A Privacy-Preserving Framework for Gaussian Mixture Models

Madhusudana Shashanka
shashanka@alum.bu.edu

***Abstract*—This paper presents a framework for privacy-preserving Gaussian Mixture Model computations. Specifically, we consider a scenario where a central service wants to learn the parameters of a Guassian Mixture Model from private data distributed among multiple parties with privacy constraints. In addition, the service also has security contraints where none of the data owners are allowed to learn the values of the trained parameters. We use Secure Multiparty Computations to propose a framework that allows such computations. In addition, we also show how such a central service can classify new test data from privacy constrained third parties without exposing the learned models. The classification occurs with the added constraint that the service learns no information about either the test data or the result of the classification.**

***Keywords*-Secure Multiparty Computation, Privacy Preserving Data Mining, Distributed Data Mining, Gaussian Mixture Models**

I. Introduction

Consider a scenario involving three parties - Alice, Bob and Charlie. Distributed among them is a dataset, the mining of which would result in insights that are of great value to a fourth person David. However, the dataset each person has is private i.e. none of the parties will share data with others. The problem for David is to set up a protocol whereby everybody can collaborate to compute quantities relevant to him but at the same time guarantees that private data is not shared with anybody (including David).

There has been a lot of work in the field of cryptography that enables privacy-preserving collaborative computations like the one described in the scenario above. This area of research is often referred to as *Secure Multiparty Computations*. The field originated from the work of Yao [1] on the millionaire problem: two millionaires want to find out who is richer but do not want to divulge specific numbers of their wealth. See [2] for a detailed discussion of the topic. In the last few years, researchers in data-mining and machine learning have utilized these developments in cryptography to propose new applications and algorithms for privacy-preserving datamining. Some examples include multiple parties performing k-means [3], computation of means and related statistics from distributed databases [4] and computer vision applications [5]–[7].

In this paper, we consider applications that use Gaussian Mixture Models (GMMs). We take inspiration from the work of Shashanka and Smaragdis [8]–[10] where they look at the problem of learning and applying Gaussian Mixture Models for classification and in the context of Hidden Markov Models for speech recognition on private datasets. We extend their work on GMM training which is constrained to the case where all the data is owned by a single party. We show how GMMs can be learned from private data distributed among multiple parties. We also present protocols to explicitly compute likelihoods from Gaussians and GMMs and utilize these to present simplified protocols for classification. The paper is organized as follows. We first present background about Secure Multiparty Computations in Section II and then describe Gaussian Mixture Models in Section III. We present protocols to compute the likelihood of a datapoint given a Gaussian and a GMM in Section IV. Section V presents privacy-preserving protocols for learning GMM parameters from distributed private datasets while Section VI presents privacy-preserving protocols for classifying a new datapoint given learned GMMs for a set of classes. Finally, we present conclusions and discuss avenues for future work in Section VII.

Author is currently with United Technologies Research Center (UTRC). This paper reports work done by the author prior to joining UTRC.

II. Secure Multiparty Computations

Let us consider a problem with two parties Alice and Bob who have private data vectors $\mathbf{a}$ and $\mathbf{b}$ respectively. They want to compute the result of a function $f(\mathbf{a}, \mathbf{b})$. Consider a trusted third-party who can take private data, calculate the result $\mathbf{c} = f(\mathbf{a}, \mathbf{b})$, and initimate the result to the parties. Any protocol that implements an algorithm to compute the result $f(\mathbf{a}, \mathbf{b})$ is said to be *secure* only if it leaks no more information about $\mathbf{a}$ and $\mathbf{b}$ than what one can learn from the result $\mathbf{c}$ from the third-party. We can implement an algorithm to compute $f(\mathbf{a}, \mathbf{b})$ in a secure fashion by following these steps:

- express the algorithm in terms of basic operations for which secure implementations are known (secure primitives),
- distribute intermediate results as random additive shares between two parties so that neither has access to the entire result.

If there are multiple parties involved in the computation, we follow the same approach by breaking the algorithm down into basic secure primitives and all the intermediate results are distributed among the parties involved as random additive shares. Here we assume that all parties are *semi-honest* where they follow the protocol but could be saving messages and intermediate results to learn more about others' private

Primitive and Inputs	Output: Alice	Output: Bob	Relation
$SIP(\mathbf{x},\mathbf{y})$	a	b	$a+b=\mathbf{x}^T\mathbf{y}$
$SMAX(\mathbf{x},\mathbf{y})$	j	j	$j=\text{argmax}_i(x_i+y_i)$
$SLOG(\mathbf{x},\mathbf{y})$	a	b	$a+b=\ln(\sum_{i=1}^{d} e^{x_i+y_i})$

Table I
SECURE PRIMTIVES UTILIZED IN THIS WORK. ALICE HAS PRIVATE VECTOR $\mathbf{x}$ AND BOB HAS PRIVATE VECTOR $\mathbf{y}$. THE ABOVE TABLE SHOWS THE COMPUTATIONS ACCOMPLISHED BY THESE PRIMITIVES AND THE DISTRIBUTED OUTPUTS THAT ARE GENERATED.

data. In other words, parties are *honest but curious* and will follow the agreed-upon protocol but will try to learn as much as possible from the data-flow between parties[1].

At the most basic level, cryptographic primitives exist that enable the evaluation of boolean gates and circuits. The values of input wires are held as additive shares by different parties and the desired result is additive shares of the output wire. A detailed treatment is out of scope of this paper and we refer interested readers to [2] for more details. In this paper, the primtives we use are computations such as scalar products and computing maximum values of distributed vectors. The tool we use for the purpose is a public-key homomorphic cryptosystem. The general idea of a homomorphic cryptosystem is that one can perform mathematical operations on encrypted data without the need for decryption. Paillier cryptosystem is an example of one such semantically secure[2] homomorphic encryption scheme [12], [13] where multiplication of the encryptions of two numbers results in the encryption of the addition of the original numbers. In other words, $\mathcal{E}(a)\times\mathcal{E}(b)=\mathcal{E}(a+b)$. Appendix A describes how this cryptosystem can be used to compute the scalar product of vectors held by two parties.

