Privacy-Preserving Naive Bayes Classification

Mengdi Huai[1,2](✉), Liusheng Huang[1,2], Wei Yang[1,2], Lu Li[1,2], and Mingyu Qi[1,2]

[1] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China
mdhuai@mail.ustc.edu.cn

[2] Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China

Abstract. In this paper, we propose differentially private protocols for Naive Bayes classification over distributed data. Compared with existing works, the privacy and security models in the proposed protocols are stronger: firstly, both the miner and parties can be arbitrarily malicious and can collude with each other to violate the remaining honest parties privacy; secondly, all communication channels between them can be assumed to be insecure. Specifically, we build a guarantee of differential privacy into the cryptographic construction so that the proposed protocols can tolerate collusions and resist eavesdropping attacks which are caused by insecure communication channels. Additionally, the proposed protocols can be implemented at lower computation and communication costs, and some extensions to our protocols (e.g. supporting parties dynamic joins or leaves) are also proposed in this paper. Both theoretical analysis and simulation results show that the proposed privacy-preserving protocols for Naive Bayes have strong security and better classification performance than the standard one.

Keywords: Distributed data mining · Naive Bayes · Differential privacy

1 Introduction

As a means of delivering valuable information, data mining has drawn more and more attention. The traditional data mining technology is based on the assumption that the miner can completely access to the data. But currently, it is common that data are distributed among various parties, who don't want to disclose their data due to privacy concerns. So the challenge here is how to accurately mine valuable knowledge from distributed data while effectively guaranteeing parties' privacy, especially when considering the miner or some parties are malicious.

In this paper, we focus on the distributed privacy-preserving of Naive Bayes (NB) learning, of which one of the most commonly used classifiers in the data-mining field. NB is based on statistic analysis and in distributed scenarios, the statistic queries over distributed parties are needed to obtain NB classifiers. However, the accurate query results always disclose private information of parties

S. Zhang et al. (Eds.): KSEM 2015, LNAI 9403, pp. 627–638, 2015.
DOI: 10.1007/978-3-319-25159-2_57

when deriving NB model parameters [1]. To protect the privacy of sensitive information, each party can add noises to their data so that the miner will derive noisy statistic results. Meanwhile, we should achieve two goals: providing useful results and preserving each party's privacy. Additionally, we can not ignore that some participating parties may be compromised and collude with the miner to infer others' sensitive information. Also, another point we need to note is insecure communication channels make their messages suffer from eavesdropping attacks.

To tackle the privacy concerns in distributed NB learning, various schemes have been proposed [4,6,10,12,14]. However, they either need to be implemented with high cost [4,6,10,14], or are easily subject to collusion attacks [4,12]. Also, the centralized private NB scheme [11] allows one party to add noises to data based on the standard differential privacy, achieving a good compromise between privacy and utility. However, their differential privacy mechanisms are not suitable for the distributed data-mining environment. This is because, too much accumulated error can be incurred from the distributed parties, and this will terribly affect the utility of NB classifiers learned from the distributed sceneries.

Given that, we propose the novel privacy-preserving NB protocols in this paper, and these protocols can deal with above situations with low computation and communication cost. In our protocols, we employ distributed differential privacy, a relaxation of differential privacy, which makes sense especially in distributed scenarios because it can provide rigorous privacy assurance and good utility. In our proposed protocols, distributed differential privacy not only lets the miner derive useful noisy results, but also guarantees the parties' data privacy even when some compromised parties collude with the untrusted miner by revealing their data and noises.

Although distributed differential privacy can preserve those honest parties' privacy, the magnitude of the noise generated by a specific party is not enough to ensure his data privacy since communication channels are insecure. So, we combine cryptography techniques with differential privacy to provide more secure guarantees, such that only negligible information of each party can be leaked even if the miner has arbitrary auxiliary information, which can be obtained in various ways (e.g. colluding with compromised parties). Specifically, each party firstly adds an appropriate noise to his data, and then encrypts the noisy data based on the encryption technology. At last, each party sends the encrypted noisy data to the miner who can decrypt the noisy results without learning anything else.

Additionally, parties' dynamic joins and leaves in distributed scenarios are well dealt with in our protocols. Another characteristic of our protocols is that they can support the incremental NB learning.

