = 1$ to p) and the class is binary, *i.e.* $C \in \{0, 1\}$ where $m_i - 1$ is the highest value of feature $\mathcal{A}_i$. Let $x_{i,a0}$ denote the number of examples in the set D for which $\mathcal{A}_i = a$ and $C = 0$ where $a \in [0 \ldots (m_i - 1)]$. Also $x_{i,a\cdot}$ denotes the number of tuples with $\mathcal{A}_i = a$, computed over all classes. Table 1 shows the different possible combinations of values for a feature $\mathcal{A}_i$.

In our scenario, we do not assume the data to be at a central location, rather distributed over a set of peers $P_1, P_2, \ldots, P_d$ connected to each other by an underlying communication infrastructure. More specifically, D is partitioned into d sets D_1 through D_d such that each peer has the same set of features, but different tuples *i.e.* $D = \bigcup_{i=1}^{d} D_i$. In the distributed data mining literature, this is referred to as horizontal data partition. $x_{i,a0}^{(\ell)}$ denotes the number of examples in the set D_ℓ for which $\mathcal{A}_i = a$ and $C = 0$ where $a \in [0 \ldots (m_i - 1)]$. Hence $x_{i,a0} = \sum_{\ell=1\ldots d} x_{i,a0}^{(\ell)}$, $x_{i,a1} = \sum_{\ell=1\ldots d} x_{i,a1}^{(\ell)}$ and $x_{i,a\cdot} = \sum_{\ell=1\ldots d} x_{i,a\cdot}^{(\ell)}$.

Misclassification gain, gini index and entropy are three popular metrics for information theoretic feature selection. While Gini index and entropy have been used in the past for feature selection, the choice of misclassification gain is not ad-hoc; it has been argued by Tan *et al.* (Tan, Steinbach and Kumar, 2006) that the choice of impurity functions has little effect on selecting the 'best' feature (see (Tan et al., 2006) page 158 for details). More empirical comparisons can be found in (Bhaduri et al., 2008). In the next three sections we develop distributed, locally computable and privacy preserving variants for each of these feature selection metrics.

4.2. Misclassification Gain

For a categorical feature $\mathcal{A}_i$, the misclassification impurity measure (Tan et al., 2006) for a particular value $\mathcal{A}_i = a$ is

$$MI_a(\mathcal{A}_i) = 1 - \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{x_{i,a\cdot}}$$

(where $x_{i,a\cdot}$ is the number of tuples with $\mathcal{A}_i = a$).

Theorem 1. Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of features and class label where $\mathcal{A}_i \in \{0, \ldots, m_i - 1\}$ and $C \in \{0, 1\}$ respectively. The feature with the highest misclassification gain A_{best} is the following:

$$A_{best} = \arg\max_{i \in [1\ldots p]} \left[\sum_{a=0}^{m_i - 1} |x_{i,a0} - x_{i,a1}| \right]$$

Proof. The misclassification gain difference between $\mathcal{A}_i$ and $\mathcal{A}_j$, denoted by $MG(\mathcal{A}_i, \mathcal{A}_j)$, is

$$\begin{aligned}
MG(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \times [MI_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \times [MI_b(\mathcal{A}_j)] \\
&= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) - \sum_{a=0}^{m_i-1} \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{|D|} \\
&\quad - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) + \sum_{b=0}^{m_j-1} \frac{max\left(x_{j,b0}, x_{i,b1}\right)}{|D|} \\
&= (1-1) + \sum_{b=0}^{m_j-1} \frac{max\left(x_{j,b0}, x_{i,b1}\right)}{|D|} - \sum_{a=0}^{m_i-1} \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{|D|}
\end{aligned}$$

Since the maximum is the average plus half the absolute difference, this is equal to

$$\begin{aligned}
MG(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{b=0}^{m_j-1} \frac{(x_{j,b0} + x_{j,b1})}{2|D|} + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} \\
&\quad - \sum_{a=0}^{m_i-1} \frac{(x_{i,a0} + x_{i,a1})}{2|D|} - \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|} \\
&= \sum_{b=0}^{m_j-1} \frac{(x_{j,b0} + x_{j,b1})}{2|D|} - \sum_{a=0}^{m_i-1} \frac{(x_{i,a0} + x_{i,a1})}{2|D|} \\
&\quad + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} - \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|} \\
&= \frac{1}{2} - \frac{1}{2} + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} - \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|}
\end{aligned}$$

Therefore, choosing the feature with the highest misclassification gain is equivalent to maximizing the quantity $\sum_{a=0}^{m_i-1} |x_{i,a0} - x_{i,a1}|$ for any feature $\mathcal{A}_i$. Thus, according to the misclassification gain function, the best feature is the following:

$$\mathcal{A}_{best} = \underset{i \in [1 \ldots p]}{\arg\max} \left[\sum_{a=0}^{m_i-1} |x_{i,a0} - x_{i,a1}| \right]$$

□

Note that, for a distributed setup, selecting the best feature according to the misclassification gain is equivalent to distributed computation of the following sum:

$$\sum_{a=0}^{m_i-1}\left|\sum_{\ell=1}^{d}x_{i,a0}^{(\ell)}-\sum_{\ell=1}^{d}x_{i,a1}^{(\ell)}\right|\Rightarrow\sum_{a=0}^{m_i-1}\left|\sum_{\ell=1}^{d}\left\{x_{i,a0}^{(\ell)}-x_{i,a1}^{(\ell)}\right\}\right| \quad (1)$$

for each feature $\mathcal{A}_i$.

4.3. Gini Index

For a categorical feature $\mathcal{A}_i$, the Gini measure (Tan et al., 2006) for a particular value $\mathcal{A}_i = a$ is

$$Gini_a(\mathcal{A}_i) = 1 - \left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2$$

(where $x_{i,a\cdot}$ is the number of tuples with $\mathcal{A}_i = a$).

Theorem 2. Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of features and class as defined in the notations section where $\mathcal{A}_i$ and C takes the values between $\{0, \ldots, m_i - 1\}$ and $\{0, 1\}$ respectively. The feature with the highest Gini index A_{best} is the following:

$$A_{best} = \arg\max_{i\in[1\ldots p]}\left[\sum_{a=0}^{m_i-1}\left\{\frac{(x_{i,a0})^2+(x_{i,a1})^2}{x_{i,a\cdot}}\right\}\right]$$

Proof.

$$\begin{aligned}
Gini(\mathcal{A}_i,\mathcal{A}_j) &= \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right)\times[Gini_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right)\times[Gini_b(\mathcal{A}_j)]\\
&= \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right) - \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right)\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right)\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2\\
&\quad - \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right) + \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right)\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right)^2 + \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right)\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)^2\\
&= \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right)\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right)^2 + \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right)\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)^2\\
&\quad - \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right)\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right)\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2\\
&= \frac{1}{|D|}\left[\sum_{b=0}^{m_j-1}\left\{\frac{(x_{j,b0})^2+(x_{j,b1})^2}{x_{j,b\cdot}}\right\} - \sum_{a=0}^{m_i-1}\left\{\frac{(x_{i,a0})^2+(x_{i,a1})^2}{x_{i,a\cdot}}\right\}\right]
\end{aligned}$$

Therefore, the best feature is the one which maximizes the following quantity:

$$\mathcal{A}_{best} = \arg\max_{i\in[1\ldots p]}\left[\sum_{a=0}^{m_i-1}\left\{\frac{(x_{i,a0})^2+(x_{i,a1})^2}{x_{i,a\cdot}}\right\}\right]$$

□

As before, for the distributed setup, the following quantity needs to be evaluated across all the peers for every feature $\mathcal{A}_i$:

$$\sum_{a=0}^{m_i-1} \left\{ \frac{\left(\sum_{\ell=1}^{d} x_{i,a0}^{(\ell)}\right)^2 + \left(\sum_{\ell=1}^{d} x_{i,a1}^{(\ell)}\right)^2}{\sum_{\ell=1}^{d} x_{i,a\cdot}^{(\ell)}} \right\} \tag{2}$$

Therefore, two separate distributed sum computation instances need to be invoked: (1) $\sum_{\ell=1}^{d} x_{i,a0}^{(\ell)}$ and (2) $\sum_{\ell=1}^{d} x_{i,a1}^{(\ell)}$.

4.4. Entropy

For a categorical feature $\mathcal{A}_i$, the entropy measure (Tan et al., 2006) for a particular value $\mathcal{A}_i = a$ is

$$Entropy_a(\mathcal{A}_i) = -\left[\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)\log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) + \left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\right]$$

(where $x_{i,a\cdot}$ is the number of tuples with $\mathcal{A}_i = a$) and all logarithm is taken with base 2.