Below, we briefly describe the secure primitives that will be used in the rest of the paper. We follow the approach used in [10] and employ secure dot-product as the main primitive of our algorithms, along with computations of the index of the maximum and logsums. Let Alice have the vector $\mathbf{x}=[x_1\dots x_d]$ and Bob have the vector $\mathbf{y}=[y_1\dots y_d]$. a and b refer to additive shares obtained by Alice and Bob respectively.

- Secure Inner Product produces random additive shares - a and b - of the result of the inner product between $\mathbf{x}$ and $\mathbf{y}$. This is denoted as $a+b=SIP(\mathbf{x},\mathbf{y})$.
- Secure Maximum Index results in Alice (and/or Bob) receiving the index of the maximum element of $\mathbf{x}+\mathbf{y}$ but neither party will know the value of the maximum. This is denoted as $j=SMAX(\mathbf{x}+\mathbf{y})$.
- Secure Logsum results in random additive shares a and b such that $a+b$ is the logsum of the elements of $\mathbf{x}+\mathbf{y}$. In other words, if $\mathbf{x}+\mathbf{y}=\ln\mathbf{z}=[\ln z_1\dots\ln z_d]$, $a+b=\ln(\sum_{i=1}^{d} z_i)$. This is denoted as $a+b=SLOG(\mathbf{x},\mathbf{y})$.

Table I summarizes the above primitives and the operations they enable. Implementations of these primtives are described in the Appendices.

[1]An alternate model is where parties are *malicious* in which case enforcing security becomes harder. In that case, one can utilize protocols involving zero-knowledge proofs and conditional disclosure of secrets, see [11] for more details.

[2]Semantic security in simple terms implies that an adversary will be unable to distinguish whether a pair of ciphertexts encode the same message or two different messages.

III. BACKGROUND: GAUSSIAN MIXTURE MODELS

A Gaussian Mixture Model is a probability density model that is comprised of several component Gaussian distributions that combine with different weights to provide a multimodal density. The mixture model can be characterized by the number of component gaussians c, the mixing weights corresponding to each component, and parameters for each component Gaussian given by the mean and variance.

Gaussian Mixture Models are widely used in data mining and machine learning for applications such as clustering and classification. In this work, we consider the application of GMMs to classification. We divide the computations involving GMMs to three groups -

- computing the likelihood of a datapoint given a GMM,
- learning the parameters of a GMM from a given set of data points, and
- using the likelihoods of a new test data point from GMMs for each class for classification.

In this section, we formulate the computations in each of the above groups in a way that makes designing secure implementations easier.

A. Log-Likelihood Computations

Let us first consider the log-likelihood of a datapoint $\mathbf{x}$ for a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$. It is given by

$$\ln p(\mathbf{x}|\mathcal{N})=-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})-\frac{d}{2}\ln 2\pi-\frac{1}{2}\ln|\boldsymbol{\Sigma}|. \tag{1}$$

We can write the above equation in a simplified form as

$$\ln p(\mathbf{x}|\mathcal{N})=\mathbf{x}^T\bar{\mathbf{W}}\mathbf{x}+\bar{\mathbf{w}}^T\mathbf{x}+w \tag{2}$$

where

$$\bar{\mathbf{W}}=-\frac{1}{2}\boldsymbol{\Sigma}^{-1},\qquad \bar{\mathbf{w}}=\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu},\qquad \text{and}$$
$$w=-\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}-\frac{1}{2}\ln|\boldsymbol{\Sigma}|-\frac{d}{2}\ln 2\pi.$$

Let us create the $(d+1)$-dimensional vectors $\bar{\mathbf{x}}$ and $\mathbf{w}$ by appending the value 1 to $\mathbf{x}$ and appending w to $\bar{\mathbf{w}}$. By changing $\bar{\mathbf{W}}$ into a $(d+1)\times(d+1)$ matrix $\mathbf{W}$ where the first d components of the last row are zeros and the last column is equal to $\mathbf{w}^T$ as illustrated in Figure 1, we can express equation (2) in a simplified form as

$$\ln p(\mathbf{x}|\mathcal{N})=\bar{\mathbf{x}}^T\mathbf{W}\bar{\mathbf{x}}. \tag{3}$$

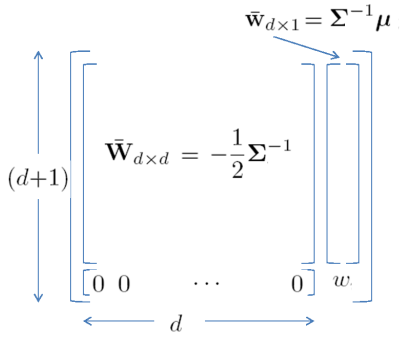

Figure 1. Schematic illustrating how parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be encoded in a $(d+1) \times (d+1)$ matrix $\mathbf{W}$ to simplify log-likelihood computations. The log-likelihood $\ln p(\mathbf{x}|\mathcal{N})$ can be expressed $\bar{\mathbf{x}}^T\mathbf{W}\bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is a $(d+1)$-vector obtained by appending 1 to $\mathbf{x}$. Above, $w = -(1/2)\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - (1/2)\ln|\boldsymbol{\Sigma}| - (d/2)\ln 2\pi$.

Now, consider a Gaussian Mixture Model density with J Gaussians

$$p(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j p(\mathbf{x}|\mathcal{N}_j), \tag{4}$$

where $\mathcal{N}_j$ is a Gaussian with mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$, and α_j are the mixture coefficients.