To sum up, our contributions in this paper are: firstly, we propose two distributed privacy-preserving NB protocols in the distributed environment where data is either horizontally or vertically partitioned, and the two protocols effectively resist both collusion and eavesdropping attacks; Secondly, compared with existing works, the proposed NB protocols can be implemented with lower computation and communication cost while ensuring only the miner having specific capability can get final NB classifiers. Thirdly, we extend them to make them more applicable in reality, such as supporting parties' dynamic joins or leaves.

2 Related Work

In the past, various privacy-preserving mechanisms for NB have been proposed. Using the rigorous privacy model of differential privacy, Vaidya et al. [11] construct a privacy-preserving NB classifier, in which the privacy model is that a data owner having centralized access to a dataset would like to release a NB classifier while preserving parties' data privacy. However, the centralized methods are not suitable to the distributed dataset.

Kantarcioglu et al. [4] in 2003 propose a private NB protocol for only horizontally partitioned data. One constrain in [4] is that they assume there are no collusion among all sites. Besides, their protocols transmit messages in a plain form, making those messages vulnerable to eavesdropping attacks. Yang et al. [14] in 2005 similarly propose a NB protocol over an horizontally distributed database, which is only suitable to a special scenario where each party just holds one instance. Using paillier cryptosystem, the authors in [12] also propose a horizontally private NB protocol, where the number of decryptions is at least $\min(\log_2 N_1, \log_2 N_2)/\log_2(n\mathcal{F})$ and there is no collusion between two parties.

The authors in [6] propose a private NB protocol over vertically distributed data streams using the secure multi-party computation, inevitably leading to high complexity. In [10], the authors present private NB protocols on both vertically and horizontally partitioned data, which cannot resist collusion attacks.

In 2005, Zhang et al. [13] combine data transform with data hiding to propose a new randomization method, to distort original data. Then, an effective NB classifier is presented to predict the class labels for unknown samples according to the distorted data. Yet, arbitrary randomization is not safe [5]. Compared with their methods, the differential privacy mechanisms used in our protocols can not only give rigorous mathematical proof but also provide good reconciliation between utility and privacy.

Motivated by those, we propose the novel privacy-preserving NB protocols. Firstly, the protocols need not any interaction among parties, which largely saves the system cost. Secondly, they can well deal with the scenarios where the miner colludes with parties and communication channels are insecure. Thirdly, they can be implemented with lower computation and communication cost.

3 System Model

3.1 System Setting

Here we consider the horizontally distributed scenarios, while leaving the vertically scenarios to section 5.4. We assume that there are a data miner, N samples and n parties and each party $\mathbf{P}_k$ $(k \in [n])$ has a local dataset.

Let $x = (a_1, ..., a_m)$ be a vector of observed random variables with no class label, where each feature a_i takes values from its domain A_i, i.e., $a_i \in A_i$. The set of all feature vectors is denoted as $\Omega = A_1 \times ... \times A_m$. The NB classifier will predict that x belongs to the class c_j (taken from $\{c_1, ..., c_r\}$) which has the highest posterior probability, conditioned on x. The NB classifier can be defined as follows: $NB(x) = argmax_{c_j} P(X = x|C = c_j)P(C = c_j) = argmax_{c_j} P(c_j) \prod_{i=1}^{m} P(a_i|c_j)$, where $P(c_j)$ represents the class prior probability and $P(a_i|c_j)$ represents the posterior probability.

By simply counting the frequency from the horizontally partitioned dataset, $p(a_i|c_j)$ is calculated as $p(a_i|c_j) = \frac{n_{ij}}{n_j} = \frac{\sum_{k=1}^{n} n_{ij}^k}{\sum_{k=1}^{n} n_j^k}$ and $p(c_j)$ is calculated as $p(c_j) = \frac{n_j}{N} = \frac{\sum_{k=1}^{n} n_j^k}{N}$, where n_j is the whole number of training samples whose class labels are c_j, and n_{ij} is the number of these training examples which also have a_i. And, n_j^k and n_{ij}^k denote the corresponding local counts. Get here, the goal we want to achieve here is how to let the miner get $\sum_{k=1}^{n} n_j^k$ and $\sum_{k=1}^{n} n_{ij}^k$ over n parties, while ensuring parties' privacy.