Theorem 3. Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of features and class as defined in the notations section where $\mathcal{A}_i$ and C takes the values between $\{0, \ldots, m_i - 1\}$ and $\{0, 1\}$ respectively. The feature with the highest entropy A_{best} is the following:

$$A_{best} = \arg\max_{i \in [1\ldots p]} \left[\sum_{a=0}^{m_i-1} \left\{(x_{i,a0})\log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) + (x_{i,a1})\log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\right\}\right]$$

Proof.

$$\begin{aligned}
Entropy(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a\cdot}}{|D|}\right) \times [Entropy_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b\cdot}}{|D|}\right) \times [Entropy_b(\mathcal{A}_j)] \\
&= -\sum_{a=0}^{m_i-1}\left(\frac{x_{i,a0}}{|D|}\right)\log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) - \sum_{a=0}^{m_i-1}\left(\frac{x_{i,a1}}{|D|}\right)\log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right) \\
&\quad + \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b0}}{|D|}\right)\log\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right) + \sum_{b=0}^{m_j-1}\left(\frac{x_{j,b1}}{|D|}\right)\log\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right) \\
&= \frac{1}{|D|}\left[\sum_{b=0}^{m_j-1}(x_{j,b0})\log\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right) + \sum_{b=0}^{m_j-1}(x_{j,b1})\log\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)\right] \\
&\quad - \frac{1}{|D|}\left[\sum_{a=0}^{m_i-1}(x_{i,a0})\log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) + \sum_{a=0}^{m_i-1}(x_{i,a1})\log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\right]
\end{aligned}$$

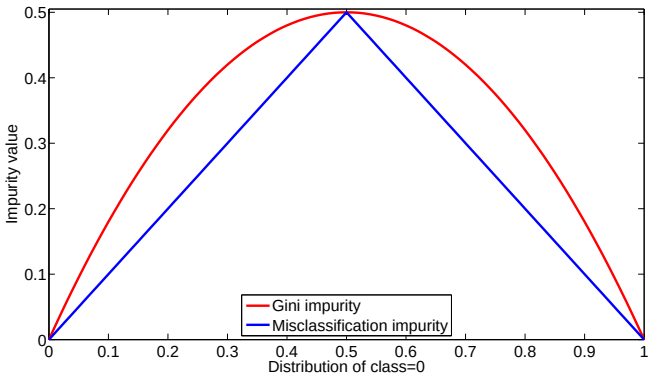


Fig. 1. Plot of Gini index and Misclassification gain for binary class distribution.

Therefore, the best feature is the one which maximizes the following quantity:

$$\mathcal{A}_{best} = \underset{i\in[1\ldots p]}{\arg\max} \left[\sum_{a=0}^{m_i-1} \left\{ (x_{i,a0}) \log \left(\frac{x_{i,a0}}{x_{i,a\cdot}} \right) + (x_{i,a1}) \log \left(\frac{x_{i,a1}}{x_{i,a\cdot}} \right) \right\} \right]$$

□

Therefore, the quantity that needs to be evaluated across all the peers for every feature $\mathcal{A}_i$ is:

$$\sum_{a=0}^{m_i-1} \left\{ \left(\sum_{\ell=1}^{d} x_{i,a0}^{(\ell)} \right) \log \left(\frac{\sum_{\ell=1}^{d} x_{i,a0}^{(\ell)}}{\sum_{\ell=1}^{d} x_{i,a\cdot}^{(\ell)}} \right) + \left(\sum_{\ell=1}^{d} x_{i,a1}^{(\ell)} \right) \log \left(\frac{\sum_{\ell=1}^{d} x_{i,a1}^{(\ell)}}{\sum_{\ell=1}^{d} x_{i,a\cdot}^{(\ell)}} \right) \right\} \quad (3)$$

Two separate distributed sum computation instances need to be invoked: (1) $\sum_{\ell=1}^{d} x_{i,a0}^{(\ell)}$ and (2) $\sum_{\ell=1}^{d} x_{i,a1}^{(\ell)}$. The following figure (Figure 1) shows both the Gini index and misclassification gain function for a binary class distribution problem.

5. Setup for Distributed Privacy Preserving Feature Selection

In a horizontally partitioned distributed data mining scenario, the data is distributed across all the peers in such a way that each peer observes different instances of the entire data for all features.

As mentioned before, $P_1, P_2, \ldots, P_d$ denotes the set of peers connected to each other by an underlying communication infrastructure. The network can be viewed as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{P_1, P_2, \ldots, P_d\}$ denotes the set of vertices and $\mathcal{E}$ denotes the set of edges. Let $\Gamma_{i,\alpha}$ denote the set of neighbors of P_i at a hop-distance of α from P_i and $|\Gamma_{i,\alpha}|$ denote the size of this set, *i.e.*, the number of neighbors in the α-neighborhood. $\alpha = 1$ refers to the set of immediate neighbors. Further, let $\Phi_{d\times d}$ denote the connectivity matrix or topology matrix of $\mathcal{G}$ representing the network where

$$\phi_{ij} = \begin{cases} 1 \text{ if } i,j \in \mathcal{E}, i \neq j \\ -|\Gamma_{i,1}| \text{ if } i,j \in \mathcal{E}, i = j \\ 0 \text{ otherwise} \end{cases}$$

Each of the misclassification gain, gini index and entropy based feature selection techniques require independent computation of one (misclassification gain) or two (Gini index, entropy) sums on the count of the class labels for each of the p features of the data. Thus, the distributed p2p feature selection algorithm should be able to compute distributed summations on the partial data in an asynchronous privacy preserving fashion and reach a globally correct result. In this section, we present the three important building blocks that we use for developing our distributed privacy preserving feature selection algorithm. We combine an asynchronous distributed averaging technique with secure sum protocol to form our privacy preserving sum computation technique, in the context of the Bayes optimal model of privacy.

5.1. Distributed Averaging

Average (and hence sum computation) is one of the most fundamental primitives useful for many advanced data mining tasks. In distributed averaging, each peer P_i has a real-valued number x_i and the task is to compute $\Delta = \frac{1}{d}\sum_{i=1}^{d} x_i$ without centralizing all the data. Researchers have proposed several solutions to this problem. In this paper, we explore the one proposed by Scherber and Papadopoulos (Scherber and Papadopoulos, 2005). The basic idea is as follows. Each peer maintains $z_i^{(t)}$, which is the current estimate of Δ at time t. $z_i^{(0)}$ is initialized to x_i. Each peer asynchronously updates $z_i^{(t)}$ based on the updates it receives from other neighbors. For any time t, the update rule can be written as (Scherber and Papadopoulos, 2005):

$$z_i^{(t)} = z_i^{(t-1)} + \rho \sum_{j \in \Gamma_i} \left[z_j^{(t-1)} - z_i^{(t-1)} \right]$$

It has been shown that as $t \to \infty$, the estimate of each peer converges to the average *i.e.* $z_i^t \to \Delta$.

The distributed averaging problem, as proposed in the literature, is not privacy preserving. Moreover it works only for symmetric network topologies, *i.e.* if peer P_i is a neighbor of P_j, then the reverse is true. However, our multi-objective optimization based privacy framework (which we discuss in details in Section 6) requires an asymmetric network topology. Therefore, the update rule for distributed averaging needs to be adapted for asymmetric topologies by generating a symmetric topology $\Phi^{''} = \Phi + \Phi^{\mathrm{T}}$ using the asymmetric topology Φ. Using this, the update rule for any peer P_i can be written as:

$$z_i^{(t)} = \left\{1 - 2\rho\left|\Gamma_{i,1}\right| - \rho(n_i^* - |\Gamma_{i,1}|)\right\} z_i^{(t-1)} + 2\rho \sum_{q \in \Gamma_{i,1}} z_q^{(t-1)} + \rho \sum_{q=1}^{n_i^* - |\Gamma_{i,1}|} z_q^{(t-1)} \quad (4)$$

Such an average computation algorithm can easily be adopted for computing the sum by pre-multiplying each x_i by d *i.e.* the input to the algorithm should now be $x_i \times d$.