Let $l_j(\mathbf{x})$ represent the logarithm of a single term within the summation on the right-hand-side of the above equation. We have

$$l_j(\mathbf{x}) = \ln(\alpha_j p(\mathbf{x}|\mathcal{N}_j)) = \ln\alpha_j + \ln p(\mathbf{x}|\mathcal{N}_j). \tag{5}$$

We know from equation (3) that $\ln p(\mathbf{x}|\mathcal{N}_j)$ can be written as $\bar{\mathbf{x}}\mathbf{W}_j\bar{\mathbf{x}}$ where the components $\bar{\mathbf{W}}$, $\bar{\mathbf{w}}$ and w are constructed from parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ (refer to Figure 1). By adding $\ln\alpha_j$ to w, we can represent $l_j(\mathbf{x})$ as

$$l_j(\mathbf{x}) = \bar{\mathbf{x}}^T\mathbf{W}_j\bar{\mathbf{x}}. \tag{6}$$

Given $l_j(\mathbf{x})$ for all j, we can write the log-likelihood as

$$\begin{aligned} \ln p(\mathbf{x}) &= \ln\left(\sum_{j=1}^{J} e^{l_j(\mathbf{x})}\right) \\ &= \text{logsum}\big(l_1(\mathbf{x}), \ldots, l_J(\mathbf{x})\big). \end{aligned} \tag{7}$$

B. GMM Training

Let the training data correspond to K d-component vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$ and the problem is to learn a mixture of c Gaussians from this dataset. Let $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ represent the i-th Gaussian where $\boldsymbol{\mu}_i$ is the mean vector and $\boldsymbol{\Sigma}_i$ is the covariance matrix. Let $P(\mathcal{N}_i)$ represent the mixture weight for the i-th gaussian. The above parameters can be estimated iteratively using update equations derived from the Expectation-Maximization algorithm.

Let us denote the estimate of a parameter after the rth iteration by using a superscript and let λ^r denote the entire parameter set after the iteration. The steps of the EM algorithm are given by the following equations:

E Step:

$$P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r) = \frac{p(\mathbf{x}_k|\mathcal{N}_i)P^r(\mathcal{N}_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k|\mathcal{N}_j)P^r(\mathcal{N}_j)}, \tag{8}$$

M Step:

$$\begin{aligned} \boldsymbol{\mu}_i^{r+1} &= \frac{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)\mathbf{x}_k}{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)}, \\ P^{r+1}(\mathcal{N}_i) &= \frac{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)}{K}, \\ \boldsymbol{\Sigma}_i^{r+1} &= \frac{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)(\mathbf{x}_k - \boldsymbol{\mu}_i^r)(\mathbf{x}_k - \boldsymbol{\mu}_i^r)^T}{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)}. \end{aligned} \tag{9}$$

Iterating through the above equations is guaranteed to converge to a local optimum.

C. GMM Classification

Consider the problem where we have a data vector $\mathbf{x}$ that we wish to classify into N classes ω_i, $i = \{1, \ldots, N\}$ and each class is modeled as a Gaussian Mixture Model. The idea is to evaluate the value of the *discriminant function*

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i) \tag{10}$$

for all classes ω_i and assign $\mathbf{x}$ to class ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$. Here, $p(\mathbf{x}|\omega_i)$ is the class-conditional probability density function (class likelihood) and $P(\omega_i)$ is the *a priori* probability of class ω_i.

Let the mean vector and covariance matrix of the j-th Gaussian in class ω_i be $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$ respectively. The class-conditional probability distribution is then given by

$$p(\mathbf{x}|\omega_i) = \sum_{j=1}^{J_i} \alpha_{ij} p(\mathbf{x}|\mathcal{N}_{ij}), \tag{11}$$

where J_i is the number of Gaussians describing class ω_i and α_{ij} are the mixture coefficients. $\mathcal{N}_{ij}$ is shortened notation for $\mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij})$.

From Section III-A, we know that the log-likelihood can be written as (by equation (7))

$$\ln p(\mathbf{x}|\omega_i) = \ln\left(\sum_{j=1}^{J_i} e^{l_{ij}(\mathbf{x})}\right), \tag{12}$$

where

$$l_{ij}(\mathbf{x}) = \ln(\alpha_{ij} p(\mathbf{x}|\mathcal{N}_{ij})) = \bar{\mathbf{x}}^T\mathbf{W}_{ij}\bar{\mathbf{x}}. \tag{13}$$

The matrix $\mathbf{W}_{ij}$ is constructed using parameters α_{ij}, $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$ as explained in the derivation of equation (6) from equation (5).

The discriminant function for the i-th class can be written as

$$\begin{aligned} g_i(\mathbf{x}) &= \text{logsum}\big(l_{i1}(\mathbf{x}), \ldots, l_{iJ_i}(\mathbf{x})\big) + \ln P(\omega_i) \\ &= \ln\left(\sum_{j=1}^{J_i} e^{l_{ij}(\mathbf{x})}\right) + \ln P(\omega_i) \end{aligned} \quad (14)$$

and can be used for classification.

IV. Privacy Preserving GMM Likelihoods

Let Alice have a d-component vector $\mathbf{x}$. In this section, we show how Alice and David can compute the log-likelihood of $\mathbf{x}$ given a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ owned by David. We present a protocol for accomplishing this computation where Alice will not disclose any information about $\mathbf{x}$ to David and David does not disclose any information to Alice about the parameters of the Gaussian. We then present a protocol that generalizes it to the situation where David has a Gaussian Mixture Model instead of a single Gaussian distribution.