3.2 Attack Model

In our model, both parties and the miner can be arbitrarily malicious. We assume that the fraction of those uncompromised parties is γ, which can be estimated from priori knowledge, and the reminding compromised parties can collude with the miner to break honest parties' privacy. Besides, communication channels between them are assumed to be insecure, making parties' messages subject to eavesdropping attacks.

3.3 Designing Goals

1. **Utility and Privacy Guarantee.** In horizontally distributed environment, we should compute the sum statistic over n parties to get NB model parameters. Yet, accurate queries can violate parties' privacy [1]. So, each party can add an appropriate noise to his data to protect the data privacy. Meanwhile, two objectives should be compromised: preserving privacy and ensuring good utility, which can be achieved by differential privacy [1]. But, standard differential privacy let each party generate a noise to protect his privacy, making an $O(n)$ accumulated error (the difference with the accurate sum result). Instead, we resort to distributed differential privacy [9], allowing n parties collectively add only a geometric noise to each summation result.
Also, the distributed differential privacy used in our protocols is collusion-tolerant, which means the privacy of trusted parties can be well protected even if the compromised parties collude with the miner.
2. **Security Guarantee.** Note that using distributed differential privacy, the accumulated error in the sum statistics is only an copy geometric noise, but the magnitude of the noise incorporated to each party' data is not large enough to protect the data privacy. So, the security of data can be violated if only based on the distributed differential privacy since communication channels make parties' data suffer from eavesdropping attacks. We can improve this by incorporating cryptographic techniques. Given this, we design a (ϵ, δ)-differential privacy mechanism combining with the cryptographic construction to provide both security and privacy guarantee. The

main security goal in our NB protocols is to ensure that the miner learns only the final summation results, and nothing else about parties' private data.

3. **Additional Guarantee.** The computation and communication cost should be low since parties' resources are usually limited in reality. Also, distributed parties' dynamic leaves and joins need also to be considered. And, some other situations, such as data pollution should also be considered.

4 Basic Protocol Blocks

4.1 Protocol Sketch

Here, we give a high level description of the proposed horizontally differential private NB protocol (HDPNB). As previously mentioned, two techniques, i.e., differential privacy and applied cryptography methods, are used in HDPNB to provide privacy and security guarantee.

Firstly, each party independently adds an appropriate noise to his data using the data perturbation scheme mentioned below, then encrypts his noisy data using the corresponding encryption scheme and at last sends the encrypted noisy data to the miner who can decrypt summation statistics queries ($\sum_{k=1}^{n} n_{ij}^{k}$ and $\sum_{k=1}^{n} n_{j}^{k}$), and then privately get NB model parameters. Note that all parties collectively add a geometric noise (required for differential privacy) to every summation query count, i.e., $\sum_{k=1}^{n} n_{ij}^{k}$ and $\sum_{k=1}^{n} n_{j}^{k}$. The data perturbation and encryption schemes are present below.

4.2 The Naive Data Perturbation Scheme

The standard differential privacy, which can provide information-theoretic privacy guarantees that hold against computationally unbounded adversaries and balance the tradeoff between privacy protection and utility loss, allows each distributed party in our HDPNB to incorporate a Laplace noise into their local data, coursing $O(n)$ accumulated error [1].

Definition 1. *Differential Privacy. A randomized function K gives ϵ-differential privacy (ϵ is the privacy parameter) if for all neighborhood dataset D_1 and D_2 differing in at most one record, and all $S \in Range(K)$,*

$$Pr[K(D_1) \in S] \leq exp(\epsilon) * Pr[K(D_2) \in S] \tag{1}$$

So, for HDPNB, to get strong privacy protection and good utility, we instead use the distributed differential privacy [9] to let the parties be responsible for ensuring the differential privacy of their own data, and this incurs only $O(1)$ accumulated error. Based on distributed differential privacy, Shi et al. [9] propose a data perturbation scheme, where each party $\mathbf{P}_k$ ($k \in [n]$) generates an additive noise r_k ($r_k \in \mathbb{Z}_p$) following β-diluted geometric distribution to his data x_k ($x_k \in \mathbb{Z}_p$). As a consequence, roughly one copy of geometric noise $Geom(\alpha)$ is added to the original summation $\sum_{k=1}^{n} x_k$, which is the minimum amount of noise required to ensure ε-differential privacy [2]. And, they in [9] present below theorem 1 to show that the naive data perturbation scheme is computationally differentially private [8].