5.2. Secure Sum Computation

Secure sum protocol (Clifton et al., 2003) computes the sum of a collection of numbers without revealing anything but the sum. It can be shown that the distribution of the output of the protocol is uniform and hence no hacker can obtain any more information from the sum other than what is implied by the sum itself. This protocol naturally lends itself to multi-party computation since the data resides at different nodes. However, execution of this protocol requires the existence of a ring topology over all the nodes. In the protocol assuming that the sum $S = \sum_{i=1}^{d} x_i$ is in the range $[0, N-1]$, the initiator of the protocol generates a random number R in the same range. It then does the following computation and sends the data to the next node in the ring:

$$(R + x_1) \bmod N$$

Any node 2 through d, gets the data of the previous node and performs the following computation:

$$y_i = (y_{i-1} + x_i) \bmod N = (R + \sum_{q=1}^{i} x_q) \bmod N,$$

where y_i is the perturbed version of local value x_i to be sent to the next peer $i+1$. The last peer sends the result to peer 1 which knows R, and hence can subtract R from y_d to obtain the actual sum. This sum is finally broadcast to all other users. This protocol assumes that the participants do not collude. However, it has been shown in the literature (Kargupta et al., 2007) that such an assumption is suboptimal and that rational parties would always try to collude. The analysis of privacy in the presence of colluding parties in the context of our feature selection algorithm is discussed in the next section.

The secure sum protocol is highly synchronous and is therefore unlikely to scale for large networks. However, combining the secure sum protocol with distributed averaging gives us a privacy preserving algorithm for feature selection under the Bayes optimal model of privacy.

5.3. Bayes Optimal Privacy Model

Bayes optimal privacy model (Machanavajjhala, Gehrke, Kifer and Venkitasubramaniam, 2006) quantifies or bounds the difference in information that any participant has before and after the data mining protocol is executed. This quantification can be presented in terms of prior and posterior distribution. Let X be a random variable which denotes the data value at each node. The value at node P_i is denoted by x_i. The prior probability distribution is $prior = P(X = x_i)$. Once the data mining process is executed, the participants can have some extra information. Given this, we define the posterior probability distribution as $posterior = P(X = x_i|\mathcal{B})$, where $\mathcal{B}$ models the extra information. Evfimievski *et al.* present a way of quantifying the Bayes optimal privacy breach.

Definition 1 ($\rho_1 - to - \rho_2$-privacy breach). (Evfimevski, Gehrke and Srikant, 2003). Let f_{prior} and $f_{posterior}$ denote the prior and posterior probability distributions of X. The $\rho_1 - to - \rho_2$ privacy breach is mathematically stated as follows: $f_{prior} \leq \rho_1$ and $f_{posterior} \geq \rho_2$, where $0 < \rho_1 < \rho_2 < 1$.

This definition is only suitable for a system in which all the nodes follow the same privacy model. However, in a large P2P network, such an assumption

is difficult to impose. Therefore, we extend this definition to a distributed data mining scenario in which each node defines its own privacy model.

Definition 2. [Multi-party $\rho_1 - to - \rho_2$ privacy breach] For the i-th peer P_i, privacy breach occurs if $f^i_{prior} \leq \rho_{1i}$ and $f^i_{posterior} \geq \rho_{2i}$. Multi-party $\rho_1 - to - \rho_2$ privacy breach occurs when the constraints are violated for any peer in the network *i.e.* $\forall i,\ f^i_{prior} \leq \rho_{1i}$ and $f^i_{posterior} \geq \rho_{2i}$, where $0 < \rho_{1i} < \rho_{2i} < 1$.

In the definition, the posterior probabilities of each peer can either be dependent or independent of each other. If the peers share the extra information ($\mathcal{B}$), their posterior distributions are also related. As discussed later, in our algorithm the dependence or the independence of the posterior probabilities does not change the privacy requirements.

6. Privacy Preserving Asynchronous Feature Selection

As noted in Section 4, distributed feature selection requires the peers to compute the sums of certain metrics defined over the partial feature vectors $\mathcal{A}_i$s (equations 1, 2, and 3) and comparing the sums across all the features for finding the best features from the information theoretic perspective. For distributed feature selection, each peer starts these sum computations. Since distributed sum computing is nothing but distributed averaging of scaled data, our algorithm uses a distributed averaging technique for the sum computation. However, since the existing distributed averaging techniques are not privacy preserving, we use a secure sum based averaging technique for privacy. Secure sum requires a ring topology. Therefore, depending on their privacy requirements, every ring forms a local ring by inviting other peers to join its computation ring and does a distributed averaging within its ring using the modified update rule described in equation 4.

Privacy is a social concept. In a distributed data mining environment, different peers have different notions and requirements of privacy. Due to sharing of private information in the process of computation, privacy of the users' data is threatened. If other users participating in the computation cheat, then the privacy of someone's data can be compromised. Every user in the network has a belief about the *threat* to its data privacy. The threat that a peer's data is exposed to can be considered as a measure of the lack of privacy of its data. Again, the amount of resources available to a peer varies across the network and hence, the cost (of computation and communication) a peer can bear to ensure its data privacy also varies. Therefore, every user in the network solves an optimization problem locally based on its cost and threat threshold, *i.e.* how much threat the user is willing to bear and how much resources it is willing to spend for ensuring that. For illustration purposes, we use a linear model for the objective function:

$$f^{obj}_i = w_{ti} \times threat - w_{ci} \times cost$$

where the *cost* is the total cost of communication of peer P_i within a neighborhood of size n^*_i and *threat* is the privacy breach that peer P_i assumes to be exposed to due to its participation in the data mining task. w_{ti} and w_{ci} are the weights (importance) associated with the *threat* and *cost*. These parameter values are local to each peer and are independent of the values chosen by any other peer in the network. In the next section we derive an expression for threat for secure sum in the presence of colluders.

Threat measure for secure sum with collusion (Kargupta and Sivakumar, 2004): Each peer forms a ring of size n_i^* (referred to as n in this section for sake of simplicity) in our algorithm. Let us assume that there are k ($k \geq 2$) nodes acting together secretly to achieve a fraudulent purpose. Let P_i be an honest node who is worried about its privacy. Let P_{i-1} be the immediate predecessor of P_i and P_{i+1} be the immediate successor of P_i. We will only consider the case when $n-1 > k \geq 2$ and the colluding nodes contain neither P_{i-1} nor P_{i+1}, or only one of them, then P_i is disguised by $n-k-1$ other nodes' values. The other cases are trivial. This can be represented as

$$\underbrace{\sum_{q=1}^{n-k-1} x_q}_{\text{denoted by Y}} + \underbrace{x_i}_{\text{denoted by X}} = \underbrace{S - \sum_{q=i+1}^{i+k} x_q}_{\text{denoted by W}},$$

where W is a constant and known to all the colluding nodes and S is the global sum. It can be shown that the posterior probability of x_i is:

$$f_{posterior}(x_i) = \frac{1}{(m+1)^{(n-k-1)}} \sum_{q=0}^{r} (-1)^q \binom{n-k-1}{q} \times \binom{n-k-1+(r-q)(m+1)+t-1}{(r-q)(m+1)+t}$$

where $z = W - x_i$ and $z \in \{0, 1, \ldots, m(n-k-1)\}$. $r = \lfloor \frac{z}{m+1} \rfloor$, and $t = z - \lfloor \frac{z}{m+1} \rfloor (m+1)$. Note that here we assume $x_i \leq W$, otherwise $f_{posterior}(x_i) = 0$. This posterior probability can be used to measure the threat faced by a peer while participating in the secure sum computation protocol, if there is collusion:

$$threat = Posterior - Prior = f_{posterior}(x_i) - \frac{1}{m+1} \tag{5}$$

It can be observed from this threat measure that (1) as k increases, the posterior probability increases, and (2) as n increases, the posterior probability decreases. Hence, assuming a fixed percentage of bad nodes in the network, each peer would like to increase the size of its ring thereby reducing the threat. However, this has an adverse effect on the cost. Including more peers means more cost both in terms of communication and synchronization. It seems that there exist a tradeoff and an optimal choice of this n_i^* is guided by the solution to the optimization problem, assuming a linear cost with respect to the size of the ring:

$$\max_n \left[w_{ti} \times threat(n) - w_{ci} \times cost(n) \right]$$

subject to the following constraints: $cost < c_i$ and $threat < t_i$ where $threat(n)$ is given by Equation 5 and $cost(n) = w_c \times g \times n$. g is the proportionality constant and c_i and t_i are constants for every peer and denote the cost threshold and privacy threshold that each peer is willing to withstand. Solution to this optimization problem gives us a range of values of n_i^*:

$$1 + k + \frac{\log(w_{ti}) - \log(t_i)}{\log(m+1)} \leq n_i^* \leq \frac{c_i}{w_{ci} \times g}. \tag{6}$$

Thus each peer chooses its own set of neighbors depending on the value of n_i^*. This naturally gives rise to asymmetric network topologies because two neighboring peers P_i and P_j may have different values of n_i^* and n_j^*. This naturally calls for a modification of the traditional secure sum protocol. The modified update rule is given by Equation 4.