Protocol IV.1: Likelihood given a Gaussian Distribution
Inputs: Alice has $(d+1)$-vector $\mathbf{x}$ (with the last entry as 1). David has parameters of a Guassian $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ encoded in the form of a matrix $\mathbf{W}$ as illustrated in Figure 1. Let $\mathbf{W}^j$ represent the j-th column of matrix $\mathbf{W}$.
Computation: $\ln p(\mathbf{x}|\mathcal{N}) = \mathbf{x}^T\mathbf{W}\mathbf{x}$
Protocol:

1) For $j = 1, 2, \ldots, d+1$
   - Alice and David perform secure inner product with $\mathbf{x}$ and $\mathbf{W}^j$ to obtain shares a^j and b^j respectively, i.e. $a^j + b^j = SIP(\mathbf{x}, \mathbf{W}^j)$.
2) Alice forms vector $\mathbf{a} = [a^1, \ldots, a^{d+1}]^T$ and David forms vector $\mathbf{b} = [b^1, \ldots, b^{d+1}]^T$.
3) Alice and David perform $SIP(\mathbf{x}, \mathbf{b})$ to obtain q and r respectively.
4) Alice's share is $\mathbf{a}^T\mathbf{x} + q$ and David's share is r.
   - $\mathbf{x}^T\mathbf{W}\mathbf{x} = \mathbf{a}^T\mathbf{x} + q + r$.

Protocol IV.2: Likelihood given a GMM
Inputs: Alice has $(d+1)$-vector $\mathbf{x}$ (with the last entry as 1). David has parameters of a Guassian Mixture Model density $p(\mathbf{x})$ given by equation (4) represented as matrices $\mathbf{W}_j$, $j = 1, \ldots, J$.
Computation: From equation (7), $\ln p(\mathbf{x})$ is given by

$$\ln\left(\sum_{j=1}^{J} e^{l_j(\mathbf{x})}\right), \quad \text{where} \quad l_j(\mathbf{x}) = \mathbf{x}^T\mathbf{W}_j\mathbf{x}.$$

Protocol:

1) For $j = 1, 2, \ldots, J$
   - Alice and David engage in Protocol IV.1 with $\mathbf{x}$ and $\mathbf{W}_j$ as inputs to obtain a_j and b_j respectively.
2) Alice forms vector $\mathbf{a} = [a_1, \ldots, a_J]^T$ and David forms vector $\mathbf{b} = [b_1, \ldots, b_J]^T$.
3) Alice and David perform Secure Logsum Protocol with $\mathbf{a}$ and $\mathbf{b}$ to obtain u and v respectively, i.e. $u + v = SLOG(\mathbf{a}, \mathbf{b})$.
   - $\ln\big(\sum_{j=1}^{J} e^{l_j(\mathbf{x})}\big) = u + v$

Alice	Bob	David	Relations
$\{x_k\}$			$1 \leqslant k \leqslant K_1$
	$\{x_l\}$		$1 \leqslant l \leqslant K_2$
$\boldsymbol{\mu}^r_{iAB}$	$\boldsymbol{\mu}^r_{iAB}$	$\boldsymbol{\mu}_{iD}$	$\boldsymbol{\mu}^r_{iAB} + \boldsymbol{\mu}^r_{iD} = \boldsymbol{\mu}^r_i$
ℓ_{iAB}	ℓ_{iAB}	ℓ_{iD}	$\ell_{iAB} + \ell_{iD} = \ln P^r(\mathcal{N}_i)$
		$\boldsymbol{\Sigma}^r_i$	

Table II
For a given Gaussian $\mathcal{N}_i$, inputs held by all parties and the relations between them for Protocols given in Section V-A.

V. Privacy Preserving GMM Training

Consider a scenario where Alice has K_1 d-component vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{K_1}$ and Bob has K_2 d-component vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{K_2}$. We shall use the subscript k to denote datapoints of Alice and subscript l to denote datapoints (with the last entry as 1)of Bob. David wants to learn a mixture of c Gaussians from the combined data from Alice and Bob. Additionally, there are privacy constraints where Alice and Bob do not wish to disclose their private data to anybody and David does not wish that the parameters learned by him are disclosed to anyone.

The protocol has to let David learn the parameters by iteratively applying equations 8 and 9 on Alice and Bob's data. As in Section III-B, let $\boldsymbol{\mu}^r_i$, $\boldsymbol{\Sigma}^r_i$ and $P^r(\mathcal{N}_i)$ denote the mean vector, covariance matrix, and the mixture weight for the i-th Gaussian after the r-th iteration.

At any given iteration, Alice and Bob have access to their respective datapoints and let David have access to the parameter $\boldsymbol{\Sigma}^r_i$. All other parameters are split between David and Alice/Bob as random additive shares.

Note that the method described below is directly applicable to the case where David wants to learn parameters from data distrbuted among more than two people. For the purpose of clarity and ease of exposition, we limit the number of data owners to two without loss of any generality.

A. E-Step

The E-step update to compute $P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)$ is given by equation (8).

Protocol V.A.1: E-Step Update
Inputs: See Table II for inputs held by each party. Alice forms $\bar{\mathbf{x}}_k = (\mathbf{x}_k - \boldsymbol{\mu}^r_{iAB})$ and David forms matrix $\mathbf{W}_i$ using $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^r_i$ and $\boldsymbol{\mu} = \boldsymbol{\mu}^r_{iD}$ as illustrated in Figure 1.
Computation: $\ln P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)$ given by equation (8).
Protocol:

1) For $j = 1, \ldots, c$
   - Alice and David engage in Protocol IV.1 with $\bar{\mathbf{x}}_k$ and $\mathbf{W}_j$ as inputs to obtain $\bar{a}_j$ and $\bar{b}_j$ respectively[3].
2) Alice forms the c-vector $\mathbf{a}$ where $a_j = \bar{a}_j + \ell_{jAB}$ and David forms the c-vector $\mathbf{b}$ where $b_j = \bar{b}_j + \ell_{jD}$.
3) Alice and David engage in the Secure Logsum protocol with $\mathbf{a}$ and $\mathbf{b}$ as inputs to obtain u and v respectively, i.e. $u + v = SLOG(\mathbf{a}, \mathbf{b})$.
4) Alice obtains her share $E_k = (a_i - u)$ and David obtains his share $F_k = (b_i - v)$.
   - $E_k + F_k = P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)$.