Definition 2. *Geometric Distribution. Let $\alpha > 1$. Geom(α) denotes the symmetric geometric distribution with parameter α. Its probability mass function at k $(k = 0, \pm 1, \pm 2, ...)$ is $\frac{\alpha-1}{\alpha+1} \cdot \alpha^{-|k|}$.*

Definition 3. *β-Diluted Geometric Distribution. Let $\alpha = exp(\frac{\varepsilon}{\Delta})$ $(\alpha > 1)$ and $\beta = min(\frac{1}{\gamma n} log \frac{1}{\delta}, 1)$ $(0 < \beta \leq 1)$. A random variable follows β-diluted Geometric distribution $Geom(\alpha)^{\beta}$ if it is sampled from Geom(α) with probability β, and is set to 0 with probability $(1 - \beta)$. ε and δ are privacy parameters, and Δ is sensitivity.*

Theorem 1. *Let $0 < \delta < 1$, $\varepsilon > 0$, $\alpha = exp(\frac{\varepsilon}{\Delta})$ and $\beta = min(\frac{1}{\gamma n} log \frac{1}{\delta}, 1)$, where γ is the fraction of honest parties. If each party adds a noise $Geom(\alpha)^{\beta}$, the above naive perturbation scheme achieves (ε, Δ)-distributed differential privacy.*

We here use this naive scheme to perturb both $\sum_{k=1}^{n} n_{ij}^{k}$ and $\sum_{k=1}^{n} n_{j}^{k}$ to protect parties' privacy, since accurate results always disclose privacy [1]. Yet, this naive scheme can not support parties' dynamic joins and leaves, which is solved next at only $O(1)$ error and low cost.

4.3 The Improved Data Perturbation Scheme

In the naive data perturbation scheme, each party utilizes the number of parties n to set parameter $\beta = min(\frac{1}{\gamma n} log \frac{1}{\delta}, 1)$ (parameters γ and δ are constant), such that all parties collectively add just one geometric noise to final results. But it requires large communication cost for each party join and leave since the exact value of n needs to be sent to the parties. Obviously, it conflicts with the lower communication cost goal.

Considering this, we give Alg.1, which relax the accuracy requirement on the value of n such that n does not have to be updated for every party' join and leave and only incurs low error and cost. Each party uses u rather than n to set parameter β. u is updated appropriately when some parties join or leave, but may not always reflect the real number of parties.

Theorem 2. *The average computation error of Alg.1 is roughly within twice of the geometric noise, required for differential privacy.*

Proof. Here, we first prove that $\forall k, u_k \in (\frac{n}{2}, n]$. Clearly, in the initial phase, we always have $\forall k, u_k \in (\frac{n}{2}, n]$. Suppose that all party's u_k initially are set as $\lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, ..., n, n$, i.e., n is even. After a party's join, the pattern becomes $\lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, \lfloor \frac{n}{2} \rfloor + 2, ..., n + 1, n + 1$ and the number of parties now becomes $n+1$ (odd), making that $\lfloor \frac{n}{2} \rfloor = \lfloor \frac{n+1}{2} \rfloor$. So, in this case, we have $\forall k, u_k \in (\frac{n}{2}, n]$. Similarly, when a party leaves, that $\forall k, u_k \in (\frac{n}{2}, n]$ also is

Algorithm 1. Procedures run by the trusted dealer to manage the values of u

```
Require: $n$ : the real number of party
Require: $u_k$ : the number of party that party $\mathbf{P}_k$ uses to set parameter $\beta$
 1: Initialization:
 2: if $n$ is even then
 3:     $u_1, u_2, ..., u_n \leftarrow \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, \lfloor \frac{n}{2} \rfloor + 2, ..., n, n$;
 4: else
 5:     $u_1, u_2, ..., u_n \leftarrow \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, \lfloor \frac{n}{2} \rfloor + 2, ..., n, n$;
 6: end if
 7:
 8: Join:
 9: if Party $\mathbf{P}_k$ joins then
10:     $n \leftarrow n + 1$;
11:     $u_k \leftarrow n$;
12:     Find a party $j$ with $u_j = \min\{u_1, u_2, ..., u_n\}$;
13:     $u_j \leftarrow n$;
14: end if
15:
16: Leave:
17: if Party $\mathbf{P}_k$ leaves then
18:     $n \leftarrow n - 1$;
19:     Find a party $\mathbf{P}_j$ with $u_j = \max\{u_1, u_2, ..., u_n\}$;
20:     if There exists another party $\mathbf{P}_m$ with $u_m = u_j$ then
21:         $u_m \leftarrow u_k$;
22:         $u_j \leftarrow \lfloor \frac{n}{2} \rfloor + 1$;
23:     end if
24: end if
```