Each node executes this privacy preserving sum computation only in its local neighborhood. For finding the distributed sum, whenever peer P_i wants to update its state (based on Equation 4 or 4), it does not get the raw $z_j^{(t-1)}$'s from all its neighbors, rather P_i forms an implicit ring among them. So every peer in the network has to execute two protocols: (i) protocol for it own ring formation and (ii) distributed averaging protocol in its own ring till asymptotic convergence of the sums. This approach has two advantages over existing algorithms: (i) it allows a peer to preserve the privacy while still correctly executing the distributed sum protocol and (ii) the ring being formed only in its local neighborhood, it does not suffer from the typical synchronization requirements. As a result, this distributed sum computation technique scales well to large networks.

6.1. Local Ring Formation Algorithm (L-Ring)

For distributed averaging, peer P_i updates its current state based on the information it gets from its n_i^* neighbors. In order to preserve privacy, P_i does not get the raw data from its neighbors; rather a ring is formed among n_i^* neighbors and sum computation is performed in that ring.

The ring formation algorithm works as follows. When initialized, each node solves the optimization problem and finds the value of n_i^*. It then launches n_i^* random walks in order to select n_i^* nodes uniformly from the network to participate in P_i's ring. The random walk we have used is the Metropolis-Hastings random walk which gives uniform samples even for skewed networks. We do not present the details here, interested readers are referred to (Das et al., 2008). Whenever one random walk ends at P_j, it first checks if $n_i^* < n_j^*$. If this is true, it poses a potential privacy breach for P_j. Hence P_j may choose not to participate in P_i's call by sending a **NAC** message along with its n_j^*. Otherwise P_j sends an **ACK** message to P_i. If P_i has received any **NAC** message, it computes $max(n_j^*)$ and checks if it violates its cost constraint. If the constraint is violated, P_i chooses a different peer P_q by launching a different random walk. Otherwise, it then sends out all of the $max(n_j^*)$ invitations again which satisfies the privacy constraints of all the participants. The pseudocode is presented in Algorithm 6.1.

Algorithm 6.1. Ring Formation for Privacy Preserving Sum Computation (**L-Ring**).

Input of peer P_i:
Threat t_i and cost c_i that peer P_i is willing to tolerate
Initialization:
Find the optimal value of n_i^* using t_i and c_i.
If P_i initializes a ring:
Contact the neighbors as dictated by n_i^* by launching n_i^* parallel random walks
When a random walk ends in node P_j:
Fetch the value of n_i^* as sent by P_i

IF ($n_i^* <= n_j^*$) Send (**NAC**,n_j^*) to P_i
ELSE Send **ACK** to P_i
ENDIF
On receiving NAC, n_j^* **from** P_j**:**
IF replies received from everyone
IF n_j^* violates cost constraint
Contact different neighbor P_q
ELSE $max = argmax_j\{n_j^*\}$; Set $n_i^* = max$
Send invitation $I(n_i^*)$ to P_j with n_i^* value
ENDIF
ENDIF

6.2. Privacy Preserving Algorithm for Feature Selection (PAFS)

The *PAFS* algorithm using misclassification gain metric works by invoking $m_1 + \cdots + m_p$ different privacy preserving distributed sum protocols for p features. For feature $\mathcal{A}_\ell$, peer P_i initializes m_ℓ estimates at time 0: $z_{i,\ell 0}^{(0)} = \left(x_{\ell,00}^{(i)} - x_{\ell,01}^{(i)}\right)$, $\ldots$, $z_{i,\ell(m_\ell-1)}^{(0)} = \left(x_{\ell,(m_\ell-1)0}^{(i)} - x_{\ell,(m_\ell-1)1}^{(i)}\right)$, where $z_{i,\ell a}^{(t)}$ denotes the estimate of peer P_i at time t when feature $\mathcal{A}_\ell$ takes on a value of a. This is done for all the features $\mathcal{A}_1, \ldots, \mathcal{A}_p$. Now each peer launches $m_1 + \cdots + m_p$ different distributed averaging computations in their local rings. Other than the initiator, whenever a peer gets data from its neighbor, it adds its data and sends it to the next one in the ring following the secure sum protocol. When the entire sum (masked by the random number) comes back to the initiator, the latter updates its estimate using Equation 4. It then sends the data again to the first member of the ring and the process continues. Once the sums converge (say at time t), each peer does the following computation with the local z's (following Equation 1):

$$s_1 = \sum_{a=0}^{m_1-1} \left| z_{i,1a}^{(t)} \right|, \ldots, s_p = \sum_{a=0}^{m_p-1} \left| z_{i,pa}^{(t)} \right|$$

The best features are the ones with the highest s_i's. Algorithm 6.2 presents the pseudo code.

In order to use gini index and entropy the following modifications are made:

- Instead of invoking m_ℓ number of distributed sum for each feature $\mathcal{A}_\ell$, we need to invoke $2 \times m_\ell$ number of private averaging computations. For any peer P_i and feature $\mathcal{A}_\ell = a$, initialize $z_{i,\ell a,0}^{(0)} = \left(x_{\ell,a0}^{(i)}\right)$, $z_{i,\ell a,1}^{(0)} = \left(x_{\ell,a1}^{(i)}\right)$. The third sum is simply the sum of the first two computations, *i.e.* $z_{i,\ell a,2}^{(0)} = \left(x_{\ell,a0}^{(i)} + x_{\ell,a1}^{(i)}\right)$.
- Once the sums converge at time t each peer computes the following quantities with its local estimates only,
  - **Gini Index**: $s_1 = \sum_{a=0}^{m_1-1} \left\{ \frac{(z_{i,1a,0}^{(t)})^2 + (z_{i,1a,1}^{(t)})^2}{z_{i,1a,2}^{(t)}} \right\}, \ldots,$
    $s_p = \sum_{a=0}^{m_p-1} \left\{ \frac{(z_{i,pa,0}^{(t)})^2 + (z_{i,pa,1}^{(t)})^2}{z_{i,pa,2}^{(t)}} \right\}$

· **Entropy**: $s_1 = \sum_{a=0}^{m_1-1} \left\{ z_{i,1a,0}^{(t)} \log \frac{z_{i,1a,0}^{(t)}}{z_{i,1a,2}^{(t)}} + z_{i,1a,1}^{(t)} \log \frac{z_{i,1a,1}^{(t)}}{z_{i,1a,2}^{(t)}} \right\}, \ldots,$
$s_p = \sum_{a=0}^{m_p-1} \left\{ z_{i,pa,0}^{(t)} \log \frac{z_{i,pa,0}^{(t)}}{z_{i,pa,2}^{(t)}} + z_{i,pa,1}^{(t)} \log \frac{z_{i,pa,1}^{(t)}}{z_{i,pa,2}^{(t)}} \right\}$

– As before, the best features are the ones with the highest values of $s_1, \ldots s_p$.

We avoid presenting the pseudo-code for Gini and entropy based techniques here due to their similarity with the misclassification gain based algorithm.

Algorithm 6.2. Privacy Preserving Algorithm for Feature Selection (PAFS).