Alice and David repeat this protocol for every data vector owned by Alice. Similarly, Bob and David can cooperate to get additive shares E_l and F_l such that $E_l + F_l = P(\mathcal{N}_i|\mathbf{x}_l, \lambda^r)$ for datapoints $\mathbf{x}_l$ owned by Bob.

B. M-Step

The M-Step equations are given by equations (9). Let Alice have $\mathbf{x}_k$, $k = \{1, \ldots, K_1\}$ and Bob have $\mathbf{x}_l$, $l = \{1, \ldots, K_2\}$. Below, we present protocols for computing each parameter.

Protocol V.B.1: Computation of Mixture Weights
Inputs: See Table III for inputs held by each party.
Computation:

$$P^{r+1}(\mathcal{N}_i) = \frac{\sum_{k=1}^{K_1} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r) + \sum_{l=1}^{K_2} P(\mathcal{N}_i|\mathbf{x}_l, \lambda^r)}{K_1 + K_2} \tag{15}$$

Protocol:

1) Alice and David engage in Secure Logsum Protocol with vectors $\mathbf{E}^A$ and $\mathbf{F}^A$ to obtain e^A and f^A, i.e. $e^A + f^A = SLOG(\mathbf{E}^A, \mathbf{F}^A)$.
   - $e^A + f^A = \ln(\sum_{k=1}^{K_1} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r))$
2) Bob and David engage in Secure Logsum Protocol with vectors $\mathbf{E}^B$ and $\mathbf{F}^B$ to obtain e^B and f^B, i.e. $e^B + f^B = SLOG(\mathbf{E}^B, \mathbf{F}^B)$.
   - $e^B + f^B = \ln(\sum_{l=1}^{K_2} P(\mathcal{N}_i|\mathbf{x}_l, \lambda^r))$
3) Alice sends e^A to Bob.
4) Bob and David perform Secure Logsum Protocol on vectors $[e^A e^B]$ and $[f^A f^B]$ respectively to obtain $\bar{e}$ and $\bar{f}$ respectively.
5) Bob's share of the desired answer is given by $\ell_{iAB} = \bar{e}$. David computes his share $\ell_{iD} = \bar{f} - \ln(K_1 + K_2)$.
   - $\ell_{iAB} + \ell_{iD} = \ln P^{r+1}(\mathcal{N}_i)$.

In the end, Bob shares $\ell_{iAB} = \bar{e}$ with Alice as well.

[3]It is easy to verify $\bar{a}_j + \bar{b}_j = \ln p(\mathbf{x}_k|\mathcal{N}_i)$. The log-likelihood is derived from equation (1) and replacing $\mathbf{x}$ by $(\mathbf{x} - \boldsymbol{\mu}_{iAB})$ and $\boldsymbol{\mu}$ by $\boldsymbol{\mu}_{iD}$ does not change the result.

Alice	Bob	David	Relations
$\{x_k\}$			$1 \leqslant k \leqslant K_1$
	$\{x_l\}$		$1 \leqslant l \leqslant K_2$
$\mathbf{E}^A$		$\mathbf{F}^A$	$E_k^A + F_k^A = \ln P(\mathcal{N}_i\|\mathbf{x}_k, \lambda^r)$
	$\mathbf{E}^B$	$\mathbf{F}^B$	$E_l^B + F_l^B = \ln P(\mathcal{N}_i\|\mathbf{x}_l, \lambda^r)$

Table III
FOR A GIVEN GAUSSIAN $\mathcal{N}_i$, INPUTS HELD BY ALL PARTIES AND THE RELATIONS BETWEEN THEM FOR PROTOCOLS GIVEN IN SECTION V-B.

Protocol V.B.2: Computation of Mean
Inputs: See Table III for inputs held by each party. In addition, for each $j = \{1, \ldots, d\}$, Alice has K_1-vector $\mathbf{h}_j^A$ formed by the j-th elements of $\mathbf{x}_1, \ldots, \mathbf{x}_{K_1}$. Similarly, Bob has the K_2-vector $\mathbf{h}_j^B$ formed by the j-th elements of his dataset.
Computation:

$$\boldsymbol{\mu}_i^{r+1} = \frac{\sum_{k=1}^{K_1} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)\mathbf{x}_k + \sum_{l=1}^{K_2} P(\mathcal{N}_i|\mathbf{x}_l, \lambda^r)\mathbf{x}_l}{\sum_{k=1}^{K_1} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r) + \sum_{l=1}^{K_2} P(\mathcal{N}_i|\mathbf{x}_l, \lambda^r)} \tag{16}$$