true. In other two situations where a parties leaves or joins when n is odd, that $\forall k, u_k \in (\frac{n}{2}, n]$ also holds, of which the analysis is leaved to the full paper. In all those four cases, the condition that $\forall k, u_k \in (\frac{n}{2}, n]$ always holds. When $u \leq n$, at least one copy of geometric noise is added to ensure differential privacy; when $u > \frac{n}{2}$, at most one more copy of geometric noise is added. Here, this theorem is proved.

4.4 The Improved Encryption Scheme

The authors in [9] propose an aggregation encryption scheme, which doesn't require parties' interactions, keeps parties' secret keys ($H(t)^{sk_i}$ and H denotes a hash function) fresh, and can achieve strong security guarantees.

Yet, this naive encryption scheme [9] leaves one open problem. To compute the aggregation value $X = \sum_{k=1}^{n} x_k$ (x_k denotes parties' private data), the miner has to compute the discrete log, making their cryptographic construction not support large plaintext spaces. At small plaintext spaces, decryption can be achieved through a brute-force search. Even using Pollard's lambda method, decryption time is roughly square root in the plaintext space. Additionally, this scheme is not failure-tolerant and can't support parties' dynamic joins and leaves, both of which are solved in our methods.

Instead, we propose a computational efficient method, allowing the miner to directly and efficiently get the final results. Specifically, we use the modular property $(1+p)^m = 1 + mp \mod p^2$ to improve the encryption scheme in [9].

Based on the above equation, we get that $\prod_{k=1}^{n}(1+p)^{x_k} = \prod_{k=1}^{n}(1+p \cdot x_k) = (1 + p\sum_{k=1}^{n} x_k) \mod p^2$ $(x_k \in \mathbb{Z}_p)$. With the above property, the decryption time is only $O(1)$. The improved scheme, used in HDPNB, has three steps:

- **Setup(n,λ):** This step, run by a trusted dealer, takes the number of parties n, and a security parameter λ as inputs. It outputs: $(params, sk_0, \{sk_k\}_{k\in[n]})$, where $params$ are system parameters. sk_0 is distributed to the miner and sk_k $(k \in [n])$ is a secret key distributed to the party $\mathbf{P}_k(k \in [n])$, such that $sk_0 + sk_1 + ... + sk_n = 0$. The parties will use their secret keys to encrypt their data, and the miner will use its sk_0 to decrypt the sum. The setup algorithm only need to be performed once during the whole learning procedure.
- **Encrypt(sk_k, x_k, t):** At time t, each $\mathbf{P}_k$ first calculates $(1 + x_k \cdot p) \mod p^2$. Then the party multiplies it by secret parameter $H(t)^{sk_k}$ to get: $\mathbf{C}_k = (1 + x_k \cdot p) \cdot H(t)^{sk_k} \mod p^2$. Then, he uploads the ciphertext $\mathbf{C}_k$ to the miner.
- **Decrypt($sk_0, \{\mathbf{C}_k\}_{k\in[n]}, t$):** After receiving $\{\mathbf{C}_k\}_{k\in[n]}$ from all parties, the miner calculates: $C = H(t)^{sk_0} \cdot \prod_{k=1}^{n} \mathbf{C}_k = H(t)^{sk_0} \cdot \prod_{k=1}^{n}(1+x_k \cdot p) \cdot H(t)^{sk_k} = (1 + p\sum_{k=1}^{n} x_k) \mod p^2$. Then, the miner only needs to calculate $(C-1)/p \mod p = \sum_{i=1}^{n} x_k \mod p$ to decrypt the summation $\sum_{i=1}^{n} x_k$ $(x_k \in \mathbb{Z}_p)$. Two different modular operations used here don't affect the decryption [3].

The decryption time in our proposed method is only $O(1)$, while that in the naive encryption scheme [9] is at least $O(\sqrt{n\Delta})$, only when the plaintext space is small. For the large plaintext space, the decryption time will be inconceivable.