Input of peer P_i:
Convergence rate ρ, local data D_i, *round*, and set of n_i^*-local neighbors arranged in a ring or $\{ring_{i,n^*}\}$
Initialization:
Initialize $\{ring_{i,n^*}\}$, ρ
Set $round \leftarrow 1$
Set $j \leftarrow$ first entry of $\{ring_{i,n^*}\}$
Compute:

–For feature $\mathcal{A}_1$: $z_{i,10}^{(0)} = \left(x_{1,00}^{(i)} - x_{1,01}^{(i)} \right), \ldots,$
$z_{i,1(m_1-1)}^{(0)} = \left(x_{1,(m_1-1)0}^{(i)} - x_{1,(m_1-1)1}^{(i)} \right),$
–For feature $\mathcal{A}_2$: $z_{i,20}^{(0)} = \left(x_{2,00}^{(i)} - x_{2,01}^{(i)} \right), \ldots,$
$z_{i,2(m_2-1)}^{(0)} = \left(x_{2,(m_2-1)0}^{(i)} - x_{2,(m_2-1)1}^{(i)} \right),$

$\vdots$

–
–For feature $\mathcal{A}_p$: $z_{i,p0}^{(0)} = \left(x_{p,00}^{(i)} - x_{p,01}^{(i)} \right), \ldots,$
$z_{i,p(m_p-1)}^{(0)} = \left(x_{p,(m_p-1)0}^{(i)} - x_{p,(m_p-1)1}^{(i)} \right),$

$\{ring_{i,n^*}\} \leftarrow \{ring_{i,n^*}\} \setminus j$
Send $\left(\left\{ z_{i,10}^{(0)}, \ldots, z_{i,p(m_p-1)}^{(0)} \right\}, \{ring_{i,n^*}\}, round \right)$ to j
On receiving a message $(\{y_1, \ldots, y_{m_1+m_2+\cdots+m_p}\}, \{ring\}, rnd)$ **from** P_j**:**
IF $\{ring\} = \emptyset$
Update $\{z_{i,10}^{(round)}, \ldots, z_{i,p(m_p-1)}^{(round)}\}$
$round \leftarrow round + 1$
Set $j \leftarrow$ first entry of $\{ring_{i,n^*}\}$
$\{ring_{i,n^*}\} \leftarrow \{ring_{i,n^*}\} \setminus j$
Send $\left(\{z_{i,10}^{(round)}, \ldots, z_{i,p(m_p-1)}^{(round)}\}, \{ring_{i,n^*}\}\ round \right)$ to j
Check if any node is waiting on this peer
Send data to all such nodes
ELSE
IF $round < rnd$
Wait
ELSE
Set $ret_1 = y_1 + z_{i,10}^{(rnd)}, \ldots, ret_{m_1+\cdots+m_p} = y_{m_1+\cdots+m_p} + z_{i,p(m_p-1)}^{(rnd)}$

Set $j \leftarrow$ first entry of $\{ring\}$
$\{ring\} \leftarrow \{ring\} \setminus j$
Send $\big(\{ret_1, \ldots, ret_{m_1+\cdots+m_p}\}, ring, rnd\big)$ to P_j
END
END

7. Analysis

In this section we analyze the performance of *L-Ring* and *PAFS* algorithms.

7.1. L-Ring Running Time Analysis

Lemma 1. For any peer P_i, and all neighbors $P_j \in \Gamma_{i,1}$, the L-Ring algorithm has a running time of $O(\max(n_i^*, n_j^*))$, where n_i^* is the optimal value for node P_i and n_j^* is the value required by node P_j where P_i and P_j belong to the same ring for the sum computation.

Proof. P_i's ring formation can have the following two cases:

1. For all $P_j \in \Gamma_{i,1}$, if $n_i^* > n_j^*$, then the running time is upper bounded by the maximum time required by P_i to contact all its neighbors *i.e.* $O(n_i^*)$.
2. Without loss of generality, assume that

   $$\Xi = \{P_1, \ldots, P_{S_i}\} \subseteq \Gamma_{i,1}$$

   be the set of nodes whose n_j^*, for all $P_j \in \Xi$ is greater than n_i^* *i.e.* $\forall P_j \in \Xi, n_j^* \geq n_i^*$. These are the number of **NAC** messages received by P_i from all $P_j \in \Gamma_{i,1}$. Computing the maximum of all entries in Ξ takes $O(|\Xi|)$. In order to accommodate all the nodes in its neighborhood, P_i increases its ring size to $\max_{P_j \in \Xi}\{n_j^*\}$. In this case, computation on this ring takes time $O(n_j^*)$.

Therefore the overall running time is $O(\max(n_i^*, n_j^*))$. □

In the *L-Ring* algorithm, each node contacts other nodes to form rings. For every such ring, there is one ring leader. Every such ring leader is also contacted by other nodes in the network to participate in their rings. Below we first state what is meant by deadlock in this system and then prove that our algorithm is deadlock-free.

Definition 7.1 (Deadlock). A deadlock is a situation wherein two or more competing actions are waiting for the other to finish, and thus neither ever does.

In our context, a deadlock can occur, for example if a node P_i has sent invitations to other nodes P_j to join its ring. These P_j's may themselves be waiting on others to send them response to their requests for forming rings. This process may be translated to all the nodes in the system and therefore, the entire system may become unresponsive. Below we prove that such a situation never arises in this context.

Lemma 2. The ring formation algorithm is deadlock-free.

Proof. Consider a node P_i whose n_i^* is the maximum of all the nodes in the network. Let us also assume that $n_i^* < d$, where d is the total number of nodes in

the network. In other words, the maximum n_i^* is such that a ring of size n_i^* can be formed in a network of size d. Consider any node P_j who sends a ring formation request message to P_i. Now by assumption, $n_j^* < n_i^*$, for all $P_j \neq P_i$. Also since, $n_i^* < d$, $n_j^* < d$ as well for all P_j. Thus it is evident that if P_i converges for ring formation so would all P_j's. Hence there can be no deadlock. In the worst case, multiple large rings will be formed which will include all the nodes in the network. Since there is no deadlock for P_i, there can be no deadlock for any of the neighbors of P_i. Thus, by simple induction on the entire network, *L-Ring* is deadlock free. □

7.2. PAFS Correctness Analysis

Our next lemma states that *PAFS* algorithm asymptotically converges to the correct ordering of the features.

Lemma 3. *PAFS* converges to the correct ordering of the feature for any impurity metric chosen above.

The proof is based on the convergence of the original distributed averaging algorithm proposed in (Scherber and Papadopoulos, 2005). In the modified version catered for asymmetric topologies, the only modification we have made is a change in the topology matrix Φ. It can be shown that this does not change the correctness or convergence criterion. We have omitted this due to similarity of proofs presented in (Scherber and Papadopoulos, 2005). It has been shown that the modified sum computation algorithm converges exponentially fast to the correct result. At each step in the computation, the error (difference between the local peer's estimate and the global sum) deceases exponentially. Since the last step in *PAFS* is ordering the features locally at each peer, it does not require any distributed computation and hence can be ignored for the convergence time analysis.

7.3. PAFS Locality Analysis

In this section we prove that *PAFS* is local. Intuitively, locality of an algorithm ensures bounded message complexity in each peer's neighborhood and hence is crucial for the algorithm's scalability. This has also been shown elsewhere (Datta, Bhaduri, Giannella, Wolff and Kargupta, 2006)(Das et al., 2008).

There are several definitions of locality proposed in the literature. The locality concept proposed by Das *et al.* (Das et al., 2008) is characterized by two quantities — (1) α – which is the number of neighbors a peer contacts in order to find answer to a query and (2) γ – which is the total size of the response which a peer receives as the answer to all the queries executed throughout the lifetime of the algorithm. The *PAFS* algorithm exhibit (α, γ)-locality in the following sense. For any given peer, the choice of α is guided by the optimal solution of the objective function defined earlier. In the worst case, a peer may choose α to be equal to the size of the entire network. Therefore, $\alpha = O(d)$ in the worst case. The bound on γ is specified by the following lemmas.

Lemma 4. Let ϵ be the error between the true sum (Δ) and the node estimates

$(\mathbf{z}_j^{(t)})$ as induced by PAFS algorithm after t rounds of computation. Then $t \geq \frac{log(\epsilon)-log(d)}{log(\lambda_{max}^2)}$, where λ_i's are the eigenvalues of the topology matrix Φ''.

Proof. From (Scherber and Papadopoulos, 2005), we know that the error at the t-th step is bounded by $d\lambda_{max}^{2t}$ *i.e.*

$$\left|\left|\mathbf{z}_j^{(t)} - 1\Delta_j\right|\right|^2 = \sum_{j=2}^{d} \lambda_i^{2t} \left|\mathbf{z}_j^{(0)}\right|^2 \text{ (assuming } \left|\mathbf{z}_j^{(0)}\right|^2 = 1)$$

Now let the error be bounded by ϵ. Thus,

$$d\lambda_{max}^{2t} < \epsilon \Rightarrow t \geq \frac{log(\epsilon) - \log(d)}{\log(\lambda_{max}^2)}$$

□

Theorem 4. The total size of the messages exchanged (γ) by any peer is upper bounded by

$$\frac{log(\epsilon) - \log(d)}{\log(\lambda_{max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right],$$

where $z_{max}^{(t)}$ is the maximum of data values at any peer in a ring at round t.