Protocol:
For each $j = \{1, \ldots, d\}$,

1) Alice and David engage in Secure Logsum Protocol with vectors $\mathbf{E}^A + \ln \mathbf{h}_j^A$ and $\mathbf{F}^A$ to obtain e'^A and f'^A, i.e. $e'^A + f'^A = SLOG(\mathbf{E}^A + \ln \mathbf{h}_j^A, \mathbf{F}^A)$.
2) Bob and David engage in Secure Logsum Protocol with vectors $\mathbf{E}^B + \ln \mathbf{h}_j^B$ and $\mathbf{F}^B$ to obtain e'^B and f'^B, i.e. $e'^B + f'^B = SLOG(\mathbf{E}^B + \ln \mathbf{h}_j^B, \mathbf{F}^B)$.
3) Alice sends e'^A to Bob.
4) Bob and David perform Secure Logsum Protocol on vectors $[e'^A e'^B]$ and $[f'^A f'^B]$ to obtain $\bar{e}'$ and $\bar{f}'$ respectively.
   - Notice that given $\bar{e}$ and $\bar{f}$ from Step 4 of Protocol V.B.1, $(\bar{e}' - \bar{e}) + (\bar{f}' - \bar{f}) = \ln \mu_{ij}^{r+1}$, the j-th element of $\boldsymbol{\mu}_i^{r+1}$.
5) Bob and David obtain the j-th elements of $\boldsymbol{\mu}_{iAB}^{r+1}$ and $\boldsymbol{\mu}_{iD}^{r+1}$ respectively, as a result of $SIP(exp(\bar{e}' - \bar{e}), exp(\bar{f}' - \bar{f}))$.

At the end, Bob shares $\boldsymbol{\mu}_{iAB}^{r+1}$ with Alice.

Protocol V.B.3: Computation of the Covariance Matrix
Inputs: See Table III for inputs held by each party. Also, Alice and Bob have $\boldsymbol{\mu}_{iAB}^{r+1}$ while David has $\boldsymbol{\mu}_{iD}^{r+1}$.
Computation:

$$\boldsymbol{\Sigma}_i^{r+1} = \frac{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)(\bar{\mathbf{x}}_k - \bar{\boldsymbol{\mu}}_i)(\bar{\mathbf{x}}_k - \bar{\boldsymbol{\mu}}_i)^T}{\sum_{k=1}^{K} P(\mathcal{N}_i|\mathbf{x}_k, \lambda^r)}, \tag{17}$$

where $\bar{\mathbf{x}}_k = (\mathbf{x}_k - \boldsymbol{\mu}_{iAB}^{r+1})$ and $\bar{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_{iD}^{r+1}$.
Protocol:
Consider the evaluation of σ_{mn}, the (mn)-th element of $\boldsymbol{\Sigma}_i^{r+1}$. We first consider the evaluation of the (mn)-th element of $(\mathbf{x}_k - \boldsymbol{\mu}_i^{r+1})(\mathbf{x}_k - \boldsymbol{\mu}_i^{r+1})^T$. This is equivalent to evaluating $(\bar{\mathbf{x}}_k - \bar{\boldsymbol{\mu}}_i)(\bar{\mathbf{x}}_k - \bar{\boldsymbol{\mu}}_i)^T$. Let the j-th elements of

$\bar{\mathbf{x}}_k$ and $\bar{\boldsymbol{\mu}}_i$ be $\bar{x}_{kj}$ and $\bar{\mu}_{ij}$ respectively. Notice that Alice has access to $\bar{\mathbf{x}}_k$, Bob has access to $\bar{\mathbf{x}}_l$ and David had access to $\bar{\boldsymbol{\mu}}_i$.

1) Alice and David perform the follwoing computations.
   - For $k = 1, \ldots, K_1$, Alice and David engage in the secure inner product protocol with vectors $exp(\gamma_k^A)[\bar{x}_{km}\bar{x}_{kn}, -\bar{x}_{km}, \bar{x}_{kn}, 1]$ and $[1, m\bar{u}_{in}, -\bar{\mu}_{im}, \bar{\mu}_{im}\bar{\mu_{in}}]$, where γ_k^A is a random scalar chosen by Alice. Let David obtain the result ϕ_k^A.
   - Alice forms the K_1-vector $\boldsymbol{\gamma}^A = [\gamma_1^A, \ldots, \gamma_{K_1}^A]$ and David forms the vector $\boldsymbol{\phi}^A = [\phi_1^A, \ldots, \phi_{K_1}^A]$.

   Alice and David engage in the secure logsum protocol with vectors $(\mathbf{E}^A - \boldsymbol{\gamma}^A)$ and $(\mathbf{F}^A + \ln \boldsymbol{\phi}^A)$ to obtain $\hat{e}^A$ and $\hat{f}^A$ i.e. $\hat{e}^A + \hat{f}^A = SLOG((\mathbf{E}^A - \boldsymbol{\gamma}^A), (\mathbf{F}^A + \ln \boldsymbol{\phi}^A))$.
2) Bob and David perform computations similar to the step above to obtain $\hat{e}^B$ and $\hat{f}^B$ respectively as a result of $SLOG((\mathbf{E}^B - \boldsymbol{\gamma}^B), (\mathbf{F}^B + \ln \boldsymbol{\phi}^B))$.
3) Alice sends $\hat{e}^A$ to Bob.
4) Bob and David perform Secure Logsum protocol on vectors $[\hat{e}^A\hat{e}^B]$ and $[\hat{f}^A\hat{f}^B]$ to obtain $\hat{e}$ and $\hat{f}$ respectively.
   - Notice that given $\bar{e}$ and $\bar{f}$ from Step 4 of Protocol V.B.1, $(\hat{e} - \bar{e}) + (\hat{f} - \bar{f}) = \ln \sigma_{mn}$, the (mn)-th element of $\boldsymbol{\Sigma}_i^{r+1}$.
5) Bob sends $(\hat{e} - \bar{e})$ to David so that he can calculate σ_{mn}.

All the above three M-step protocols are run in each iteration. At the end of all iterations, Bob sends his (and Alice's) shares $\boldsymbol{\mu}_{iAB}$ and ℓ_{iAB} to David so he can calculate the mean $\boldsymbol{\mu}_i$ and the mixture weight $P(\mathcal{N}_i)$ for $i = 1, 2, \ldots, c$.

In this section, we presented protocols so that David can learn parameters of a GMM using private data from Alice and Bob. The protocols presented here can be extended in a straightforward manner to cases where data is distributed among more than two people.