Updating Secrets. In the above encryption scheme, when a party joins or leaves, all parties' encryption keys need to be updated, resulting high communication cost in a large system. We address this by employing the interleaved grouping technique, behind which the key idea is to divide the parties into interleaved groups, where each group shares some parties with other groups. Owe to the space restriction, its detailed introduction is omitted here.

5 The Horizontally Differential-private NB Protocol

5.1 Computation of Sensitivity

Before presenting the proposed horizontally privacy-preserving NB protocol, we firstly analyze the sensitivity Δ. Note that, each party perturbs private data by adding a noise variable which follows β-diluted Geometric distribution $Geom(\alpha = exp(\frac{\varepsilon}{\Delta}))^{\beta}$ (parameters ε and β are usually predetermined), where Δ is the sensitivity of sum with respect to one party's change. In other words, if a single participants changes his data, the sum changes by at most Δ. Obviously, in our summation situation, the sensitivity is set as $\Delta = max(N_1, N_2, ..., N_n)$. Remarkably, $\Delta = max(N_1, N_2, ..., N_n)$ can be privately and efficiently computed using the approaches proposed in [3], where N_k denotes the size of the local dataset owned by the party $\mathbf{P}_k$ ($k \in [n]$), such that $N = \sum_{k=1}^{n} N_k$.

5.2 Protocol Description

HDPNB works as follows:

- **Setup.** Similar to the improved encryption scheme, each party $\mathbf{P}_k$ obtains the private key sk_k ($sk_k \in \mathbb{Z}_p$), and the miner obtains the capability sk_0.
- **CountQuery.** In this phase, each party $\mathbf{P}_k$ locally computes n_{ij}^k and n_j^k.
- **DataPert.** To ensure their differential privacy, each party $\mathbf{P}_k$ adds an appropriate noise which is produced based on the naive data perturbation scheme to the original data before encrypting them. We use the notation $\hat{n}_{ij}^k$ and $\hat{n}_j^k$ to denote the noisy plaintext of each party $\mathbf{P}_k$. Note that, honest participants will follow this protocol, but compromised participants may not add noise or even reveal their noise to the miner. The naive data perturbation scheme ensures that the accumulated noise to n_{ij} and n_j added by honest parties is large enough to protect their privacy. In the end, each party $\mathbf{P}_k$ will derive his randomized data $\hat{n}_{ij}^k$ and $\hat{n}_j^k$ by the above additive noise.
- **DataEnc.** Using the improved encryption scheme, each party $\mathbf{P}_k$ respectively encrypts the randomized data, i.e., $\hat{n}_{ij}^k$ and $\hat{n}_j^k$. Here, we use $\bar{n}_{ij}^k$ and $\bar{n}_j^k$ to represent the ciphertexts of $\hat{n}_{ij}^k$ and $\hat{n}_j^k$ respectively.
- **ResultDec.** As soon as receiving those encrypted ciphertexts $(\bar{n}_{ij}^1, \bar{n}_{ij}^2, ..., \bar{n}_{ij}^n)$ and $(\bar{n}_j^1, \bar{n}_j^2, ..., \bar{n}_j^n)$ from all parties, the miner then can obtain the summation plaintexts ($\hat{n}_{ij} = \sum_{k=1}^{n} \hat{n}_{ij}^k$ and $\hat{n}_j = \sum_{k=1}^{n} \hat{n}_j^k$) by simply summing up these ciphertexts. That is to say, through the decryption algorithm in the improved encryption scheme, the miner can obtain the noisy statistic $\hat{n}_{ij} = \sum_{k=1}^{n} \hat{n}_{ij}^k = \mathrm{sum}(\bar{n}_{ij}^1, \bar{n}_{ij}^2, ..., \bar{n}_{ij}^n)$ and $\hat{n}_j = \sum_{k=1}^{n} \hat{n}_j^k = \mathrm{sum}(\bar{n}_j^1, \bar{n}_j^2, ..., \bar{n}_j^n)$. Finally, the miner calculates $\hat{p}_{ij} = \hat{n}_{ij}/\hat{n}_j$. The probability p_j can also be calculated as $\hat{p}_j = \hat{n}_j/N$.