Proof. At round t, the number of bits necessary to store the maximum of all the $z_i^{(t)}$-s is $\log(z_{max}^{(t)})$. While performing the secure sum at any round ℓ, peer P_i with $\Gamma_{i,1} = \{P_{i-1}, P_{i+1}\}$ does the following computation: $(z_{i-1}^{(\ell)} + z_i^{(\ell)}) \mod N$, where N is the parameter of the sum computation protocol. Hence for every peer, the number of bits required to represent the new sum will increase by 1 at most. Therefore, the total number of bits required for each message is upper bounded by $\left[\log(z_{max}^{(\ell)}) + n_i^*\right]$. In each round of the sum computation, a peer exchanges only one message (due to ring topology). Hence, for t rounds, we get the total number of bits exchanged as $t\left[\log(z_{max}^{(t)}) + n_i^*\right]$. Using Lemma 4,

$$\gamma \leq \frac{log(\epsilon) - \log(d)}{log(\lambda_{\max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right].$$

□

Lemma 5. PAFS algorithm is $\left(O(d), \frac{\log(\epsilon)-\log(d)}{\log(\lambda_{\max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right]\right)$-local.

Proof. As stated, for any node P_i the maximum size of ring is equal to the size of the network. So according to the definition of locality, $\alpha = O(d)$. Also as shown in Theorem 4,

$$\gamma \leq \frac{\log(\epsilon) - \log(d)}{\log(\lambda_{\max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right]$$

. Therefore, *PAFS* algorithm is $\left(O(d), \frac{\log(\epsilon)-\log(d)}{\log(\lambda_{\max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right]\right)$-local. □

7.4. PAFS Privacy Analysis

Finally we prove that the *PAFS* algorithm is distributed ρ_1-to-ρ_2 privacy preserving.

Lemma 6. *PAFS* protocol is distributed ρ_1-to-ρ_2 privacy preserving.

Proof. First note that, for any node P_i there are two rings in which it participates. The ring for which P_i is the initiator satisfies the privacy model since the solution is found by solving the optimization problem. For any ring to which P_i is invited to, it only participates if $n_j^* > n_i^*$ which also guarantees conformity to the ρ_{1i}-to-ρ_{2i} model for P_i. Therefore, using the privacy model defined in 2, the *PAFS* protocol is distributed ρ_1-to-ρ_2 privacy preserving. □

7.5. PAFS Scalability Analysis

Next we analyze the scalability of the *PAFS* algorithm both with respect to quality of results and communication cost. *PAFS* asymptotically converges to the correct sum independent of the number of tuples or features. This because first local sums are computed based on a peer's data and then the averaging proceeds. So quality is not affected by either the number of features or tuples. For the communication cost, consider the following:

Variation with # features For p features, where feature $\mathcal{A}_i \in \{0, \ldots, m_i\}$, total number of distributed averaging computations initialized are $\alpha \sum_{i=1}^{p} m_i$ where $\alpha = 1$ for misclassification gain and $\alpha = 2$ for gini and entropy. Thus the communication complexity of *PAFS* is ($\sum_{i=1}^{p} m_i \times$ the time required for each distributed averaging to converge). As shown earlier, the time required for the distributed averaging to converge is bounded by logarithm of the number of nodes in the network.

Variation with # tuples There is no effect of the communication complexity on the number of tuples since each peer locally computes the counts based on all its local tuples and then uses these counts in the distributed averaging.

In contrast, it is worthwhile to mention that the naive solution of centralizing all the data for performing the same computation has a communication complexity of $O(\sum_{i=1}^{p} m_i) \times d$. This is because $\sum_{i=1}^{p} m_i$ local averages need to be centralized for all nodes d. However, the communication complexity of the centralized technique can be reduced assuming a tree overlay on the network and performing a convergecast-style computation. However, both these techniques have one major drawback — they require synchronization of the entire network.

There does not exist a lot of literature on distributed feature selection. Das *et al.* (Das et al., 2008) developed an algorithm for identifying the top features using a probabilistic random walk based algorithm. The communication cost of the probabilistic algorithm is expected to be lower than the algorithm proposed in this paper. However, the distributed algorithm developed in this paper is (1) eventually correct *i.e.* converges to the same result compared to a centralized algorithm, and (2) provides a guaranteed privacy protection.

8. Experimental Results

To validate the performance of the *PAFS* algorithm, we conducted experiments on a simulated network of peers. Our implementation of the algorithm was done in Java using the DDMT[2] toolkit developed at UMBC. For the topology, we used the BRITE topology generator[3]. We experimented with the *Waxman* model (Waxman, 1991) which first distributes a set of nodes uniformly in the plane, and then connects any two nodes (P_i and P_j) with a probability

$$Prob(i,j) = \alpha e^{-d(P_i,P_j)/\beta L},$$

where $0 < \alpha, \beta < 1$, $d(P_i, P_j)$ is the euclidean distance from node P_i to node P_j, and L is the maximum distance between any two nodes. As evident, the probability of connecting two nodes in inversely proportional to the distance between them and so nodes in the vicinity are heavily connected, thereby generating power-law topologies. Power-law random graph is often used in the literature to model large non-uniform network topologies. It is believed that P2P networks conform to such power law topologies (Saroiu, Gummadi and Gribble, 2002). α and β are constants which control the degree of connectivity for a fixed distance between nodes. For example, larger values of both α and β increase the interconnection probability, and therefore, tend produce fully connected graphs. Internet topologies generally exhibit heavily connected cluster of nodes instead. Hence, we have used small default values of $\alpha = 0.15$ and $\beta = 0.2$ in our experiments. We convert the edge delays in *Waxman* model to simulator ticks for time measurement since wall time is meaningless when simulating thousands of computers on a single PC.

We have experimented with two publicly available datasets at the UCI KDD archive[4]. The first is the mushroom dataset[5]. This dataset has been previously used for classification and prediction tasks. In our experiments, we have not used any semantics of the data; rather we have chosen this dataset because of the presence of categorical features with binary class labels. The full dataset has approximately 8000 tuples and 23 features. Of these features, 22 categorical features are used to describe the mushroom and the class feature is binary depicting if this is edible or not. We convert the nominal features to categorical (integer valued) by assigning an integer value to each possible symbol that the category can take. The maximum value of any categorical feature is 12. The second dataset that we have used is the forest cover dataset[6]. This dataset has 54 features — 44 binary and the rest categorical. It has a total of 581012 tuples. The last column is the class label which can take values between 1 to 7. Since our algorithm can only handle binary class labels, we create a one-vs.-all class distribution by re-assigning all tuples which have a class label of 2 (Lodgepole Pine) as 1 and the rest as 0. Our goal is to identify the set of features which are important for identifying the Lodgepole Pine forest type. Although this dataset is located at a single location, but many high-dimensional earth science data sets are distributed based on geographical locations. Once *PAFS* can identify

[2] http://www.umbc.edu/ddm/Sftware/DDMT/
[3] http://www.cs.bu.edu/brite/
[4] http://kdd.ics.uci.edu/
[5] http://archive.ics.uci.edu/ml/datasets/Mushroom
[6] http://kdd.ics.uci.edu/databases/covertype/covertype.html

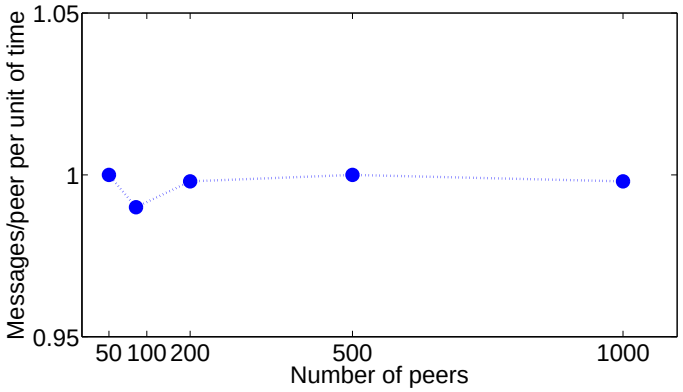


Fig. 2. Plot of the number of messages transferred vs. number of peers (misclassification gain).

the most important features in a distributed fashion, only the data corresponding to these features can be centralized to build a classifier. The cost of this two step process will be much less compared to centralizing the entire dataset with comparable accuracy.

In order to apply our distributed feature selection algorithm, the total number of tuples is equally split into non-overlapping blocks sequentially such that each block becomes the data of a peer.