VI. Privacy Preserving GMM Classification

Consider a situation where Alice has a d-component vector $\mathbf{x}$ and David has N classes ω_i, $i = \{1, \ldots, N\}$ where each class is modeled as a Gaussian Mixture Model with the probability density given by equation (11). Alice would like David to classify her data but does not want to disclose to him any information about the data or the result of the classification. Also, David does not want Alice to learn any information about the parameters of his models. In this section, we present a protocol which makes the above computation possible.

Protocol VI.1: Classification

Inputs: Alice has $\mathbf{x}$. For class ω_i, $i = 1, \ldots, N$, David has encoded the GMM parameters α_{ij}, $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$ in the form of matrices $\mathbf{W}_{ij}$ as explained in the derivation of equation (6) from equation (5) for all j, $j = 1, \ldots, J_i$.

Computation: Alice learns I such that $g_I(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq I$ where $g_i(\mathbf{x})$ is defined by equation (14).

Protocol:

1) For $i = 1, 2, \ldots, N$
   - Alice and David engage in Protocol IV.2 with $\mathbf{x}$ and the set of matrices $\mathbf{W}_{ij}$, $j \in \{1, \ldots, J_i\}$ as inputs to obtain u_i and $\bar{v}_i$ respectively.
2) Alice forms the vector $\mathbf{u} = [u_1, \ldots, u_N]$ and David forms vector $\mathbf{v}$ where $v_i = \bar{v}_i + \ln P(\omega_i)$.
   - $u_i + v_i = g_i(\mathbf{x})$
3) Alice and David perform the Secure Maximum Index protocol between vectors $\mathbf{u}$ and $\mathbf{v}$ and Alice obtains $I = SMAX(\mathbf{u}, \mathbf{v})$.

VII. Discussion and Conclusions

In this paper, we have presented privacy preserving protocols that enable computations involving Gaussian Mixture Models such as log-likelihoods and parameter estimation. We first proposed protocols to compute the likelihood of a data vector given a Gaussian Mixture Model when the data and model are held by different parties with privacy constraints. We then showed how these protocols can be utilized for classification of a datapoint from a third-party service. We also presented protocols that enables a third party service to learn the parameters of a Gaussian Mixture Model from private data distributed among multiple parties.

Privacy-preserving protocols are typically evaluated for correctness, security and efficiency. It is straightforward to verify the correctness of the proposed protocols. All the protocols presented are exact and security is obtained not from statistically perturbing data but from utilizing underlying cryptographic primitives such as the Secure Inner Product and distributing intermediate results as random additive shares. Assuming that the underlying primitives are secure, a quick exercise through the steps of each protocol should demonstrate that no information leaks between parties. However, the security and efficiency of the proposed protocols depend greatly on the security and efficiency of the underlying primitives used. For example, consider the Secure Inner Product (SIP) primitive which is the most important primitive repeatedly used in all our protocols. The Secure Logsum primtive $SLOG$ is also based on one call to SIP. The quality of the SIP implementation directly affects the quality of the proposed protocols. There are two broad ways to implement SIP - (a) cryptographic protocols such as the one described in Appendix A [14] and an oblivious transfer based protocol proposed by [15], and (b) algebraic protocols such as the one based on linear transformations proposed by [16]. Typically, algebraic protocols are more efficient than cryptographic protocols in terms of computational complexity but may leak more information.

A good choice of the type of implementation to be used will have to balance tradeoffs such and computational and communication efficieny as opposed to security and privacy. We skip a detailed discussion of these issues as it is out of scope of this paper.

This work points to many interesting directions for future research. One of these is to design and implement privacy-preserving protocols for models other than mixtures of Gaussians. Our proposed approach points a principled way to achieve it if the computations can be expressed as a series of inner products. The other important direction is to work on improved primitives such as SIP and provide a better understanding of the security and efficiency tradeoffs.

We have presented these protocols as a first step towards the utilization of GMMs in privacy-preserving datamining. With advances in the cryptography community in terms of faster and more efficient cryptographic primitives, we expect the computational and communication issues to be less important factors. Our hope is that these ideas are utilized in real-world scenarios in the future. An example would be a cloud-based analytics service using Gaussian Mixture Models that also guarantees privacy for the users.

Appendix A. Secure Inner Product using Homomorphic Encryption

The following protocol is based on homomorphic encryption and was proposed by [14]. Let the triple (Ge, En, De) denote a public-key homomorphic cryptosystem (probabilistic polynomial time algorithms for key-generation, encryption and decryption). The key generation algorithm generates a valid pair (sk, pk) of private and public keys for a security parameter k. The encryption algorithm En takes as an input a plaintext m, a random value r and a public key pk and outputs the corresponding ciphertext $\texttt{En}(\texttt{pk}; m, r)$. The decryption algorithm De takes as an input a ciphertext c and a private key sk (corresponding to the public key pk) and outputs a plaintext $\texttt{De}(\texttt{sk}; c)$. It is required that $\texttt{De}(\texttt{sk}; \texttt{En}(\texttt{pk}; m, r)) = m$.

If the cryptosystem is homomorphic, we have $\texttt{En}(\texttt{pk}; m_1, r_1) \cdot \texttt{En}(\texttt{pk}; m_2, r_2) = \texttt{En}(\texttt{pk}; m_1 + m_2, r_1 + r_2)$.

Let Bob and Alice have pivate vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. The desired result is to obtain shares a and b such that $a + b = \mathbf{x}^T\mathbf{y}$. The protocol is described below.