Note that, n distributed parties collectively add one copy of geometric noise $Geom(\alpha)$ to $\sum_{k=1}^{n} \hat{n}_{ij}^k$, i.e., $\sum_{k=1}^{n} \hat{n}_{ij}^k = \sum_{k=1}^{n} n_{ij}^k + r$ (r is a geometric noise). The same is to $\sum_{k=1}^{n} \hat{n}_j^k$.

5.3 Protocol Privacy and Security Analysis

The theorem 2 indicates that even if the compromised parties collude with the miner, the noise added by honest parties is large enough to ensure their differential privacy, i.e., achieving (ε, Δ)-distributed differential privacy, which shows that HDPNB is collusion-tolerant. The security analysis in [9] shows that the encryption scheme are secure enough under insecure communication channels to resist polynomial-time adversaries. To sum up, HDPNB provides differential privacy guarantee to resist polynomial-time adversaries, as our encryption schemes is secure enough against polynomial-time adversaries.

5.4 Extension

Vertically Differential-Private NB. Here, we address vertically differential private NB classification (VDPNB). Since vertically partitioned data must have a key attribute at all parties, we assume that each record owned by each party includes its class attribute value. We suppose the party $\mathbf{P}_k$ ($k \in [n]$) just have the k-th attributes A_k. In this way, there are n attribute in total. A sample x with n possible attributes values $a_1, ..., a_n$ is classified as: $NB(x) = argmax_{c_j} \frac{n_j}{N} \prod_{i=1}^{n} \frac{n_{ij}}{n_j}$, where N is the number of training samples, n_j ($n_j, n_{ij} \in \mathbb{Z}_p$) is the total number of training examples whose class label is c_j and n_{ij} is the number of those training examples that also have a_i. And, N and n_j are publicly known. So, the goal is to privately obtain the product $\prod_{i=1}^{n} n_{ij}$ over those n parties, where n_{ij} just need to be estimated from one party's local data. To ensure differential privacy, each party can himself independently generate an additive Laplace noise [1] to his n_{ij} (the sensitivity Δ is set as 1). To give more strong security guarantee, we also allow parties to firstly perturb their data, then encrypt his noisy data and lastly send encrypted ciphertexts to the miner. The encryption scheme used here is an variant of the naive encryption scheme used in HDPNB, where each party sends the ciphertext $n_{ij} \cdot H(t)^{sk_i}$ rather than $(1 + n_{ij} \cdot p) \cdot H(t)^{sk_i}$ to the miner, which can directly decrypt the product $\prod_{i=1}^{n} n_{ij}$.

Dynamic Joins or Leaves. In distributed environment, some parties, who agree to perform the above NB protocols, may dynamically leave, and new parties may join, which we address both problems in the improved data perturbation scheme, i.e., Alg.1.

Others. In reality, some other challenges, including data pollution, fault tolerance, malicious modification and the incremental NB learning, also need to be considered, which we don't introduce due to the limited space.

6 Performance Analysis

6.1 Complexity Analysis

Here, we discuss the computation and communication complexities in our protocols. In the horizontal NB protocol, the communications just exist in steps **DataEnc** and **ResultDec**, where each party sends his encrypted ciphertext c to the miner. Therefore, the total number of bits transferred by each party will be $O(|c|)$, where $|c|$ represents the total bit length of c. The dominant computation cost in the step **CountQuery** is $O(l)$ (l is the number of samples owned by every party), while the computation cost in other steps is only $O(1)$.

Similarly, the computation and communication cost for each party is $O(l)$ and $O(|c|)$ respectively, where l is number of whole samples and $|c|$ is the bit length of transferred ciphertext c in each party side.

The communication cost in Alg.1 is very small: when a party leaves, at most two remaining parties with the maximum u are updated; when a party joins, the joining one and another one with the minimum u are updated.