In all our experiments we measure two quantities: the quality of our results and the cost incurred by our algorithm. We compare these quantities to the centralized execution of the same algorithms. Next we present the performance analysis of each of the variants of the *PAFS* algorithms on these two datasets.

8.1. Distributed Misclassification Gain

The *PAFS* algorithm is provably correct. In all our experiments of *PAFS* using misclassification gain, we have seen that it generates the same ordering of attributes when compared to the centralized algorithm.

Figure 2 shows the variation of the cost of the feature selection algorithm using misclassification gain when the number of nodes increases from 50 to 1000. The results are on the *mushroom data set*. As seen in Figure 2, the y-axis refers to the number of messages sent by each peer per unit of time. It varies between 0.16 and 0.167 which is a really low increase considering a 20-fold increase in the number of peers. As pointed out in Section 7, the total number of messages exchanged per round is $\sum_{i=1}^{i} m_i$. In this case, $\sum_{i=1}^{i} m_i = 60$. Assuming 4-bytes per integer, the size of a message per round is $60 \times 4 = 240$ bytes. Hence we claim that our algorithm shows excellent scalability in terms of the number of messages transferred.

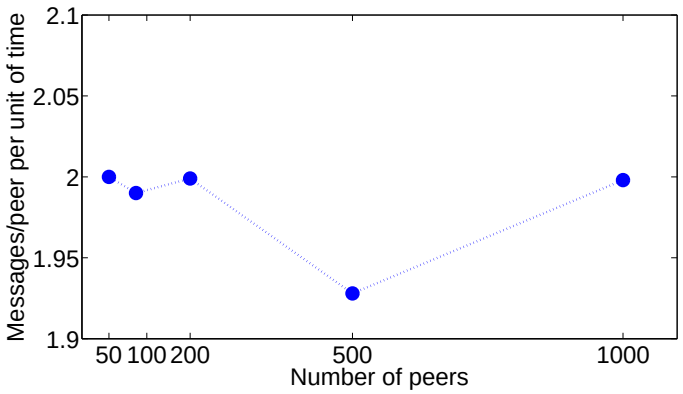


Fig. 3. Plot of the number of messages transferred vs. number of peers (gini index).

8.2. Distributed Gini Index

In our distributed experiment using the Gini measure on the same mushroom data set, the *PAFS* algorithm do not report the same ordering compared to centralized scenario. One pair of attributes are interchanged compared to the centralized ordering. This can be explained by the fact that for computing the gini index, we need to find the ratio of two sums. Since these sums are correct only asymptotically, there is always a small deviation from the true gini index. This can lead to error in the distributed algorithm.

The cost of the algorithm is shown in Figure 3. The number of messages vary between 1.97 and 2.0. Note that for Gini index, for each attribute and each possible value of an attribute, we need to execute 2 distributed sum protocols. For the same scenario, we need only 1 sum computation for misclassification gain. As a result, the number of messages per peer per unit of time doubles in this scenario. As before, the size of a message per round is $2\sum_{i=1}^{i} m_i \times 4 = 2 \times 60 \times 4 = 480$ bytes.

8.3. Distributed Entropy

In our last experiment with the mushroom data set, we test the entropy based distributed *PAFS* algorithm. The quality results are similar to the distributed Gini algorithm and can be attributed to the fact that in this case we need to compute the logarithm of sums. This introduces some error in the value and hence some features may be ordered differently compared to centralized execution. In our empirical analysis, we noticed three attributes mis-ordered by the distributed algorithm.

The number of messages per peer per unit of time varies between 1.98 and 2.0. In this case as well, the size of a message per round is $2\sum_{i=1}^{i} m_i \times 4 = 2 \times 60 \times 4 = 480$ bytes.

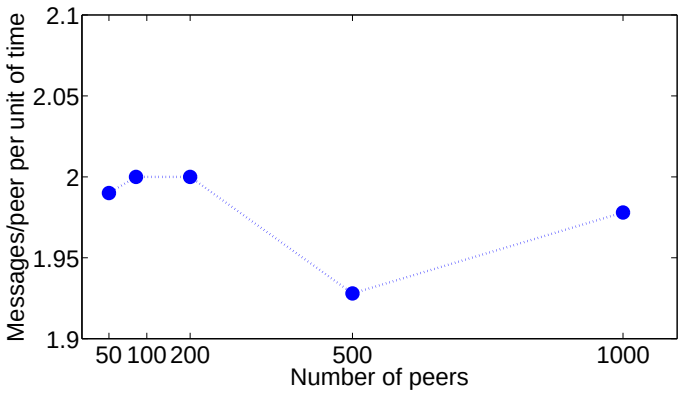


Fig. 4. Plot of the number of messages transferred vs. number of peers (entropy).

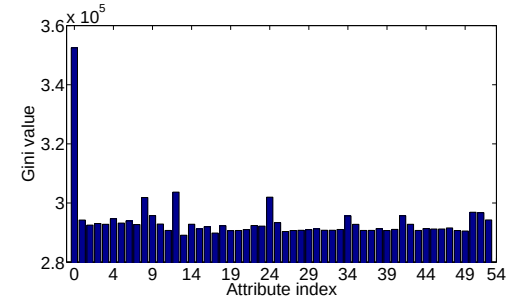
8.4. Experiments with Forest Cover dataset

In this set of experiments our focus is to identify the set of attributes which contribute highly towards classifying the Lodgepole Pine forest type. We have run all three variants of *PAFS*. Figure 5 shows the attributes along x-axis along with the measurement metric on the y-axis. Note that ordering of the attributes is not the same for all three measurements. In all these cases, we have run a centralized algorithm which produced the same results. We do not present any graphs on communication complexity because they are similar to what has been presented for the mushroom data set. In this case, $\sum_{i=1}^{54} m_i = 19746$. Thus, per round, *PAFS* exchanges $19746 * 4 = 78984$ bytes compared to 1974600 bytes needed for centralization.

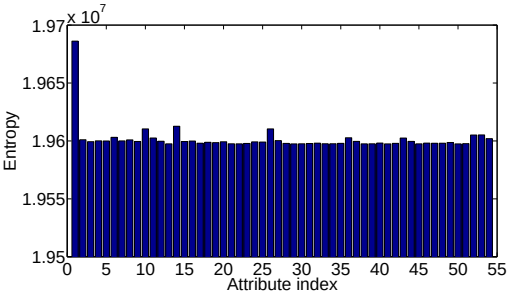
Therefore to sum up, all the proposed techniques demonstrate the superior quality of the algorithms at moderate cost of communication.

8.5. Discussion

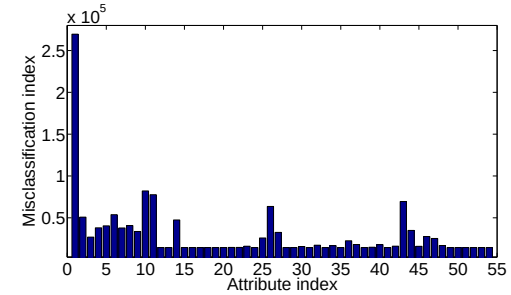
As shown in the above experimental results, *PAFS* converges to the correct result with a fraction of the cost necessary for centralization. Consider a scenario in which there are multiple users each having a number of songs. Each song has various features associated with it such as type, time, genre etc. Also based on its own preference each user has classified each song as *liked* or *not-liked*. Now, for a company which does song advertisement of new or yet unknown songs, it might be interested to send the advertisement to only the interested users. One way of achieving this is to run *PAFS* first to identify the top few features. Then a classifier can be built based on only these features by either centralizing the data associated with those features over all users or using a distributed classifier such as the one proposed in (Bhaduri et al., 2008). The classifier will essentially create a mapping from the users to songs based on the features selected by *PAFS*. Due to the unavailability of such real-life P2P user and song dataset, we could not run any such experiments.



(a) Gini index values vs. feature indices.



(b) Entropy measure vs. feature indices.



(c) Misclassification measure vs. feature indices.

Fig. 5. Relative values of the three feature selection measures for all the features of the forest cover dataset as found by *PAFS*.

9. Conclusions

In this paper we discuss the need for developing technology for taking personal content and delivering it to interested parties in a large population with diverse interests in a distributed, decentralized manner. We argue that the existing client-server models may not work very well in solving this problem because of scalability and privacy issues. We suggest that distributed and P2P data mining is likely to play a key role in many information and knowledge management tasks such as indexing, searching, and linking the data located at different nodes of a network in the future. Therefore, we also emphasize the need for developing local, asynchronous, distributed privacy-preserving data mining algorithms.