1) Setup phase. Bob:
   - generates a private and public key pair (sk, pk).
   - sends pk to Alice.
2) For $i \in \{1, \ldots, d\}$, Bob:
   - generates a random new string r_i.
   - sends $c_i = \texttt{En}(\texttt{pk}; x_i, r_i)$ to Alice.
3) Alice:
   - sets $z \leftarrow \prod_{i=1}^{d} c_i^{y_i}$.
   - generates a random plaintext b and a random nonce r'.
   - sends $z' = z \cdot \texttt{En}(\texttt{pk}; -b, r')$ to Bob.
4) Bob computes $a = \texttt{De}(\texttt{sk}; z') = \mathbf{x}^T\mathbf{y} - b$.

Appendix B. Secure Maximum Index using Permutations

Let Alice and Bob have private data vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. They would like to compute the index of the maximum element in $\mathbf{x} + \mathbf{y}$.

Consider a permutation scheme π chosen by Alice but unknown to Bob. [17] presents a protocol that enables Alice and Bob to receive additive shares $\mathbf{q}$ and $\mathbf{s}$ of $\pi(\mathbf{x} + \mathbf{y})$ - the permutation of the vector $\mathbf{x} + \mathbf{y}$. Alice then sends $\mathbf{q} - r$, where r is a random number chosen by her, to Bob. Bob sends back the index of the maximum element of $\mathbf{q} + \mathbf{s} - r$ to Alice who then computes the real index using the inverse of the permutation π. Neither party learns the value of the maximum element and Bob does not learn the index of the maximum element.

Appendix C. Secure Logsum

Let Alice and Bob have private data vectors $\mathbf{x}$ and $\mathbf{y}$ respectively such that $\mathbf{x} + \mathbf{y} = \ln \mathbf{z}$. [10] present a protocol that results in additive shares q and s such that $q + s = \ln \sum_i z_i$. The protocol is described below.

1) Alice and Bob compute the dot product between vectors $e^{\mathbf{x}-q}$ and $e^{\mathbf{y}}$ using $SIP(e^{\mathbf{x}-q}, e^{\mathbf{y}})$ where q is a random number chosen by Alice. Let Bob obtain ϕ, the result of the dot product.
2) Notice that Bob has $s = \ln \phi = -q + \ln(\sum_{j=1}^{d} e^{x_j + y_j})$ and Alice has q.

References

[1] A. C.-C. Yao, "Protocols for secure computation," in *Proc. of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 160–164.

[2] O. Goldreich, "Secure multi-party computation," Working Draft, 2000. [Online]. Available: citeseer.ist.psu.edu/goldreich98secure.html

[3] J. Vaidya and C. Clifton, "Privacy preserving k-means clustering over vertically partitioned data," in *ACM SIGKDD Conf on Knowledge Discovery and Data Mining*, 2003. [Online]. Available: citeseer.ist.psu.edu/vaidya03privacypreserving.html

[4] E. Kiltz, G. Leander, and J. Malone-Lee, "Secure computation of the mean and related statistics," in *Proceedings of the Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, vol. 3378, 2005, pp. 283–302. [Online]. Available: http://eprint.iacr.org/2004/359.pdf

[5] S. Avidan and M. Butman, "Blind vision," in *ECCV*, 2006. [Online]. Available: http://www.merl.com/reports/docs/TR2006-006.pdf

[6] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Privacy Enhancing Technologies*, ser. LNCS, E. Goldberg and M. Atallah, Eds. Springer, 2009, vol. 5672.

[7] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "SCiFI: A System for Secure Face Identification," in *IEEE Symposium on Security and Privacy*, 2010.

[8] M. Shashanka and P. Smaragdis, "Secure sound classification: Gaussian mixture models," in *Proc. of ICASSP*, 2006. [Online]. Available: http://www.merl.com/reports/docs/TR2006-065.pdf

[9] P. Smaragdis and M. Shashanka, "A framework for secure speech recognition," in *Proc. of ICASSP*, 2006.

[10] ——, "A framework for secure speech recognition," *IEEE Trans. on Audio, Speech and Language Proc.*, vol. 15, no. 4, pp. 1404–1413, 2007.

[11] S. Laur and H. Lipmaa, "Additive conditional disclosure of secrets and applications," Cryptology ePrint Archive, Report 2005/378, 2005, http://eprint.iacr.org/. [Online]. Available: citeseer.ist.psu.edu/laur05additive.html

[12] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of Advances in Cryptology - EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592, 1999, pp. 223–238. [Online]. Available: citeseer.ist.psu.edu/paillier99publickey.html

[13] I. Damgard and M. Jurik, "A generalisation, simplification and some applications of paillier's probabilistic public-key system," in *Proceedings of the Intl. Workshop on Practice and Theory in Public Key Cryptography*, ser. Lecture Notes in Computer Science, vol. 1992, 2001, pp. 119–136. [Online]. Available: http://citeseer.ist.psu.edu/383099.html

[14] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On private scalar product computation for privacy-preserving data mining," in *Intl. Conference on Information Security and Cryptology*, ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds., vol. 2506, 2004, pp. 104–120. [Online]. Available: http://citeseer.ist.psu.edu/goethals04private.html

[15] Y.-C. Chang and C.-J. Lu, "Oblivious polynomial evaluation and oblivious neural learning," in *Advances in Cryptology, Asiacrypt '01*, ser. Lecture Notes in Computer Science, vol. 2248, 2001, pp. 369–384. [Online]. Available: citeseer.ist.psu.edu/chang01oblivious.html

[16] W. Du and Z. Zhan, "A practical approach to solve secure multi-party computation problems," in *Proceedings of New Security Paradigms Workshop*, Virginia Beach, virginia, USA, September 23-26 2002. [Online]. Available: citeseer.ist.psu.edu/du02practical.html

[17] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *Proceedings of Workshop on Privacy in the Electronic Society*, Washington, DC, USA, October 2003. [Online]. Available: http://citeseer.ist.psu.edu/atallah03secure.html