6.2 Experimental Evaluations

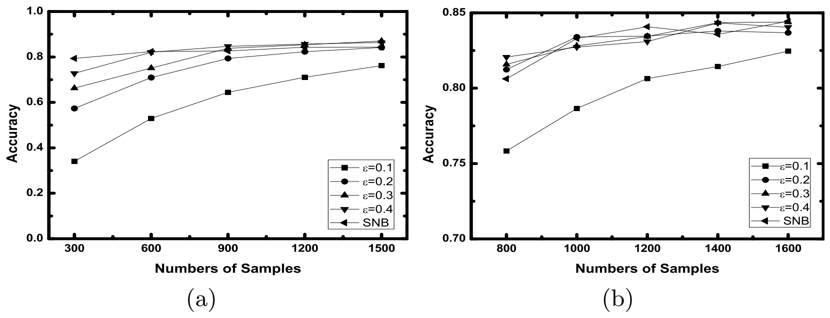

Fig. 1. Classification accuracy comparison

In this section, we respectively compare the classification performance between: standard NB (SNB for short) and HDPNB, SNB and VDPNB. Fig.1 shows simulation results on the dataset of Car Evaluation [7]. Fig.1 (a) gives the classification accuracy comparison between SNB and HDPNB, while Fig.1 (b) between SNB and VDPNB. For every subfigure in Fig.1, we vary the number of parties n, and compare their practical utility (i.e., classification accuracy) under fixed privacy parameters ($\epsilon = 0.1, 0.2, 0.3, 0.4$ respectively). For each n, we average and record the ten-fold cross-validation accuracy over 2000 runs, since it is a randomized algorithm. Specifically, for the proposed HDHNB, we assume that each party just has one sample, and γ is set to be 1 in Fig.1 (a) (assuming no compromised parties). The simulation results show that both HDPNB and VDPNB have comparable or even better classification performance when compared with SNB, especially when the number of samples increases. In addition, from Fig.1, we can clearly see that the larger ε is, the better classification performance the two proposed HDPNB and VDPNB have. Thus, the proposed privacy-preserving NB protocols are practical in reality.

7 Conclusion

This paper presents privacy-preserving protocols for learning NB classifiers over both horizontally and vertically distributed data. The proposed protocols guarantee the privacy of sensitive information even a subset of malicious parties collude with the untrusted miner, and still ensure the reminding honest parties' differential privacy while guaranteeing good NB classification performance, and they are also secure enough under insecure communication channels, while at low cost. Additionally, we also make some extension to it. The experimental results show that our protocols are effective to be applicable in practice.

Acknowledgments. This work is partially supported by the National Natural Science Foundation of China (Nos. U1301256, 61170058, 61202028, 61303206, 61202407).

References

1. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
2. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. SIAM Journal on Computing (2012)
3. Jung, T., Li, X.-Y.: Collusion-tolerable privacy-preserving sum and product calculation without secure channel (2014)
4. Kantarcioglu, M., Vaidya, J., Clifton, C.: Privacy preserving naive bayes classifier for horizontally partitioned data. In: IEEE ICDM Workshop on Privacy Preserving Data Mining, pp. 3–9 (2003)
5. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. pp. 99–106. IEEE (2003)
6. Keshavamurthy, B.N., Toshniwal, D.: Privacy-preserving Naïve Bayes classification using trusted third party computation over vertically partitioned distributed progressive sequential data streams. In: Nagamalai, D., Kaushik, B.K., Meghanathan, N. (eds.) CCSIT 2011 Part II. CCIS, vol. 132, pp. 444–452. Springer, Heidelberg (2011)
7. Lichman, M.: UCI machine learning repository (2013)
8. Mironov, I., Pandey, O., Reingold, O., Vadhan, S.: Computational differential privacy. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 126–142. Springer, Heidelberg (2009)
9. Shi, E., Chan, T-.H.H., Rieffel, E.G., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS (2011)
10. Vaidya, J., Kantarcioglu, M., Clifton, C.: Privacy-preserving naive bayes classification. VLDB **17**(4), 879–898 (2008)
11. Vaidya, J., Shafiq, B., Basu, A., Hong, Y.: Differentially private naive bayes classification. In: IEEE/WIC/ACM on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 1, pp. 571–576. IEEE (2013)
12. Yi, X., Zhang, Y.: Privacy-preserving naive bayes classification on distributed data via semi-trusted mixers. Information Systems **34**(3), 371–380 (2009)
13. Zhang, P., Tong, Y., Tang, S., Yang, D.: Privacy preserving naive bayes classification. In: Li, X., Wang, S., Dong, Z.Y. (eds.) ADMA 2005. LNCS (LNAI), vol. 3584, pp. 744–752. Springer, Heidelberg (2005)
14. Yang, Z., Zhong, S., Wright, R.N.: Privacy-preserving classification of customer data without loss of accuracy. In: SDM. SIAM (2005)