We have taken a very simple example problem and proposed a privacy preserving asynchronous algorithm for doing feature selection in a large P2P network. Distributed and privacy preserving versions of three popular feature selection techniques have been proposed. The algorithms are scalable, accurate and offers low communication overhead. Feature selection is a key step in matchmaking since it provides a compact representation of the huge volume of data otherwise available. This paper opens up a whole new genre of research in distributed privacy preserving data mining where the users are in control of both the privacy and the quality of the results.

Acknowledgements. We thank anonymous reviewers for their very useful comments and suggestions. This research is supported by the NASA Grant NNX07AV70G and the AFOSR MURI Grant 2008-11.

References

Bhaduri, K., Wolff, R., Giannella, C. and Kargupta, H. (2008). Distributed Decision Tree Induction in Peer-to-Peer Systems, *Statistical Analysis and Data Mining Journal* **1**(2): 85–103.

Chen, R., Sivakumar, K. and Kargupta, H. (2004). Collective Mining of Bayesian Networks from Distributed Heterogeneous Data, *Knowl. Inf. Syst.* **6**(2): 164–187.

Cho, V. and Wüthrich, B. (2002). Distributed Mining of Classification Rules, *Knowl. Inf. Syst.* **4**(1): 1–30.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. and Zhu, M. (2003). Tools for Privacy Preserving Distributed Data Mining, *ACM SIGKDD Explorations* **4**(2).

Das, K., Bhaduri, K. and Kargupta, H. (2009). A Distributed Asynchronous Local Algorithm using Multi-party Optimization based Privacy Preservation, *Proceedings of P2P'09*, Seattle, WA, pp. 212–221.

Das, K., Bhaduri, K., Liu, K. and Kargupta, H. (2008). Distributed Identification of Top-l Inner Product Elements and its Application in a Peer-to-Peer Network, *TKDE* **20**(4): 475–488.

Datta, S., Bhaduri, K., Giannella, C., Wolff, R. and Kargupta, H. (2006). Distributed Data Mining in Peer-to-Peer Networks, *IEEE Internet Computing* **10**(4): 18–26.

Datta, S., Giannella, C. and Kargupta, H. (2006). k-Means Clustering over a Large, Dynamic Network, *Proceedings of SDM'06*, MD, pp. 153–164.

Evfimevski, A., Gehrke, J. and Srikant, R. (2003). Limiting privacy breaches in privacy preserving data mining, *Proceedings of SIGMOD/PODS'03*, San Diego, CA.

Gilburd, B., Schuster, A. and Wolff, R. (2004). k-TTP: A New Privacy Model for Large-Scale Distributed Environments, *Proceedings of KDD'04*, Seattle, WA, pp. 563–568.

Jung, J. J. (2009). Consensus-based Evaluation Framework for Distributed Information Retrieval Systems, *Knowl. Inf. Syst.* **18**(2): 199–211.

Kargupta, H., Das, K. and Liu, K. (2007). Multi-Party, Privacy-Preserving Distributed Data Mining using a Game Theoretic Framework, *Proceedings of PKDD'07*, Warsaw, Poland, pp. 523–531.

Kargupta, H., Huang, W., Sivakumar, K. and Johnson, E. L. (2001). Distributed Clustering Using Collective Principal Component Analysis, *Knowl. Inf. Syst.* **3**(4): 422–448.

Kargupta, H. and Sivakumar, K. (2004). *Existential Pleasures of Distributed Data Mining. Data Mining: Next Generation Challenges and Future Directions*, AAAI/MIT press.

Keogh, E. J., Chakrabarti, K., Pazzani, M. J. and Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases, *Knowl. Inf. Syst.* **3**(3): 263–286.

Liu, H. and Motoda, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, London, GB.

Liu, K., Bhaduri, K., Das, K., Nguyen, P. and Kargupta, H. (2006). Client-side Web Mining for Community Formation in Peer-to-Peer Environments, *SIGKDD Explorations* **8**(2): 11–20.

Machanavajjhala, A., Gehrke, J., Kifer, D. and Venkitasubramaniam, M. (2006). l-diversity: Privacy beyond k-anonymity, *Proceedings of ICDE'06*, Atlanta, GA, p. 24.

Maulik, U., Bandyopadhyay, S. and Trinder, J. C. (2001). SAFE: An Efficient Feature Extraction Technique, *Knowl. Inf. Syst.* **3**(3): 374–387.

Saroiu, S., Gummadi, P. K. and Gribble, S. D. (2002). A measurement study of peer-to-peer file sharing systems, *Proceedings of Multimedia Computing and Networking (MMCN'02)*, San Jose, CA.

Sayal, M. and Scheuermann, P. (2001). Distributed Web Log Mining Using Maximal Large Itemsets, *Knowl. Inf. Syst.* **3**(4): 389–404.

Scherber, D. and Papadopoulos, H. (2005). Distributed Computation of Averages Over ad hoc Networks, *IEEE Journal on Selected Areas in Communications* **23**(4): 776–787.

Schuster, A., Wolff, R. and Trock, D. (2005). A High-performance Distributed Algorithm for Mining Association Rules, *Knowl. Inf. Syst.* **7**(4): 458–475.

Tan, P.-N., Steinbach, M. and Kumar, V. (2006). *Introduction to Data Mining*, Addison-Wesley.

Teng, Z. and Du, W. (2009). Hybrid Multi-group Approach for Privacy-preserving Data Mining, *Knowl. Inf. Syst.* **19**(2): 133–157.

Waxman, B. M. (1991). Routing of Multipoint Connections, pp. 347–352.

Wolff, R. and Schuster, A. (2004). Association Rule Mining in Peer-to-Peer Systems, *IEEE Transactions on Systems, Man and Cybernetics - Part B* **34**(6): 2426 – 2438.

Yang, Y. and Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization, *Proceedings of ICML-97*, Nashville, US, pp. 412–420.

Author Biographies

Kamalika Das received her B.Tech degree in CSE from Kalyani University, India, in 2003 and her MS and PhD degrees in CS from University of Maryland Baltimore County, in 2005 and 2009 respectively. Currently she is working as a postdoc with the Intelligent Data Understanding group at the NASA Ames Research Center. Kamalika has published several conference and journal papers in PKDD, SDM, P2P, IEEE Transactions and SIGKDD Explorations. Her work has been nominated for the 2007 PET award, and invited for fast-track submission to Peer-to-Peer Networking and Applications journal as one of the best papers of P2P 2009. She has received the 2008 Grace Hopper Celebration of Women in Computing Scholarship Award and the NSF-sponsored SIAM Student travel award for SDM 2009. Kamalika serves regularly as a reviewer for SDM, SIGKDD, ICDM conferences and TKDE, TIFS and SMC journals. More information about her can be found at http://www.csee.umbc.edu/~kdas1.

Kanishka Bhaduri is currently working as a research scientist in the Intelligent Data Understanding group at NASA Ames Research Center. He received his B.E. in Computer Science and Engineering from Jadavpur University India (2003) and PhD from the University of Maryland Baltimore County (2008). His research interests include distributed and P2P data mining, data stream mining, and statistical data mining. Several of his papers in top conferences have been selected as 'best' papers. Kanishka regularly serves as a reviewer and/or PC member for SDM, KDD, ICDM conferences and TKDE, SMC and DMKD journals. More information about him can be found at http://ti.arc.nasa.gov/profile/kbhaduri/.

Hillol Kargupta is a Professor at the Department of CSEE, University of Maryland Baltimore County. He received his Ph.D. in Computer Science from University of Illinois at Urbana-Champaign in 1996. He is also a co-founder of AGNIK LLC, a ubiquitous data intelligence company. His research interests include distributed data mining, data mining in ubiquitous environment, and privacy-preserving data mining. Dr. Kargupta won a US National Science Foundation CAREER award in 2001 for his research on ubiquitous and distributed data mining. He received the best paper award at the 2003 IEEE International Conference on Data Mining. He has published more than 90 peer-reviewed articles in journals, conferences, and books. He is an associate editor of the IEEE TKDE, the IEEE SMC Part B, and the SAM Journal. He regularly serves on the organizing and program committees of many data mining conferences. More information about him can be found at http://www.csee.umbc.edu/~hillol.

Correspondence and offprint requests to: Kamalika Das, Stinger Ghaffarian Technologies Inc., IDU Group, NASA Ames Research Center, Moffett Field, CA-94035, USA. Email: kdas1@cs.umbc.edu